Veritas InfoScale™ 7.2 Replication Administrator's Guide - Linux

February 2020

VERITAS™

Veritas InfoScale 7.2 Replication Administrator's Guide

Last updated: 2020-02-10

Document version: 7.2 Rev 0

Legal Notice

Copyright © 2017 Veritas Technologies LLC. All rights reserved.

Veritas, the Veritas Logo, Veritas InfoScale, and NetBackup are trademarks or registered trademarks of Veritas Technologies LLC or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

This product may contain third party software for which Veritas is required to provide attribution to the third party ("Third Party Programs"). Some of the Third Party Programs are available under open source or free software licenses. The License Agreement accompanying the Software does not alter any rights or obligations you may have under those open source or free software licenses. Refer to the third party legal notices document accompanying this Veritas product or available at:

https://www.veritas.com/about/legal/license-agreements

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 Veritas Technologies LLC 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. VERITAS TECHNOLOGIES LLC 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, et seq. "Commercial Computer Software and Commercial Computer Software Documentation," as applicable, and any successor regulations, whether delivered by Veritas as on premises or hosted services. 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.

Veritas Technologies LLC

500 E Middlefield Road
Mountain View, CA 94043

http://www.veritas.com

Technical Support

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

https://www.veritas.com/support

You can manage your Veritas account information at the following URL:

https://my.veritas.com

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

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

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The document version appears on page 2 of each guide. The latest documentation is available on the Veritas website:

https://sort.veritas.com/documents

Documentation feedback

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

doc.feedback@veritas.com

You can also see documentation information or ask a question on the Veritas community site:

http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

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

https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Section 1 Getting started with Volume Replicator 18

Chapter 1 Introducing Volume Replicator 19

What is VVR? 19
Features of VVR 20
VVR terminology defined 21
Components of VVR 22
How the VVR components fit together 26
Modes of VVR replication 28
VCS agents for VVR 28

Chapter 2 Understanding how Volume Replicator works 30

How VVR processes application writes 31
How VVR uses kernel buffers for replication 32
About replication in synchronous mode 33
Data flow when reading back from the SRL 33
How data flows in VVR asynchronous mode 34
About secondary logging enabled asynchronous mode 35
About bulk transfer with secondary logging 36
How data flows in VVR asynchronous mode with secondary logging 37
How data flows in VVR synchronous mode 38
How data flows in an RDS containing multiple Secondary hosts 39
Replication in a shared disk group environment 41
The role of the logowner 42
How VVR processes a write in a shared disk group 43
How VVR processes a read in a shared disk group 46
Understanding how VVR logs writes to the SRL 47
Understanding Storage Checkpoints 49
Volume sets in VVR 51
Changing membership of an RVG and a volume set 51
Using SmartTier with VVR 53

Best practices for using SmartTier with VVR 53
Cross-platform Data Sharing in VVR 53
Understanding the VVR snapshot feature 54
About snapshots of RVGs containing volume sets 55
About the traditional snapshot feature 55
About the instant snapshot feature 56
How VVR creates instant space-optimized snapshots 57
Comparing VVR snapshot methods based on different features 60
About VVR compression 61
General functionality considerations for VVR compression 62

Chapter 3 Planning and configuring replication 63

Introduction to planning and configuring replication 63
Before you begin configuring 64
Understanding business needs 64
Understanding application characteristics 65
Choosing the mode of volume replication 66
Asynchronous mode replication considerations 66
Synchronous mode replication considerations 67
Asynchronous volume replication versus synchronous volume replication 69
Choosing latency and SRL protection 70
Planning the network 72
Choosing the network bandwidth 72
Choosing the network protocol 73
Choosing the network ports used by VVR 74
Configuring VVR in a firewall environment 75
Choosing the packet size 76
Choosing the network maximum transmission unit 77
Sizing the SRL 77
Peak usage constraint for sizing the SRL 78
Synchronization period constraint for sizing the SRL 79
Secondary backup constraint for sizing the SRL 80
Secondary downtime constraint for sizing the SRL 81
Additional factors for sizing the SRL 81
Example - Calculating SRL size for a VVR configuration 82
Best practices for setting up replication 84
How the agents for hybrid applications work 85

Chapter 4 Understanding replication settings for a Secondary 87

About replication settings for a Secondary 87
Modes of VVR replication 88
About asynchronous volume replication 88
About synchronous volume replication 89
About the synchronous attribute 89
Protecting against SRL overflow 90
About the srlprot attribute 91
VVR in a Flexible Storage Sharing environment 93
Setting up latency protection 93
About the latencyprot attribute 94
Controlling the network bandwidth used for replication 95
Choosing the VVR compression mode 96
Enabling compression 97
Disabling compression 98

Chapter 5 Configuring VVR in a VCS environment 100

Overview of how to configure VVR in a VCS environment 100
Using the primary-elect feature to choose the primary site after a site disaster or network disruption 101
Application availability in the case of a network disruption using the primary-elect feature 102
Configuring VCS global clustering so you can choose the Primary site 104
Choosing the Primary site after a site disaster or network disruption 104
Troubleshooting the primary-elect feature 106
Requirements for configuring VVR in a VCS environment 108
Best practices for setting up VVR agents 108
Generic VVR setup in a VCS environment 109
Example VVR configuration in a VCS environment 110
Example RVG configuration for a failover application 111
Example RVG configuration for a parallel application 112
Example setting up VVR in a VCS environment 114
Setting up the VVR configuration 115
Verifying the VVR replication state 118
Configuring the VVR agents for failover applications 118
Configuring the VVR agents for parallel applications 125
Configuring the agents for a bunker replication configuration 128
VCS configuration for a bunker using the STORAGE protocol 128

VCS configuration for a bunker using IP 130
How the RVGPrimary works in a bunker setup 130
Administering VCS service groups 132

Section 2 Setting up and administering VVR 134

Chapter 6 Setting up replication 135

About configuring VVR replication 135
Creating a Replicated Data Set 136
Creating a Primary RVG of an RDS 136
Adding a Secondary to an RDS 139
Changing the replication settings for a Secondary 144
Synchronizing the Secondary and starting replication 152
Methods to synchronize the Secondary 152
Using the automatic synchronization feature 154
Example for setting up replication using automatic synchronization 156
About SmartMove for VVR 157
About thin storage reclamation and VVR 157
Determining if a thin reclamation array needs reclamation 158
Starting replication when the data volumes are zero initialized 159
Example: Starting replication when the data volumes are zero initialized 159

Chapter 7 Displaying configuration information 160

Displaying RVG and RDS information 160
Displaying RDS information 161
Displaying individual RVG information 161
Interpreting RVG flag settings 162
Displaying information about data volumes and volume sets 162
Displaying data volumes in a Replicated Data Set 162
Displaying a list of data volumes 163
Displaying information about all failed data volumes 163
Displaying an individual data volume 163
Displaying a volume set 163
Displaying information about Secondaries 164
Displaying consolidated replication status 164
Displaying a list of RLINKs 171
Displaying a specific RLINK 171
Displaying the status of a Secondary 174
Displaying a list of Storage Checkpoints 176

Displaying statistics with the vrstat display commands 177
Displaying the consolidated statistics 177
Displaying the RLINK information for all the hosts in the RDS 177
Displaying information about all the data volumes for all the hosts in the RDS 179
Displaying information about the SRL volumes for all the hosts in the RDS 180
Displaying information about the memory tunable parameters for all the hosts in the RDS 181
Determining VVR network bandwidth usage and compression ratio 182
Collecting consolidated statistics of the VVR components 183
Understanding how VVR stores the statistics 185
Displaying network performance data 186
Displaying extended replication statistics 187
Identifying the most up-to-date Secondary 189
VVR event notification 192

Chapter 8 Administering Volume Replicator 195

Administering data volumes 195
Associating a volume to a Replicated Data Set 196
Associating a volume set to an RDS 201
Associating a Data Change Map to a data volume 205
Resizing a data volume in a Replicated Data Set 208
Renaming a data volume in a Replicated Data Set 211
Dissociating a data volume from its Replicated Data Set 211
Mapping the name of a Secondary data volume to a differently named Primary data volume 214
Mapping disk groups 216
Administering the SRL 217
Protecting from SRL overflow 217
Incrementally synchronizing the Secondary after SRL overflow 218
Changing the size of the SRL on the Primary and the Secondary 223
Decreasing the size of the SRL on the Primary 225
Administering replication 227
Changing the replication settings 227
Pausing and resuming replication to a Secondary 227
Stopping replication to a Secondary 228
Changing the IP addresses used for replication 229

Changing the network ports used for replication 236
Administering the Replicated Data Set 240
Removing a Secondary RVG from a Replicated Data Set 241
Removing a Primary RVG from a Replicated Data Set 242
Administering Storage Checkpoints 243
Creating Storage Checkpoints 244
Ending Storage Checkpoints 244
Viewing Storage Checkpoints 245
Deleting Storage Checkpoints 245
Creating RVG snapshots 245
Using the instant snapshot feature 246
About instant full snapshots 247
About instant space-optimized snapshots 252
About instant plex-breakoff snapshots 257
Administering snapshots 259
Using the traditional snapshot feature 265
Using Veritas Volume Manager FastResync 270
Verifying the DR readiness of a VVR setup 271
Performing a failover 272
Performing a fire drill 272
Verifying the data on the Secondary 273
Backing up the Secondary 279
Restoring the Secondary RLINK from a Secondary Storage Checkpoint 279
Restoring the Secondary from online backup 280

Chapter 9 Using VVR for off-host processing 283

About using VVR for off-host processing 283
What is off-host processing? 284
In-Band Control Messaging overview 284
How to use the data on the Secondary 284
In-BandControl Messaging explained 285
Performing off-host processing tasks 288
Tasks to perform for off-host processing 288
Using the IBC messaging command vradmin ibc 289
Examples of off-host processing 294
Example - Decision support using the snapshot feature and the vradmin ibc command 294
Example - Backing up using the snapshot feature and the vradmin ibc command 296
Example - Performing block-level backup of the Secondary data using the vradmin ibc command 297

Chapter 10 Transferring the Primary role 299

About transferring the Primary role 299
Migrating the Primary 300
Prerequisites for migrating the Primary 302
Important notes for migrating the Primary role 303
Example - Migrating from a healthy Primary 304
Example - Migrating the Primary role in a setup with multiple Secondaries 305
About taking over from an original Primary 308
Important notes about taking over from an original Primary 310
Example -Taking over from an original Primary 313
Example - Taking over from an original Primary in a setup with multiple Secondaries 314
Failing back to the original Primary 316
Fast failback versus difference-based synchronization 317
Failing back using fast failback synchronization 317
Failing back using difference-based synchronization 323
About choosing the Primary site after a site disaster or network disruption 326
Application availability in the case of a network disruption 327
Configuring VCS global clustering so you can choose the Primary site 329
Choosing the Primary site after a site disaster or network disruption 329
Example - Choosing the original Primary as the Primary going forward 330
Example - Choosing the original Secondary as the Primary going forward 331
Troubleshooting the primary-elect feature 331
Troubleshooting failures in the RVGPrimary online agent function 332
Troubleshooting failures in VVR ElectPrimary command 332
Primary-elect configuration limitations 333

Chapter 11 Replication using a bunker site 334

Introduction to replication using a bunker site 334
About replication using a bunker site during normal operations 335
How the bunker site is used for disaster recovery 336
Best practices for setting up replication using a bunker site 338
Sample bunker configuration 338
Setting up replication using a bunker site 339

Requirements for replication using a bunker site 339
Best practices for setting up replication using a bunker site 340
Adding a bunker to an RDS 340
Changing replication settings for the bunker Secondary 344
Starting replication to the bunker 346
Reinitializing the bunker 346
Administering replication using a bunker site 347
Using a bunker for disaster recovery 347
Updating the Secondary from the bunker 347
Restoring the original Primary in a bunker setup 349
Replication using a bunker site in a VCS environment 352
Automating local cluster failover for a bunker 352
About bunker replay in a VCS environment 353
Removing a bunker 354
About bunker commands 354

Chapter 12 Troubleshooting VVR 356

Recovery from RLINK connect problems 356
Recovery from configuration errors 359
Errors during an RLINK attach 359
Errors during modification of an RVG 363
Recovery on the Primary or Secondary 368
About recovery from a Primary-host crash 368
Recovering from Primary data volume error 368
Primary SRL volume error cleanup and restart 371
Primary SRL volume error at reboot 372
Primary SRL volume overflow recovery 372
Primary SRL header error cleanup and recovery 373
Secondary data volume error cleanup and recovery 374
Secondary SRL volume error cleanup and recovery 375
Secondary SRL header error cleanup and recovery 376
Secondary SRL header error at reboot 378

Chapter 13 Tuning replication performance 379

Overview of replication tuning 379
SRL layout 379
How SRL affects performance 379
Striping the SRL 380
Choosing disks for the SRL 380
Mirroring the SRL 380
Tuning Volume Replicator 380
VVR buffer space 381

DCM replay block size 390
Heartbeat timeout 390
Memory chunk size 390
UDP replication tuning 391
Tuning the number of TCP connections 391
Message slots on the Secondary 392
VVR and network address translation firewall 393
Tuning VVR compression 393

Section 3 Getting started with File Replicator 396

Chapter 14 Introducing File Replicator 397

About File Replicator 397
Features of VFR 398
File Replicator limitations 398
How File Replicator works 399
About error recovery after a site disaster or network disruption using VFR 400
About File Replicator log files 400

Chapter 15 Administering File Replicator 402

About `vfradmin` utility 403
Starting the vxfstaskd and vxfsrepld file replication daemons 403
Protecting a target file system 405
Creating a file replication job 405
Creating a consistency group 407
Managing a file replication job 408
Displaying file replication job information 409
Displaying file replication jobs 410
Displaying the status of a file replication job 410
Displaying file replication job statistics 412
Displaying a file replication Storage Checkpoint 412
Displaying consistency groups 413
Modifying a file replication job 413
Modifying a consistency group 416
Working with pattern lists 418
Deleting a file replication job 420
Deleting a consistency group 421
Performing a VFR switchover 421
Performing a VFR failover after a disaster 424
Recovering a failed site if the failed source node comes up again 426

Recovering a failed site if a new node is assigned as the target 428

Section 4 Analyzing your environment with Volume Replicator Advisor 430

Chapter 16 Introducing Volume Replicator Advisor (VRAdvisor) 431

Audience 431
Related Veritas InfoScale documents 432
Overview of VRAdvisor 432
How VRAdvisor works 433
Data collection 433
Data analysis 433
What-if analysis 434

Chapter 17 Collecting the sample of data 435

About collecting the sample of data 435
Best practices for collecting the sample of data 435
Collecting the sample of data on UNIX 436
Prerequisite for collecting the sample of data 436
Supported locales 436
Collecting data using the VRAdvisor wizard 437
Collecting data using the vxstat command 440
Collecting data using the data collection script 440
Examples of collecting data with the data collection script 443
Collecting the sample of data on Windows 444
Prerequisite for collecting the sample of data 444
Collecting the sample of data using the VRAdvisor wizard 444
Collecting the sample of data using the diskStats command 445

Chapter 18 Analyzing the sample of data 447

About analyzing the sample of data 447
Prerequisites for analyzing the sample of data 448
Launching the VRAdvisor wizard 448
Analyzing the collected data 448
Specifying the data for analysis 449
Specifying the parameters for analysis 452
Understanding the results of the analysis 453
Viewing the analysis results 453
Recalculating the analysis results 455

Recording and viewing the results 459

Chapter 19 Installing Volume Replicator Advisor (VRAdvisor) 461

VRAdvisor System requirements 461
Installing VRAdvisor on Solaris 462
Uninstalling VRAdvisor on Solaris 462
Installing VRAdvisor on Windows 462
Uninstalling VRAdvisor on Windows 463

Section 5 VVR reference 465

Appendix A VVR command reference 466

VVR command reference 466

Appendix B Using the In-band Control Messaging utility vxibc and the IBC programming API 477

About the IBC messaging utility vxibc 477
In-band Control Messaging overview 478
Using the IBC messaging command-line utility 479
Registering an application name 479
Displaying the registered application name for an RVG 479
Receiving an IBC message 480
Sending an IBC message 480
Unfreezing the Secondary RVG 481
Unregistering an application name 481
Receiving and processing an IBC message using a single command 482
Sending and processing an IBC message using a single command 482
Examples—Off-host processing 483
Example 1—Decision support using the traditional snapshot feature and the vxibc utility 484
Example 2—Backing up using the snapshot feature and the vxibc utility 485
Example 3—Trial failover using the snapshot feature 486
Example 4—Decision support using the instant full snapshot feature and the vxibc utility 487
In-band Control Messaging API 488
IOCTL commands 489

Using the IBC API 495

Appendix C Volume Replicator object states 496

Volume Replicator Kernel State 496
RVG KSTATEs 496
RLINK KSTATEs 497
Volume Replicator utility states 497
RVG utility states 497
RLINK utility states 498
Inactive RLINKs 498
STALE RLINK state 499
FAIL RLINK state 499
Inconsistent RLINKs 500
Pausing, resuming, and restoring RLINK states 501

Appendix D Alternate methods for synchronizing the Secondary 502

Using the full synchronization feature 502
Example—Synchronizing the Secondary using full synchronization with Storage Checkpoint 504
Using block-level backup and Storage Checkpoint 505
Example—Synchronizing the Secondary using block-level backup 506
Using the Disk Group Split and Join feature 507
Using difference-based synchronization 509
Example—synchronizing the Secondary based on differences 510
Examples for setting up a simple Volume Replicator configuration 511
Creating a Replicated Data Set for the examples 512
Example for setting up replication using full synchronization 513
Example for setting up replication using block-level backup and checkpointing 514
Example for setting up replication using Disk Group Split and Join 515
Example for setting up replication using differences-based synchronization 518
Example for setting up replication when data volumes are initialized with zeroes 518

Appendix E Migrating VVR from Internet Protocol version 4 to Internet Protocol version 6 520

Overview of VVR migration from IPv4 to IPv6 520
About migrating to IPv6 when VCS global clustering and VVR agents are not configured 521
Understanding the current IPv4 configuration when VCS global clustering and VVR agents are not configured 521
Migration prerequisites when VCS global clustering and VVR agents are not configured 524
Migrating to IPv6 when VCS global clustering and VVR agents are not configured 524
About migrating to IPv6 when VCS global clustering and VVR agents are configured 533
Understanding the current IPv4 configuration when VCS global clustering and VVR agents are configured 533
Migration prerequisites when VCS global clustering and VVR agents are configured 535
Migrating to IPv6 when VCS global clustering and VVR agents are configured 536
Migrating the VCS global clustering service group to IPv6 when VCS global clustering and VVR agents are configured 536
Adding IP and NIC resources for IPv6 addresses in the RVG agent group when VCS global clustering and VVR agents are configured 539
Migrating VVR RLINKs from IPv4 to IPv6 when VCS global clustering and VVR agents are configured 541
Removing the IPv4 resources from the VCS configuration when VCS global clustering and VVR agents are configured 542
About migrating to IPv6 when VCS global clustering and VVR agents are configured in the presence of a bunker 542
Understanding the current IPv4 configuration when VCS global clustering and VVR agents are configured in the presence of a bunker 542
Migration prerequisites when VCS global clustering and VVR agents are configured in the presence of a bunker 546
Migrating to IPv6 when VCS global clustering and VVR agents are configured in the presence of a bunker 546

Appendix F Sample main.cf files 569

Globally clustered VCS and VVR main.cf 569

Index 574

Section 1

Getting started with Volume Replicator

- Chapter 1. Introducing Volume Replicator
- Chapter 2. Understanding how Volume Replicator works
- Chapter 3. Planning and configuring replication
- Chapter 4. Understanding replication settings for a Secondary
- Chapter 5. Configuring VVR in a VCS environment

Chapter 1

Introducing Volume Replicator

This chapter includes the following topics:

- What is VVR?
- Components of VVR
- Modes of VVR replication
- VCS agents for VVR

What is VVR?

Volume Replicator (VVR) is data-replication software designed to contribute to an effective disaster recovery plan. VVR enables you to maintain a consistent copy of application data at one or more remote locations.

VVR is an option of Veritas Volume Manager (VxVM) that works as a fully integrated component of VxVM. VVR benefits from the robustness, ease of use, and high performance of VxVM, and at the same time, adds replication capability to VxVM. VVR can replicate existing VxVM configurations, and can be transparently configured while the application is active.

VVR is a separately licensed option of VxVM. You can start using VVR in a VxVM configuration by installing a valid VVR license.

VVR replicates the application writes on the volumes at the source location to one or more remote locations across any distance. It provides a consistent copy of application data at the remote locations. If a disaster occurs at the source location, you can use the copy of the application data at the remote location and restart the application at the remote location.

The host at the source location on which the application is running is known as the Primary host, and the host at the target location is known as the Secondary host. You can have up to 32 Secondary hosts in a VVR environment.

The volumes on the Primary host must be initially synchronized with the volumes on the Secondary host. VVR provides several methods to initialize the application data between the primary location and the remote location, such as using the network, using tape backup, and moving disks physically.

Features of VVR

Volume Replicator (VVR) includes the following features:

- Can be configured to work with any storage hardware supported by Veritas Volume Manager.
- Replicates data for up to 32 remote locations over any IP network in a LAN or WAN environment.
- Performs replication of volume groups in asynchronous or synchronous modes, ensuring data integrity and consistency in both modes.
- Maintains write-order fidelity, which applies writes on the Secondary host in the same order that they were issued on the Primary host.
- Enables you to easily recover your application at the remote site.
- Provides effective bandwidth management using bandwidth throttling and multiple connections.
- Provides the ability to perform off-host processing such as Decision Support Systems (DSS) and backup, by enabling you to break off a consistent mirror or snapshot of the data volumes on the Secondary to use for these operations.
- Provides the command-line interface and the graphical user interface for online management of the VVR environment.
- Provides multiple methods to synchronize the data at the Secondary location with the data at the Primary location.
- Easily accommodates growth of application data and system configurations.
- Supports cross-platform replication, that is, the Primary and Secondary can have different operating systems.
- Supports volume-level replication of application or file system data, which includes support for all commercial database management systems, such as Oracle, DB2, Sybase, and Informix.
- Supports volume-level replication of data in a shared storage environment, for use with parallel applications, such as Oracle RAC (Real Application Cluster).

- Supports replication of VxVM volume sets, including ensuring consistency between the component volumes of the volume set on the Primary and on the Secondary.
- Supports replication in a PDC (Portable Data Container) environment.
- Supports different volume layouts on Primary and Secondary hosts. For example mirrored layout is supported on Primary hosts and non-mirrored layout is supported on Secondary hosts.
- Provides the ability to validate data on the Secondary without application downtime or stopping replication.
- Provides a configuration check utility, `/etc/vx/diag.d/vvrcheck`, that displays current replication status, detects and reports configuration anomalies, and creates statistics files that can be used by display tools.
  For more information, see the `vvrcheck`(`1M`) man page.

VVR terminology defined

Table 1-1 defines common VVR terminology.

Table 1-1 VVR terminology defined

Write-order fidelity	To use the Secondary in a disaster recovery scenario, write-order fidelity must be maintained. The term write-order fidelity means that VVR tracks writes on the Primary in the order in which they are received and applies them on the Secondary in the same order. It is important to maintain write-order fidelity to ensure that the data on the Secondary is consistent with the data on the Primary. While the data at the Secondary can be behind in time, it must be a consistent image of the Primary RVG at a point in the past. Without write order fidelity, there is no guarantee that a Secondary has consistent, recoverable data. VVR maintains write-order fidelity regardless of the mode of replication and across all the data volumes in an RVG. For example, in a database environment, the log and data space are typically on different volumes. On the Primary, VVR applies writes to the log and data spaces in a fixed order and maintains this fixed order when applying the writes on the Secondary. If write-order fidelity is not maintained, a database application may not recover successfully when failed over to the Secondary.

Table 1-1 VVR terminology defined *(continued)*

Consistent data versus current or up-to-date data	Data is consistent if the system or application using it can be successfully restarted to a known, usable state. The data on the Secondary is consistent if it correctly reflects the data on the Primary at some point in the past. At all times, VVR maintains the data at the Secondary in a consistent state with the data at the Primary. For example, if the data being replicated is used by a database, the data is consistent if the database can be started and recovered to a usable state with zero data corruption. If the data contains a file system, the data is consistent if the file system check utility can be run and it can recover with no file system corruption. Data is considered consistent only if it contains all updates up to a point in time and none of the updates that come after that point. For example, if it is a file system, the most recently created files may be missing when it is restarted. Or, if it is a database, one or more of the most recently committed transactions might be missing. Data that is current or up-to-date contains the latest changes made at the Primary. For example, if you are replicating a database, the most recent transaction is available at the Secondary. Whether or not the data on the Secondary must always be current is a business decision and can be controlled by choosing between synchronous and asynchronous modes of replication.
IPv4-only node	A node that implements only IPv4. An IPv4-only node does not understand IPv6. The current installed base of IPv4 nodes and routers are IPv4-only node. IPv4-only node is one that only has an IPv4 address in the name service database.
IPv6-only node	A node that implements only IPv6 and only has IPv6 addresses in the name service database.
Dual-node / Dual-stack	A node that implements both IPv4 and IPv6. It is expected that the nodes that are upgraded from IPv4-only will be upgraded to dual nodes. This is also called an IPv4/IPv6 node.This does not mean that the node has an IPv6 configured interface, but only indicates that the node has IPv6 support enabled.
IPv6-enabled node	A node that implements a dual node and has at least one IPv6 interface configured. This node would have both IPv4 and IPv6 addresses in the respective name services database.

Components of VVR

Table 1-2 lists the various components of VVR.

Table 1-2 Components of VVR

Components	Description
Replicated Volume Group (RVG)	A Replicated Volume Group (RVG) is a group of volumes within a given VxVM disk group configured for replication. The volumes on the Secondary RVG are consistent while replicating data due to write-order fidelity. An RVG is always a subset of a VxVM disk group. One or more related volumes in a disk group can be configured as an RVG. By related volumes, we mean a set of volumes to which application writes must be replicated in order on the Secondary. In the case of a database, several processes perform writes to disks. Database processes write in a specific order. This order must be maintained at all times including when recovering from a disk failure. For example, the database posts any database change to the log before writing to the table space. To convey to VVR that these two volumes are related, these two volumes must be grouped. All related volumes must be part of the same disk group. Unrelated volumes should not be grouped together in an RVG. Multiple RVGs can be configured inside one disk group, although this is not a recommended configuration. Volumes that are associated with an RVG and contain application data are called data volumes. The data volumes in the RVG are under the control of an application, such as a Database Management System, that requires write-order fidelity among the writes to the volumes. Write-ordering is strictly maintained within an RVG during replication to ensure that each remote volume is always consistent, both internally and with all other volumes of the group. Each RVG can have a maximum of 2048 data volumes. VVR replicates data from a Primary RVG, on the host where the application is running, to the Secondary RVG. An RVG also contains the Storage Replicator Log (SRL) and Replication Link (RLINK), which are used internally by VVR. **Note:** A Primary RVG can have multiple Secondary RVGs. When this document refers to the Secondary host, it implicitly refers to all the Secondary RVGs.

Table 1-2 Components of VVR *(continued)*

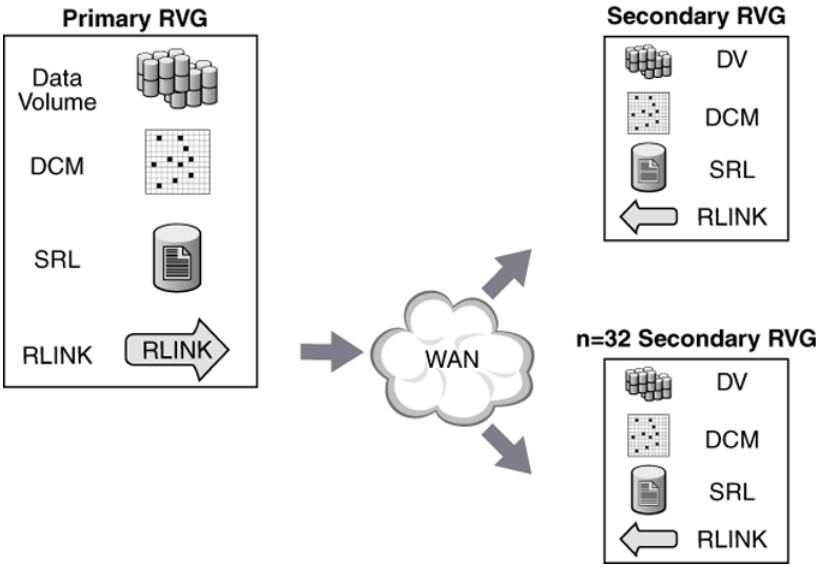
Components	Description
Storage Replicator Log (SRL)	The Storage Replicator Log (SRL) is a circular buffer of writes for an RVG. Each RVG contains one SRL. Writes to the data volumes in the RVG are first queued in the SRL on the Primary host before they are sent to the Secondary. VVR uses the SRL to track the order of writes to data volumes in the RVG. The SRL enables VVR to maintain write-order fidelity at the Secondary RVG. In addition to the replication functionality, the SRL provides the functionality provided by the DRL (Dirty Region Log). Therefore, VxVM DRL logging is explicity disabled when a volume is added to an RVG. From a VxVM perspective, the SRL is just another volume. Because all writes are written to the SRL first, it is important for the SRL to have optimum write performance. This means all performance techniques used to increase write performance of a volume apply to the SRL. For most implementations, the SRL is striped across multiple drives for write performance, and mirrored to an equal set of drives for protection. Each write to the disks generates two writes: one to the SRL, and one to a data volume. For this reason, it is recommended that the data volumes and SRL volumes should be located on different physical disks or RAID sets to improve write performance. Note that VVR uses the SRL log internally. Application writes directly to the SRL are not allowed.
Replication Link (RLINK)	An RLINK is associated with an RVG and establishes the link between the Primary and a Secondary RVG. Each RLINK associated with a Primary RVG represents one Secondary. Each RLINK associated with a Secondary RVG represents the Primary. The attributes of an RLINK specify the replication parameters for the corresponding Secondary. For example, the network to be used for replication between the Primary and the Secondary can be specified as the attribute of the RLINK. A Primary RVG can have up to 32 associated RLINKs. Although a Secondary RVG can also have 32 associated RLINKs, it can have only one active RLINK; this active RLINK represents the Primary that is currently replicating to this Secondary RVG.

Table 1-2 Components of VVR *(continued)*

Components	Description
Data Change Map (DCM)	The Data Change Map (DCM) is a component of VVR that is used to track writes when the SRL overflows and thus enables you to avoid complete resynchronization of the data on the Secondary. The DCM is active only on the Primary side. The DCM becomes active only when the SRL is no longer large enough to hold accumulated updates. While the DCM is active, each bit that has been set in the DCM represents a region whose contents are different between the Primary and the Secondary. At an appropriate time, the administrator initiates a resynchronization and causes VVR to incrementally synchronize the Secondary with the Primary by looking up the bitmap. When the resynchronization of the DCM starts, the Secondary becomes inconsistent because the DCM resynchronization writes are not necessarily in the same order as the application writes. As a result, the Secondary cannot be used for disaster recovery while the DCM is resynchronizing. After the resynchronization of the DCM is complete, the Secondary RVG is consistent and replication resumes as usual. The Automatic Synchronization, SRL Overflow Protection with DCM, and Fast Failback features use the DCM. Each data volume in the RVG must have a valid DCM associated with it before the DCM can be used.
Replicated Data Set (RDS)	A Replicated Volume Group (RVG) on the Primary host and its counterparts on the Secondary hosts make up a Replicated Data Set (RDS). An RDS is not a Volume Manager object but a concept used in VVR. An RDS enables grouping of the RVG on the Primary and its counterparts on the Secondaries. Most VVR commands operate on an RDS, that is, the Primary RVG and all the Secondaries in the RDS. You can issue VVR commands from any host in an RDS unless otherwise noted. VVR performs the appropriate tasks on the required hosts in the RDS. The concept of Primary host and Secondary host is used only in the context of a particular Replicated Data Set (RDS). A system can simultaneously be a Primary host for some RDSs and Secondary host for others. This allows for very flexible replication configurations.

Figure 1-1 shows the VVR components for a sample configuration.

Figure 1-1 Sample configuration to illustrate the VVR components



How the VVR components fit together

This section describes how the VVR components fit together to enable replication as follows:

- See "VVR at the Primary" on page 26.
- See "VVR at the Secondary" on page 27.
- See "Local host (localhost)" on page 27.

VVR at the Primary

VVR is configured such that the volumes to be replicated for a specific application are placed in an RVG. Writes to the data volumes are persistently queued in the SRL. The SRL on the Primary tracks all writes in the order in which they were received and transmits the writes to the Secondary using a replication link (RLINK).

On the Primary, each write to an RVG generates two writes: one to a data volume, and one to the SRL. While VVR generates two writes, only the write to the SRL affects the application. The write to the SRL is a fast write to a sequentially accessed log while the data volume write is a normal write performed asynchronously. The write to the data volume is not in the critical path for the application. The write to the data volume is written in the background and does not affect application performance.

If the Primary crashes at any point after the write completes to the SRL and before the write completes to the data volume, data is fully recoverable from the SRL. This is very similar to a database writing to a redo log and later writing to the data files. This two phase write gives VVR the ability to maintain write-order fidelity at the Secondary.

VVR at the Secondary

Writes are sent to the Secondary in the order in which they are received at the Primary. VVR sends data to the Secondary RVG as a message encompassing an application write. This means VVR sends messages based on application write sizes. When the Secondary receives the message in VVR kernel memory, the Secondary immediately sends an initial acknowledgment of receipt. This is known as the network acknowledgment. The network acknowledgment allows the Primary to immediately continue processing, as required. The data is not yet written to disk on the Secondary RVG, but it is still safe because it is stored in the Primary SRL. After the Secondary writes to the local disk (either the Secondary SRL if configured, or to the secondary data volume), it sends the second acknowledgment, the data acknowledgment.

The reason for the two-phase acknowledgment is so that VVR can maintain application performance when it is configured in synchronous mode. If VVR were to wait for the write to complete on the Secondary as well as the Primary, it would increase latency considerably. Instead, the Primary waits for the network acknowledgment from the Secondary before it completes the write at the application. Because data is persistently queued in the Primary SRL, safety of the data for the Secondary is maintained.

VVR receives a packet into memory on the Secondary RVG, holds the packet until all the previous packets have been received, then writes to the data volumes in the correct sequence to maintain consistency at the Secondary. The writes to the data volumes use the Secondary SRL as a staging location if logging is enabled on the RVG. Secondary logging may be automatically disabled if all requirements for the feature are not met. Holding the packets in memory enables VVR to reassemble out-of-order network traffic before writing, and discover and handle missing packets. To maintain consistency at the Secondary RVG, VVR never writes an I/O out of order with the Primary RVG. VVR serializes and checksums incoming data from the Primary RVG to support accurate replay to the Secondary volumes.

See “About secondary logging enabled asynchronous mode” on page 35.

Local host (localhost)

The host from which a command is issued is called the local host. The name of the Replicated Volume Group (RVG) on the local host represents the RDS. For example,

to add a data volume to an RDS, issue the command from any host in the RDS, using the name of the RVG on that host to specify the RDS; VVR adds a data volume to the corresponding RVGs on all the hosts in the RDS.

Modes of VVR replication

VVR replicates in synchronous and asynchronous modes. The decision to use synchronous or asynchronous mode must be made with an understanding of the effects of this choice on the replication process and the application performance.

See "Modes of VVR replication" on page 88.

Asynchronous Replication	Asynchronous mode is useful when it is acceptable for the Secondary to not be up-to-date. When replicating in asynchronous mode, an update to the Primary volume is complete when it has been recorded in the Primary SRL. Asynchronous mode does not guarantee the data is current at all times, but it has less impact on application performance and provides the ability to use more cost-effective telecommunications. All completed updates to the Primary volumes are guaranteed to be made on the Secondary data volumes with some delay.
Synchronous Replication	Synchronous mode ensures that a write has been posted to the Secondary and the Primary before the write completes at the application level. When replicating in synchronous mode, the data on the Secondary is completely up-to-date and if a disaster occurs at the Primary, data can be recovered from any surviving Secondary without any loss. If the Secondary must reflect all writes that have successfully completed on the Primary, synchronous mode is the correct choice. Synchronous replication provides data currency but can impact application performance in high latency or limited bandwidth environments. The response time experienced by the application is affected because the write has to wait for the Secondary to acknowledge it before the write can complete on the Primary.

VCS agents for VVR

VCS provides agents that manage applications and resources in a cluster.

The different types of agents follow:

- VCS comes packaged (bundled) with a set of agents that enable VCS to provide high availability. These include agents for mount points, IP addresses, file systems, VVR, and virtual environments. These agents are immediately available to you after installing VCS.

  For more information about VCS bundled agents, refer to the *Cluster Server Bundled Agents Reference Guide*.

Agents are processes that manage predefined resource types. When an agent is started, it obtains configuration information from VCS. It then periodically monitors the resources and updates VCS with the resource status.

Typically agents do the following:

- Bring resources online
- Take resources offline
- Monitor resources and report any state changes to VCS

Chapter 2

Understanding how Volume Replicator works

This chapter includes the following topics:

- How VVR processes application writes
- How VVR uses kernel buffers for replication
- How data flows in VVR asynchronous mode
- About secondary logging enabled asynchronous mode
- About bulk transfer with secondary logging
- How data flows in VVR asynchronous mode with secondary logging
- How data flows in VVR synchronous mode
- How data flows in an RDS containing multiple Secondary hosts
- Replication in a shared disk group environment
- Understanding how VVR logs writes to the SRL
- Understanding Storage Checkpoints
- Volume sets in VVR
- Changing membership of an RVG and a volume set
- Using SmartTier with VVR
- Cross-platform Data Sharing in VVR
- Understanding the VVR snapshot feature

- About VVR compression

How VVR processes application writes

This section helps you understand how application writes are directed when VxVM is not being used, when VxVM is added, and when VVR is added.

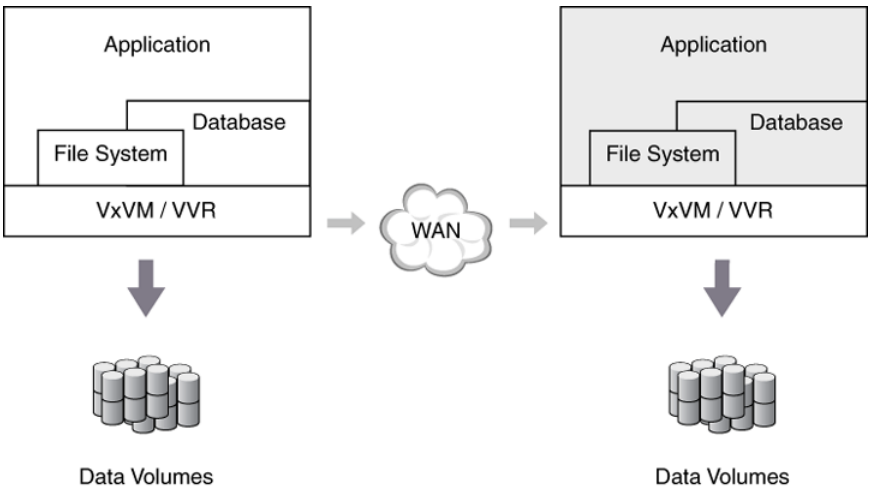
When VxVM is not being used, the application writes to a file system placed on a disk partition. In the case of applications or databases on raw devices, the database writes directly to the disk partition, instead of to a file system. In either case, the application, that is, the database or a file system, sends data to the operating system to be written to disk and the operating system communicates directly with the disks.

When VxVM is being used, the applications write to logical devices called volumes, rather than physical disks. A volume is a virtual disk device that appears as a physical disk to applications, such as databases and file systems. However, a volume does not have the limitations of a physical disk.

When VVR is added, it resides between the application and the underlying VxVM volumes. All writes to these replicated volumes are intercepted and replicated to the Secondary host in the order in which they were received at the Primary. Writes are also applied to the local volumes. However, reads are directly processed using the local volumes.

Figure 2-1 shows how application writes are processed when VxVM and VVR are used.

Figure 2-1 How application writes are processed when VxVM and VVR are used



VVR sends writes to the Secondary in the order in which they were received on the Primary. The Secondary receives writes from the Primary and writes to local volumes.

While replication is active, you should not use the application directly on the data volumes on the Secondary. The application on the Secondary is used only if a disaster occurs on the Primary. If the Primary fails, the application that was running on the Primary can be brought up on the Secondary, and can use the data volumes on the Secondary.

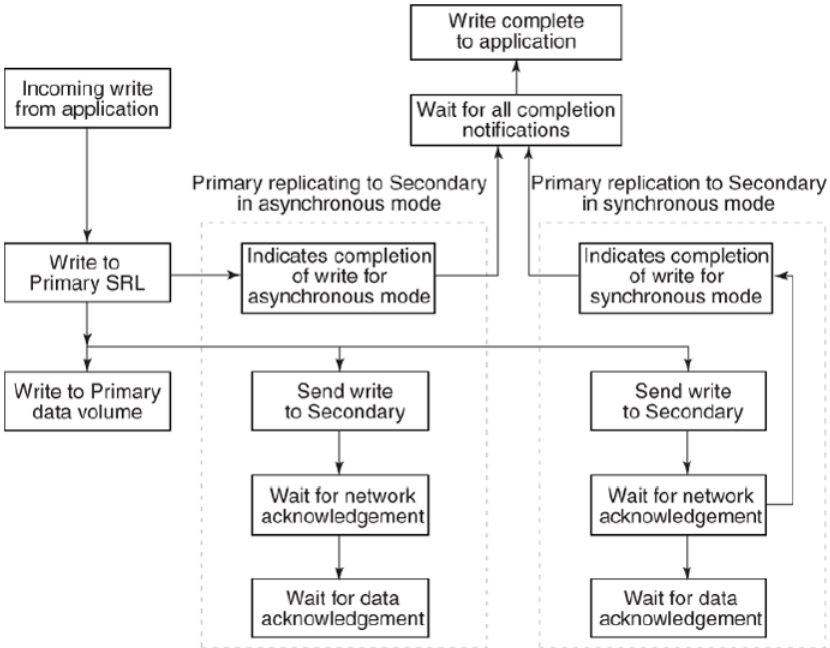
To use the data on the Secondary while the Primary is active, use the snapshot feature to create a version of the data that is not being changed.

How VVR uses kernel buffers for replication

This section explains how data flows in VVR and how VVR uses the kernel buffers for replication.

Figure 2-2 shows the flow of data for a VVR configuration containing two Secondary hosts with the Primary replicating to one host in asynchronous mode and the other host in synchronous mode.

Figure 2-2 Data flow with multiple Secondary hosts



When a write is performed on a data volume associated with a Replicated Volume Group (RVG), VVR copies the data into a kernel buffer on the Primary. VVR then writes a header and the data to the SRL; the header describes the write.

From the kernel buffer, VVR sends the write to all Secondary hosts and writes it to the Primary data volume. Writing the data to the Primary data volume is performed asynchronously to avoid adding the penalty of a second full disk write to the overall write latency. Until the data volume write to the Primary is complete, the kernel buffer cannot be freed.

About replication in synchronous mode

For all Secondary hosts replicating in synchronous mode, VVR first sends the write to the Primary SRL. VVR then sends the write to the Secondary hosts and waits for a network acknowledgment that the write was received. When all Secondary hosts replicating in synchronous mode have acknowledged the write, VVR notifies the application that the write is complete. The Secondary sends the network acknowledgment as soon as the write is received in the VVR kernel memory on the Secondary. The application does not need to wait for the full disk write, which improves performance. The data is subsequently written to the Secondary data volumes. When the write is completed on the Secondary data volumes, VVR sends a data acknowledgment back to the Primary.

For all Secondary hosts replicating in asynchronous mode, VVR notifies the application that the write is complete after it is written to the Primary SRL. Therefore, the write latency consists of the time to write to the SRL only. VVR then sends the write to the Secondary hosts. The Secondary sends a network acknowledgment to the Primary as soon as the write is received in the VVR kernel memory on the Secondary. When the write is completed on the Secondary data volumes, VVR sends a data acknowledgment back to the Primary.

The application considers the write complete after receiving notification from VVR that the data is written to the Primary SRL, and, for any Secondary hosts replicating in synchronous mode, that the write has been received in the kernel buffer. However, VVR continues to track the write until the data acknowledgment is received from all the Secondary hosts. If the Secondary crashes before writing to the data volumes on the Secondary or if the Primary crashes before it receives the data acknowledgment, the write can be replayed from the SRL.

Data flow when reading back from the SRL

A Secondary in asynchronous mode might be out of date for various reasons, such as network outages or a surge of writes which exceed available network bandwidth. As the Secondary falls behind, the data to be sent to the Secondary starts accumulating in the write-buffer space on the Primary. If the Secondaries in

asynchronous mode cannot keep up with the application write rate, VVR might need to free the Primary kernel buffer, so that incoming write requests are not delayed.

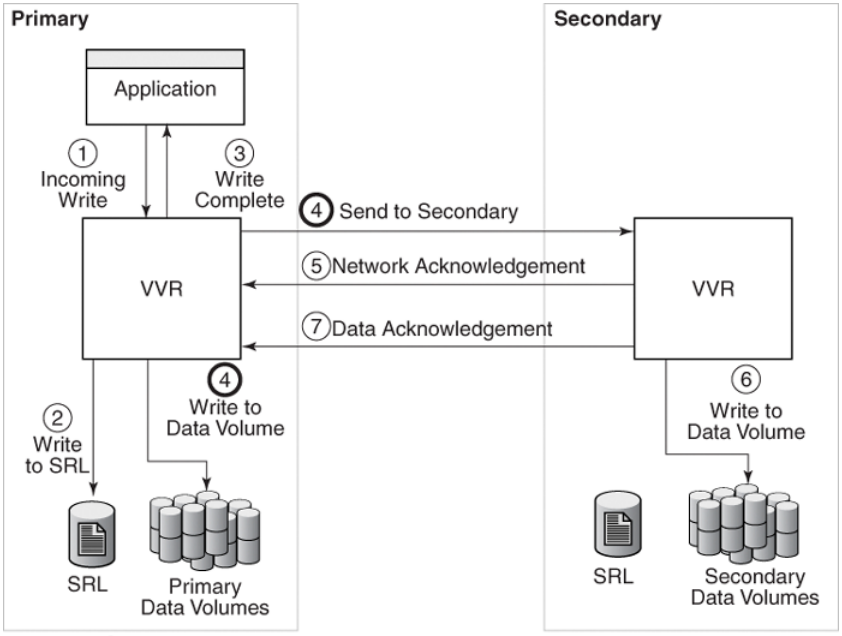
Secondary hosts that fall behind in this manner are serviced by reading back the writes from the Primary SRL. In this case, the writes are sent from the Read Back Buffer, rather than from the Primary buffer as described earlier. The read back process continues until the Secondary catches up with the Primary; at this point, the process of sending writes to the Secondary reverts back to sending from the kernel buffer, instead of sending by reading back from the SRL.

How data flows in VVR asynchronous mode

This section explains how VVR processes an incoming write when replicating in asynchronous mode.

Figure 2-3 shows how data flows in the asynchronous mode of replication.

Figure 2-3 Example—how data flows in the asynchronous mode of replication



In the asynchronous mode of replication, VVR processes an incoming write by performing the following steps in the order listed below:

- VVR receives a write on the Primary.
- Writes it to the Primary SRL.
- On the Primary, acknowledges to the application that the write is complete.
- Sends the writes to the asynchronous Secondary hosts, in the order in which they were received on the Primary, and at the same time, writes to the Primary data volumes.
- When the Primary receives the network acknowledgment, it knows that the write has been received in the Secondary VVR memory buffer.
- VVR sends the writes to the data volumes on the Secondary and then sends a data acknowledgment to the Primary.
- When the Primary receives the data acknowledgment, VVR marks the write as complete in the SRL.

About secondary logging enabled asynchronous mode

Secondary logging is an advanced feature that improves replication performance throughput. This feature uses the Secondary SRL to stage the data before writing to the data volume. The data volume write requires write-order fidelity, which slows the replication throughput. Staging the data to the Secondary SRL allows an immediate acknowledgment of the data write, and allows the Primary to push more data. The Secondary applies the data to the volume in a parallel manner to maintain write-order fidelity. The data acknowledgment is performed immediately after writing on the SRL.

The requirements for automatically enabling support for Secondary logging are as follows:

- The Primary SRL and the Secondary SRL volumes must be of the same size. The SRL can be resized using the `vradmin resizesrl` command.
- The RVG version on both the Primary and the Secondary must be 40 or above. The `vxprint -Vl` command displays the RVG version. The `vxrvg -g` *`diskgroup`* `upgrade rvg` command can be used to upgrade the RVG version. The RVG version can be upgraded only when the RLINK is up to date.
- The disk group version on both the Primary and the Secondary must be 170 or above. The `vxdg list diskgroup` command displays the disk group version. The `vxdg upgrade diskgroup` command can be used to upgrade the disk group version. Upgrading the disk group using this command automatically upgrades the RVG versions of all the RVG's in the diskgroup. Note that before upgrading

the disk group, it is recommended that you pause the replication prior to the upgrade, and resume replication after the upgrade. The RLINK needs to be up to date, otherwise only the disk group will be upgraded, and the RVG upgrade will fail.

If any of these requirements are not met, the Secondary logging feature is automatically disabled, and traditional replication is used, which may provide lesser replication throughput.

About bulk transfer with secondary logging

To effectively use network bandwidth for replication, data is replicated to a disaster recovery (DR) site in bulk at 256 KB. This bulk data transfer reduces VVR CPU overhead and increases the overall replication throughput. With compression enabled, bulk data transfer improves the compression ratio and reduces the primary side CPU usage.

Bulk transfer requires secondary logging to be enabled, and is supported with both TCP and UDP protocols. Bulk transfer is not supported with bunker replication, and in cross-platform replication.

Bulk transfer for replication is automatically enabled from disk group version 190. If bulk transfer is enabled, the `vxprint -Vl` command displays `bulktransfer` under the RVG `flags` field.

For example:

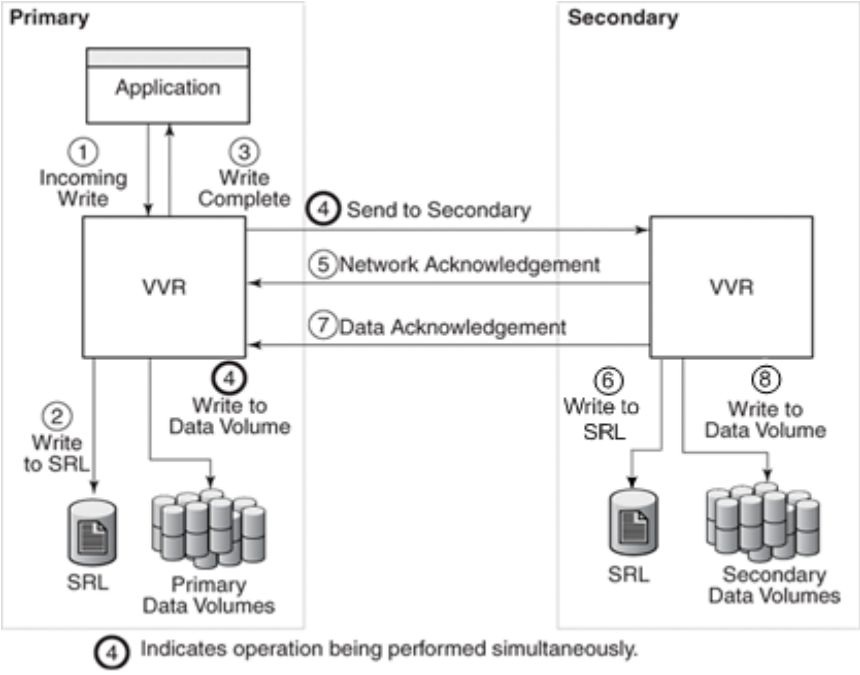
```
# vxprint -Vl -g vvrdg rvg
Rvg:      rvg
info:     rid=0.1061 version=2 rvg_version=45 last_tag=1
state:    state=ACTIVE kernel=ENABLED
assoc:    datavols=vol
          srl=arl
          rlinks=rlk_c1059rhel6_rvg
          exports=(none)
          vsets=(none)
att:      rlinks=rlk_c1059rhel6_rvg
flags:    closed primary enabled attached bulktransfer
device:   minor=23002 bdev=199/23002 cdev=199/23002 \
          path=/dev/vx/dsk/vvrdg/rvg
perms:    user=root group=root mode=0600
```

How data flows in VVR asynchronous mode with secondary logging

This section explains how VVR processes an incoming write when replicating in asynchronous mode with secondary logging.

Figure 2-4 shows how data flows in the asynchronous mode of replication.

Figure 2-4 Example—how data flows in the asynchronous mode of replication with secondary logging



In the asynchronous mode of replication with secondary logging, VVR processes an incoming write by performing the following steps in the order listed below:

- VVR receives a write on the Primary.
- Writes it to the Primary SRL.
- On the Primary, acknowledges to the application that the write is complete.

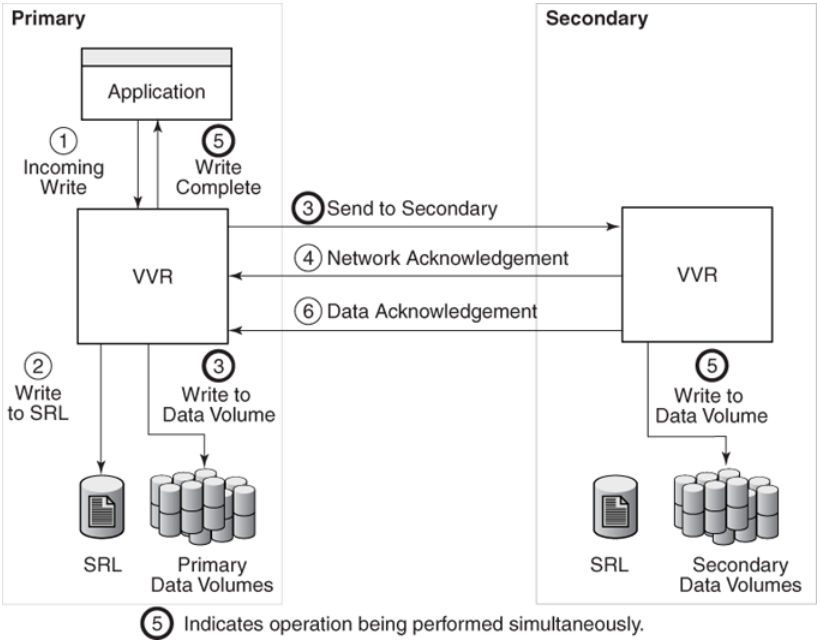
- Sends the writes to the asynchronous Secondary hosts, in the order in which they were received on the Primary, and at the same time, writes to the Primary data volumes.
- When the Primary receives the network acknowledgment, it knows that the write has been received in the Secondary VVR memory buffer.
- VVR sends the write to the Secondary SRL and then sends a data acknowledgment to the Primary.
- In parallel, the Secondary issues the write to the data volumes on the Secondary data volume. This write is queued and started in parallel with other queued writes to the data volume.
- When the Primary receives the data acknowledgment, VVR marks the write as complete in the SRL.

How data flows in VVR synchronous mode

This section explains how VVR processes an incoming write when replicating in synchronous mode.

Figure 2-5 shows how data flows in the synchronous mode of replication.

Figure 2-5 Example—how data flows in the synchronous mode of replication



In synchronous mode of replication, VVR processes an incoming write by performing the following steps in the order listed below:

- VVR receives a write on the Primary.
- Writes it to the Primary SRL.
- Sends the write to the Secondary hosts and waits for the synchronous network acknowledgments from the Secondary hosts. At the same time, VVR writes to the data volumes on the Primary.
- On the Secondary, VVR receives the write, processes it, and sends a network acknowledgment to the Primary.
- Sends writes to the data volumes on the Secondary; when the Primary receives a network acknowledgment from all the Secondary hosts, VVR acknowledges to the application that the write is complete.
  Note that the Secondary RVG sends the network acknowledgment as soon as the write is received in the VVR kernel memory. This removes the time required to write to the Secondary data volumes from the application latency. On the Primary, VVR does not wait for data to be written to the Secondary data volumes. This improves application performance. However, VVR tracks all such acknowledged writes that have not been written to the data volumes. VVR can replay these tracked writes if the Secondary crashes before writing to the data volumes on the Secondary or if the Primary crashes before it receives the data acknowledgment.
- When the write is written to the data volumes on the Secondary, VVR on the Secondary sends a data acknowledgment to the Primary. VVR marks the write as complete in the SRL when the Primary receives the data acknowledgment from all the Secondary hosts.

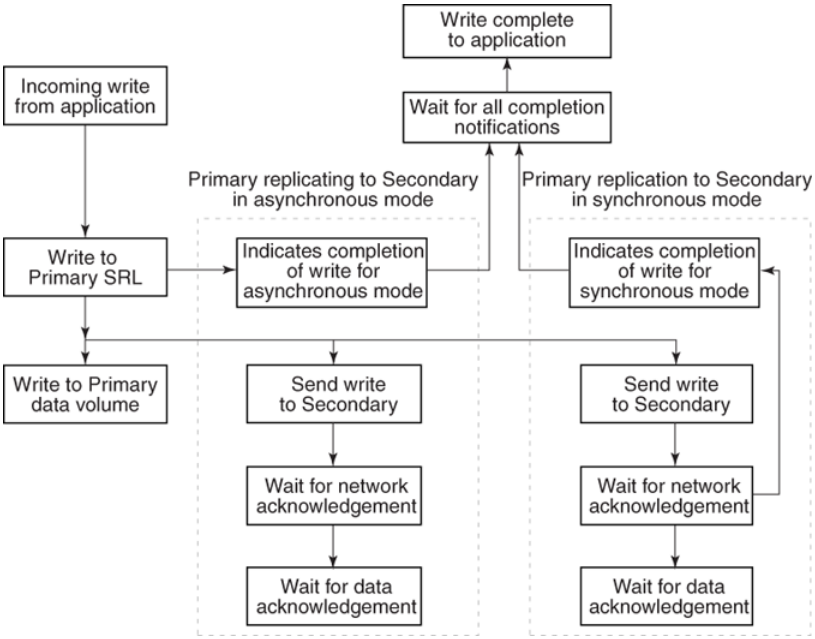
When an RDS containing multiple Secondary RVGs is replicating in synchronous mode, the application latency is determined by the slowest Secondary. Overall performance in synchronous mode is determined by the time to write to the SRL, plus the round-trip time required to send data to the Secondary RVG and receive the acknowledgment.

How data flows in an RDS containing multiple Secondary hosts

This section explains how VVR processes an incoming write for a Replicated Data Set (RDS) containing multiple Secondary hosts, some replicating in asynchronous mode and some in synchronous mode.

Figure 2-6 shows how data flows in an RDS with multiple Secondaries.

Figure 2-6 How data flows in an RDS with multiple Secondaries



In asynchronous and synchronous mode of replication, VVR processes an incoming write with the following steps in the order shown:

- Receives a write from the application.
- Writes it to the SRL.
- Sends the write to the Secondary hosts replicating in synchronous mode and in asynchronous mode. At the same time, VVR writes to the data volume on the Primary.
- On the Secondary, VVR receives the write, processes it, and sends a network acknowledgment to the Primary.
- When the Primary receives a network acknowledgment from the Secondary hosts replicating in synchronous mode, VVR acknowledges to the application that the write is complete.

  Note that the Secondary RVG sends the network acknowledgment as soon as the write is received in the VVR kernel memory. This removes the time required to write to the Secondary data volumes from the application latency. On the Primary, VVR waits only for the network acknowledgment from all the synchronous Secondary hosts and not for the data to be written to the Secondary data volumes. This improves application performance. However, VVR tracks all

such acknowledged writes that have not been written to the data volumes. VVR can replay these tracked writes if the Secondary crashes before writing to the data volumes on the Secondary or if the Primary crashes before receiving the data acknowledgment.

- When the write is written to the data volumes on the Secondary, VVR sends a data acknowledgment from the Secondary to the Primary in both synchronous and asynchronous mode.
- When the Primary receives the data acknowledgment from all the Secondary hosts, VVR marks the write as complete in the SRL.

Replication in a shared disk group environment

VVR enables you to replicate data volumes in a shared disk group environment, for use with parallel applications that use Cluster Volume Manager (CVM) for high availability. You can replicate data volumes in a shared disk-group to a remote site, for disaster recovery or off-host processing.

A shared disk group is shared by all nodes in a cluster. A shared (or cluster-shareable) disk group is imported by all nodes in a cluster. VVR supports configurations in which both the Primary and Secondary disk group are shared, or either the Primary or the Secondary disk group is shared. Therefore, if the Primary disk group is shared, the Secondary disk group can be a private disk group or vice versa.

When replicating data from a shared disk group to the remote site, VVR works with the cluster functionality of Veritas Volume Manager. The cluster functionality of VxVM requires that one node act as the master node; all other nodes in the cluster are slave nodes.

Note: Currently, replication support is limited to 8-node cluster applications.

For complete information on the cluster functionality (CVM) provided by VxVM, see the *Storage Foundation Cluster File System High Availability Administrator's Guide*

VVR includes the VCS agents for VVR to provide support for VVR in a shared disk group environment.

See “VCS agents for VVR” on page 28.

For information about VCS, see the Cluster Server documentation set.

Note: Cluster Server is a separately licensed product. Cluster Server is not included with Volume Replicator. Veritas Cluster Volume Manager (cluster functionality) is included with Veritas Volume Manager, however you must have a separate license to use this feature. VVR also supports the cluster functionality of Veritas File System (VxFS), which is a separately licensed product.

VVR adheres to the same model as CVM for most commands that change the configuration. However, some commands that are specific to VVR are exceptions and must be issued from the CVM master node. These commands include `vradmin createpri`, `vxibc`, `vxrvg`, and the `vxrlink` commands including `vxrlink assoc`, `vxrlink dis`, `vxrlink att`, `vxrlink det`, `vxrlink pause`, `vxrlink resume`, `vxrlink restore`, and `vxrlink checkdelete`. Informational commands such as `vxrlink stats`, `vxrlink status`, `vxrlink verify`, `vxrlink cplist`, and `vxrlink updates` can be run on any cluster node.

Note that the `vxrlink status` command and the `vxrlink stats` command display the same information on all the nodes in the cluster, whereas, the `vxrvg stats` command displays information pertaining to the node on which you run the command. The `vxrvg stats` command provides information about the reads and writes performed on the node on which you run it, therefore the information applies to that node only.

The role of the logowner

To use the Secondary in a disaster recovery scenario, the order of writes (write-order fidelity) must be maintained. When replicating in shared disk-group environment, VVR maintains the order of writes by designating one node in the cluster as the logowner. The logowner manages the writes to the SRL on the Primary. The writes are handled differently depending on whether the replication is set to synchronous or asynchronous.

For synchronous RLINKs, all writes are performed on the logowner; writes issued on nodes other than the logowner are sent over the cluster network to the logowner, to be performed there. This process is called write shipping.

For asynchronous RLINKs, the writes are performed on the node where they are issued. However, before writing to the SRL, the node sends a request to the logowner. The logowner responds with a message indicating the position in the SRL that was assigned for that write. After receiving the response from the logowner, the node writes to the SRL and then to the data volumes. This process is called metadata shipping. The information about the position in the SRL and how much space is allocated is known as metadata. If an RVG has both synchronous and asynchronous RLINKs, the RVG uses write shipping.

The logowner also is responsible for replicating the writes for the entire cluster to the Secondary site. If the RLINK is using metadata shipping, the logowner must read back the writes from the SRL before sending the writes to the Secondary.

The master is the default logowner. The logowner follows the CVM master on a master switch.

Note: When you upgrade VVR from a previous version, the logowner role is not preserved. After the cluster upgrade, the CVM master node becomes the default logowner.

How VVR processes a write in a shared disk group

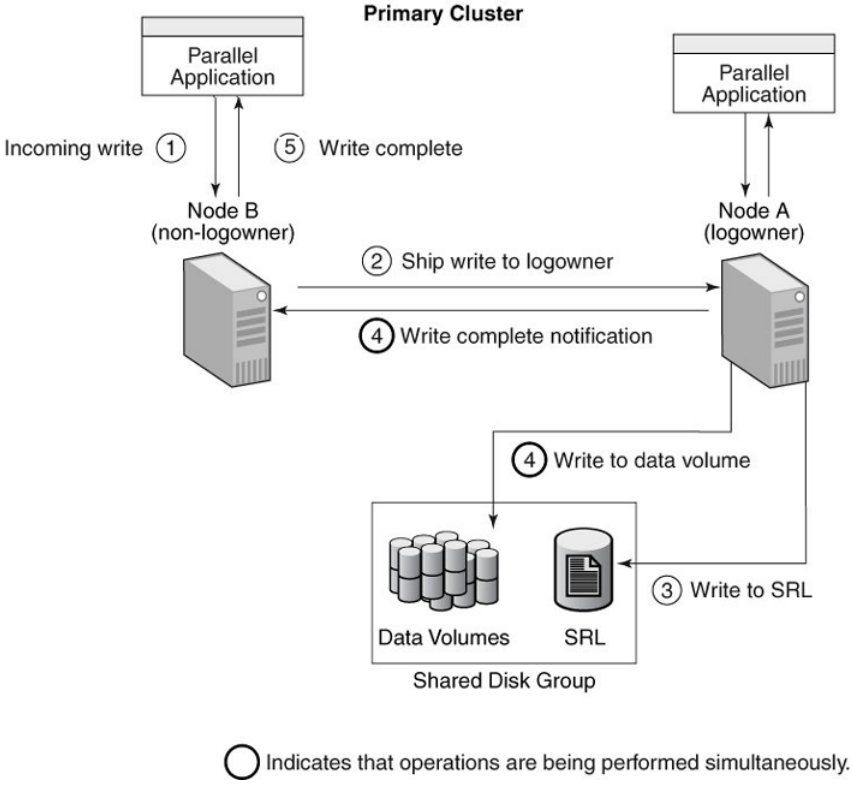
This section explains how VVR processes an incoming write for a Primary cluster containing two nodes. In a shared disk group environment, VVR processes an incoming write on the logowner in the same way as in a private disk group environment.

For a non-logowner, VVR processes an incoming write in one of the following ways:

- write shipping
- metadata shipping

Figure 2-7 shows how VVR processes an incoming write on the non-logowner for an RVG that is using write shipping.

Figure 2-7 Example—how VVR processes a write on the non-logowner using write shipping

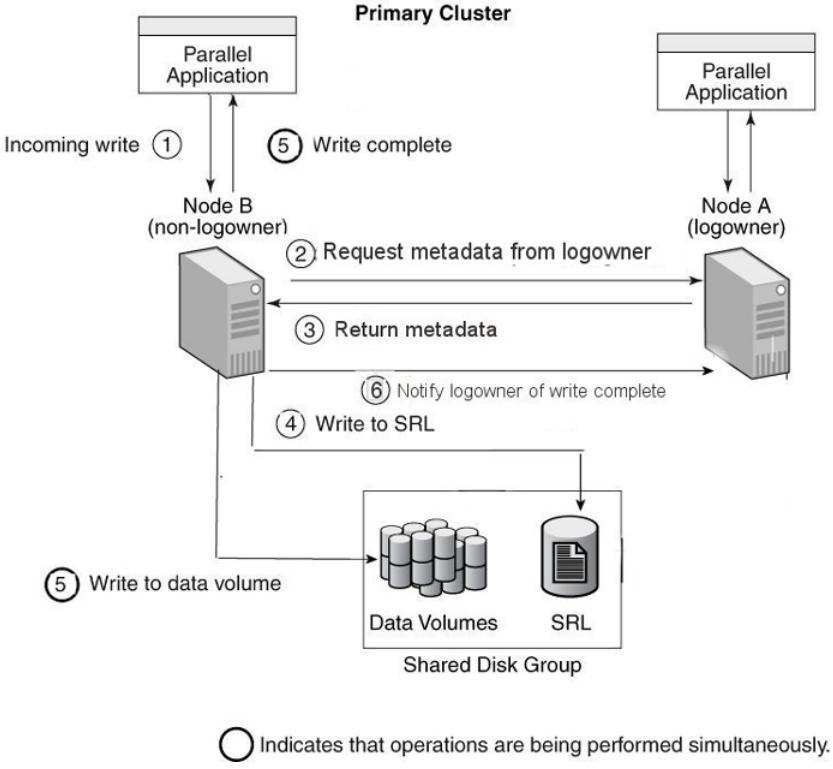


As shown in the illustration Figure 2-7, VVR processes an incoming write on the non-logowner (Node B) with the following steps in the order shown:

- VVR receives a write from the application on the non-logowner, Node B.
- Node B ships the write to the logowner, Node A.
- Node A writes to the Primary SRL.
- Node A notifies Node B that the write is complete. Simultaneously, Node A writes to the data volumes.
- Node B completes the write to the application.

Figure 2-8 shows how VVR processes an incoming write on the non-logowner for an RVG that is using metadata shipping.

Figure 2-8 Example—How VVR Processes a Write on the Non-logowner Using Metadata Shipping



As shown in the illustration Figure 2-8, VVR processes an incoming write on the non-logowner (Node B) as follows:

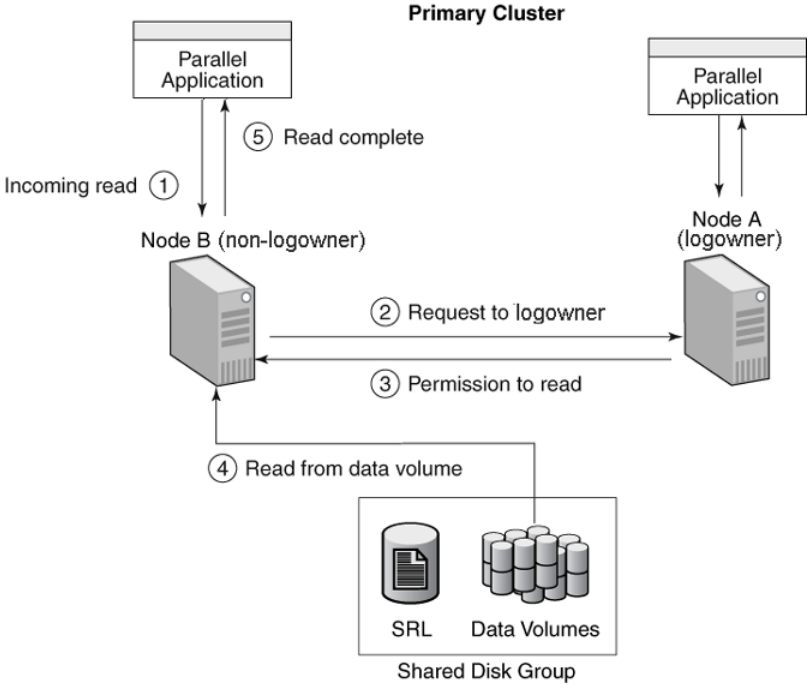
- VVR receives a write from the application on the non-logowner, Node B.
- Node B requests metadata to write to the SRL from the logowner, Node A.
- Node A sends metadata to write to Node B.
- After receiving metadata from Node A, Node B writes to the Primary SRL.
- Node B writes to the data volumes. Simultaneously, Node B completes the write to the application.
- Node B notifies the logowner that the write is complete.

How VVR processes a read in a shared disk group

This section explains how VVR processes an incoming read for a Primary cluster containing two nodes. In a shared-disk group environment, VVR processes an incoming read on the master in the same way as in a private disk group environment.

Figure 2-9 shows how VVR processes an incoming read on the non-logowner.

Figure 2-9 Example—how VVR processes a read on the non-logowner



As shown in the illustration, Figure 2-9, VVR processes an incoming read on the non-logowner, (Node B) with the following steps in the order shown:

- VVR receives a read from the application on the non-logowner, Node B.
- Node B sends a request for read permission to the logowner, Node A.

  Note: All requests for read and write permissions are sent to the logowner. If the logowner is not the master, it sends permission requests to the master.

- Node B receives permission to read from Node A.
- Node B reads from the data volumes.

- Node B completes read to the application.

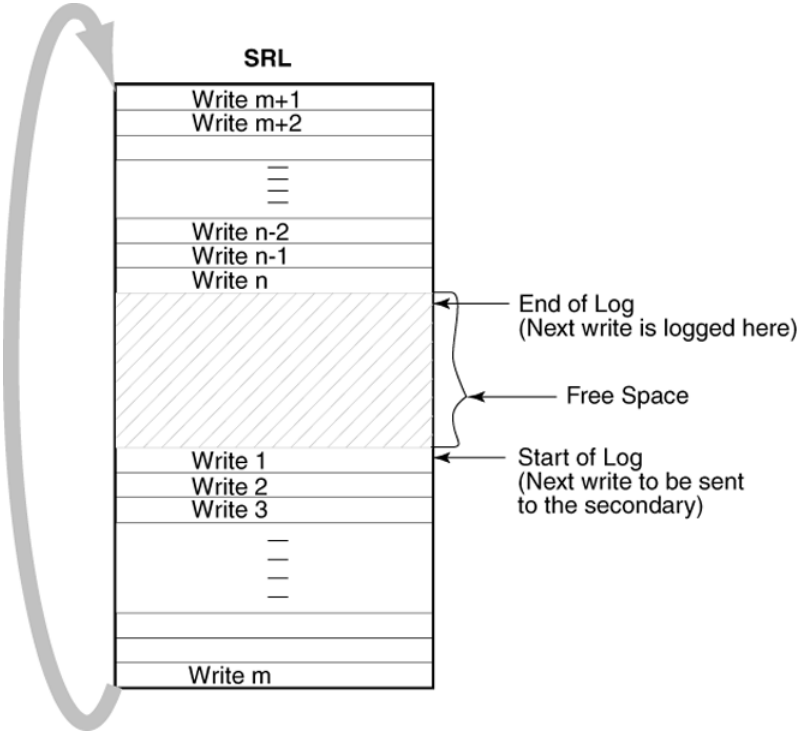
Understanding how VVR logs writes to the SRL

VVR receives writes from the application and queues them in the SRL for transmission to the Secondary hosts. All the RLINKs of an RVG share the SRL. The SRL header contains a specific set of pointers for each RLINK that indicates the writes that have not been sent to the corresponding Secondary.

This section explains the working of the SRL as a circular buffer.

Figure 2-10 shows how writes are logged in the SRL.

Figure 2-10 Example—How VVR Logs Writes to the SRL



As shown in Figure 2-11, the earliest write that came in is Write 1, which also represents the Start of Log for the Secondary.

VVR logs Write 2, Write 3, Write m one after the other until it reaches the end of the SRL. Because the SRL is a circular log the next write, Write m+1 wraps around and logging continues. When the Primary receives the data acknowledgment from

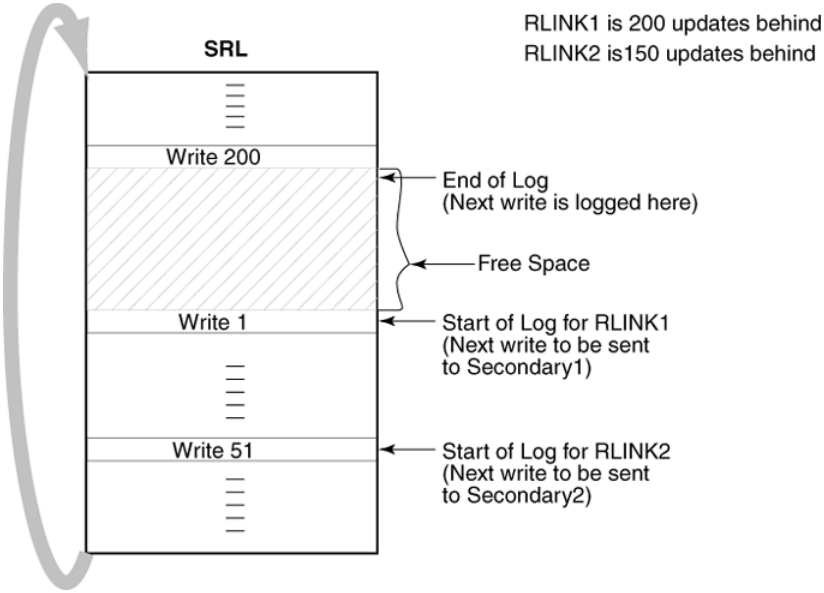
this Secondary host for Write 1, VVR marks the Write 1 as complete in the SRL. VVR then processes Write 2, Write 3, and so on.

VVR maintains the following types of pointers in the SRL header:

The Start of Log pointer	Each RLINK has a Start of Log pointer that designates the next write, Write 1, to be sent to the Secondary.
The End of Log pointer	Designates the location to be written to by the next incoming write after Write n.

Figure 2-11 shows how VVR logs writes to the SRL in an example configuration.

Figure 2-11 Example—How VVR Logs Writes to the SRL in a Multiple RLINK Set Up When Each RLINK is Behind by a Number of Updates



In this example, RLINK1 is 200 writes or updates behind, whereas RLINK2 is 150 writes behind. If the End of Log pointer reaches the Start of Log pointer of the RLINK, the SRL overflows for this RLINK

Synchronous RLINKs are usually up-to-date. Typically, the Start of Log and End of Log pointers of synchronous RLINKs are separated by the number of simultaneous I/O operations the application performs. For asynchronous RLINKs, the difference between the Start of Log pointer and End of Log pointers reflect how many outstanding writes have yet to be processed, that is, how behind is the RLINK.

Different RLINKs usually have Start of Log pointers indicating different places in the SRL; this reflects the difference in the rate at which data is sent to the Secondary.

Understanding Storage Checkpoints

VVR Storage Checkpoints are user-defined markers in the SRL. Each Storage Checkpoint has a start (checkstart) and an end (checkend). Storage Checkpoints are used to perform the following tasks:

- Synchronizing the Secondary while the Primary application is active
- Restoring the Secondary data volumes

The Secondary data volumes must be synchronized with the Primary data volumes before replication can start: that is, after adding a Secondary to the RDS, after a Secondary data volume error, or after SRL overflow. VVR enables you to synchronize the Secondary data volumes while the application is active on the Primary. If you use the automatic synchronization feature of VVR to synchronize the Secondary data volumes over the network, VVR ensures that the Secondary data volumes are consistent and up-to-date when the synchronization process completes. However, you can also synchronize the Secondary data volumes by performing a backup of the Primary data volumes and applying it on Secondary or by copying the data over the network using the VVR `vradmin` command or any other utility. If the Primary application is active during the synchronization process, the Secondary data volumes are inconsistent and not up-to-date when the synchronization is complete.

Typically, a backup or synchronization utility performs sequential reads starting with the first block of the volume until it reaches the last block of the volume and transfers those blocks either to tape or over the network. If the Primary application is active during this process, some Primary data volume blocks might have changed while the data volumes are read sequentially. It is likely that the application changes several blocks, some of which are read by the synchronization process before they were changed and some after they were changed. This results in the Secondary data volumes being inconsistent and not completely up-to-date at the end of the synchronization process.

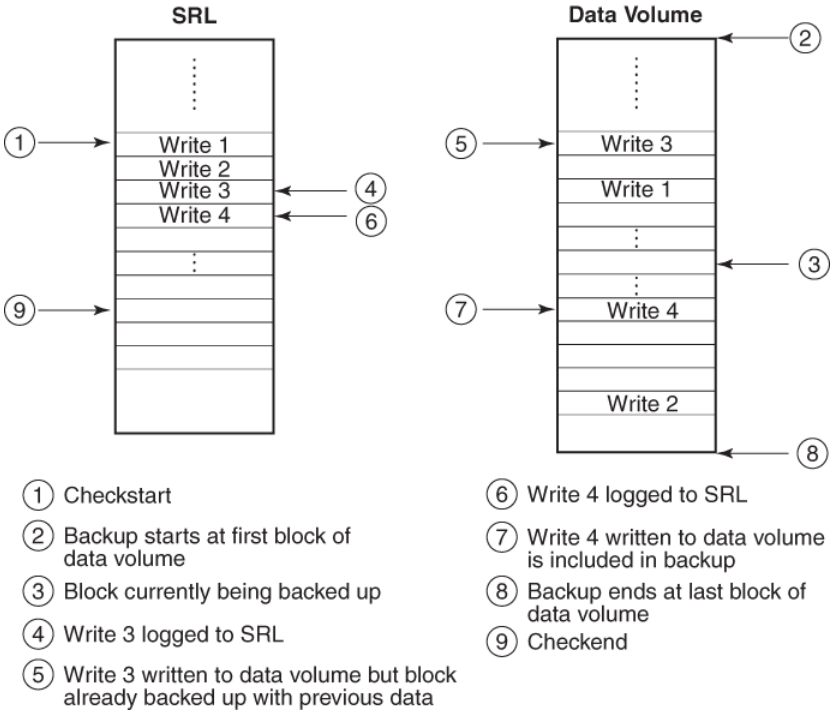
To make the Secondary consistent and up-to-date, VVR must transfer in order all the blocks that changed during the synchronization process. In a VVR environment, all writes to the Primary data volumes are logged to the SRL; therefore, VVR can transfer the writes that occurred during the synchronization to the Secondary. To do this, VVR must know the start and end of the synchronization process. VVR Storage Checkpoints are used to indicate this start position (checkstart) and end position (checkend) in the SRL.

Because the Storage Checkpoint information is stored in the SRL, Storage Checkpoints become invalid when the SRL wraps around. The same Storage Checkpoint and tape backups can be used to synchronize the data volumes on multiple Secondary hosts if the Storage Checkpoint remains valid.

VVR enables you to create a maximum of forty-six Storage Checkpoints. If the number of Storage Checkpoints exceeds this number VVR displays an error message asking you to delete the earlier Storage Checkpoints. You can selectively delete the required Storage Checkpoints.

Figure 2-12 shows how VVR uses the Storage Checkpoints.

Figure 2-12 Example—how VVR uses the Storage Checkpoints



As shown in the illustration, a backup utility may copy previous contents of the blocks corresponding to Write 3 (event 5) but copy updated contents of block corresponding to Write 4 (event 7). However, VVR logs all the writes to the SRL (events 4 and 6). Note that a checkstart was performed (event 1) before the backup was started (event 2) and a checkend was performed (event 9) after the backup was completed (event 8). On starting replication with this Storage Checkpoint after the synchronization is complete, VVR can transfer all the writes between checkstart and checkend and make the Secondary data volumes up-to-date and consistent.

Volume sets in VVR

Volume Replicator supports replication for volume sets. Volume sets are an enhancement to VxVM that allow several volumes to be represented by a single logical object. 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-device 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, metadata for a file system could be stored on volumes with higher redundancy, and user data on volumes with better performance.

VVR also supports replication for volume sets with SmartTier or Sub-file SmartTier.

In previous releases of VVR, component volumes of a volume set could be associated to an RVG. The individual component volumes were replicated to the Secondary. Now, VVR supports associating a volume set to an RDS, and replicating the component volumes. When a volume set is associated with an RVG, VVR internally associates all the component volumes to the RVG. A component volume can later be explicitly disassociated from the RVG; however, a volume should only be excluded from the RVG if it is not important for the application.

After the volume set is associated with an RVG, replicating that RVG replicates all the component volumes. If a Primary RVG contains one or more volume sets, the Secondary RVG must have the corresponding volume sets. The volume sets on the Secondary RVG must have at least the same component volumes as the Primary RVG.

The volumes in a volume set that is associated with an RVG are treated like any other volume in the RVG for all operational purposes. That is, any operation on the RVG that operates on the volumes includes the volumes that are part of associated volume sets.

Changing membership of an RVG and a volume set

The volume set represents a logical grouping of volumes from the application perspective. In order for VVR to replicate the volume set successfully, the same volume configurations must exist on the Primary and the Secondary. Commands that break the configuration consistency will fail.

VVR tracks which component volumes make up the replicated volume set, and ensures that the component volumes of the volume set remain consistent between the Primary and the Secondary. If a component volume is added or deleted to the volume set, VVR makes the corresponding change to the RVGs on each host of

the RDS. The component volumes with the same names and lengths must already exist on each host.

Table 2-1 shows the operations which affect the membership of the volume set.

Table 2-1 Membership Operations

Command	Action	Result
`vradmin -tovset` *`vset`* `addvol` *`rvg vol`*	Adding a volume to a volume set that is associated to an RVG.	Adds the volume to the volume set and to the RVG.
`vradmin addvol` *`rvg vset`*	Associating a volume set to an RVG	All the component volumes of the volume set are internally associated to the RVG.
`vradmin addvol` *`rvg vol`*	Adding a component volume of a volume set to an RDS.	If the volume set is already associated to the RDS, but some component volumes are excluded, use this command to add the component volume to the RDS. This operation fails if the volume set is not associated to the RDS.
`vradmin delvol` *`rvg vset`*	Removing a volume set from an RVG	All the component volumes of the volume set are removed from the RVG. The membership of component volumes in the volume set is not affected.
`vradmin -fromvset` *`vset`* `delvol` *`rvg vol`*	Deleting a volume from a volume set that is associated to an RVG.	Deletes the volume from the volume set and from the RVG.
`vradmin delvol` *`rvg vol`*	Deleting a component volume of a volume set from the RDS	Deletes the volume from the RDS but the volume will still remain associated with the volume set. **Note:** Deleting a volume this way means that the volume set is only partially being replicated.

Using SmartTier with VVR

The volume set feature supports the multi-device enhancement to Veritas File System (VxFS). This feature (SmartTier) allows file systems to make best use of the different performance and availability characteristics of the underlying volumes. For example, metadata for a file system could be stored on volumes with higher redundancy, and user data on volumes with better performance.

Best practices for using SmartTier with VVR

When using VVR to replicate a multi-device file system, we recommend the following best practices:

- If you partially associate a volume set to an RVG, ensure that all component volumes with metadata information (typically of type `metadataok`) are part of the RVG. If you exclude the component volume that includes metadata, then the file system cannot be brought up on the Secondary because the metadata is not replicated.
- By default, every volume in an SmartTier is `metadataok`, which means that the volume can also include metadata. If you want to exclude certain volumes for replication, then we recommend marking those volumes as `dataonly` volumes.

Cross-platform Data Sharing in VVR

Cross-platform Data Sharing (CDS) allows the sharing of data between heterogeneous systems where each system has direct access to the physical devices used to hold the data. Sharing in this manner requires the capability to share the devices at various levels in the software hierarchy.

Volume Replicator (VVR) is CDS compliant. VVR uses CDS format to support the following functionality:

- Ability to migrate data between heterogeneous systems (either on the Primary or the Secondary) where each system has direct access to the physical devices used to hold the data. Note that CDS does not support simultaneous use of data from more than one platform.

  Warning: Stop all the applications working on that disk group before migrating data between hosts. Failure to do so may result in data loss.

- Ability to replicate data between heterogeneous systems as a result of CDS. The Primary host could be a different platform from the Secondary host, and each host would be able to access the data in the CDS format.

The Storage Replicator Log (SRL) is created in a CDS format. Starting with the 5.0 release, the SRL is created in the CDS format, regardless of the disk group type. When you upgrade from a previous release of VVR to 5.0 or later, the upgrade process dissociates the SRL and creates a new SRL in CDS format.

When you use VVR to replicate between targets that have different byte orders, for example targets on different operating systems, you need to use application-specific byte conversion utilities to interpret the replicated data. For example, if the Veritas File System (VxFS) is used on top of the replicated volumes, you must use `fscdsconv` to convert the file system to the native byte order.

For more information, see the *Veritas InfoScale™ Solutions Guide*.

Understanding the VVR snapshot feature

VVR enables you to create an image of the online data volumes, at a given point in time and such an image is referred to as a snapshot. The data in the original volume may change; however, the snapshot can still be used as a stable and independent copy for various purposes, including the following tasks:

- to restore data both on the Primary and Secondary if the original data gets corrupted because of logical errors, administrative errors, or media errors such as disk failures.
- to verify the Disaster Recovery (DR) readiness of the DR site or perform fire drill activities. The VCS fire drill feature uses space-optimized snapshots to support testing on the DR site.
- to create a copy of the data for application development or testing.
- to support off-host processing for applications such as Decision Support Systems (DSS) or for report generation.
- to perform online data verification of the volumes in an RVG when replication is in progress.
- to retain a consistent copy of the Secondary data volumes during Data Change Map (DCM) resynchronization.

Note: You can use the snapshot feature on the Primary and the Secondary host.

The snapshot feature in VVR is the same as the snapshot feature in VxVM, because VVR is fully integrated with VxVM. In VVR, an RVG is a group of VxVM volumes; therefore, taking a snapshot of an RVG is the same as taking a snapshot of the data volumes in the RVG. VVR enables you to create traditional and instant snapshots.

See "About the traditional snapshot feature" on page 55.

See "About the instant snapshot feature" on page 56.

About snapshots of RVGs containing volume sets

If an RVG contains a volume set, creating a snapshot of the RVG (using the `vxrvg snapshot` command) takes a snapshot of each of the component volumes of the volume set that are associated to that RVG. The snapshot consists of a container volume set object with snapshots of the associated component volumes. The volumes in the snapshot volume set have the same indexes as the volumes in the original volume set.

When a snapshot of a volume in an RVG is taken, IO is quiesced on all volumes in the RVG. If a volume set is associated to an RVG, taking a snapshot of the RVG will quiesce all of the volumes of the RVG, including the components of the volume set.

If an RVG contains a volume set, use the `vxrvg snapshot` command to take a snapshot of the RVG.

To display snapshots of a volume set, use the `vxrvg snapprint` command.

The `vxrvg snapshot` command provides the exclude keyword, to exclude volumes from the snapshot creation. Additional keywords (`instantso`, `instantfull`, and `instantplex`) are used to create snapshots of the indicated type for the specified volumes. For any of these keywords, you can specify the name of a volume set or the name of an independent volume; however, do not specify a name of a component volume of the volume set. The container snapshot for the volume set therefore will include snapshots of the same type.

See "Creating RVG snapshots" on page 245.

About the traditional snapshot feature

The traditional snapshot feature of VVR enables you to create snapshots of all the data volumes in an RVG at a single point in time, by breaking off the volume plexes. You can create snapshots when the volume plexes are completely synchronized with the data volume.

This method requires you to create and attach the appropriate snapshot plexes that are the same size as the original volumes, before taking the snapshot.

See the *Storage Foundation Administrator's Guide*.

After creating and attaching the snapshot plexes, they must be synchronized with the data volume. The time required for synchronizing the plexes in the beginning

is directly proportional to the size of the volume. Thus, depending on the size of the volume it may take a long time before the plexes are ready to be used.

When the plexes are synchronized, you can then create snapshots after freezing replication using the IBC commands or after pausing replication.

See "Using the traditional snapshot feature" on page 265.

See "About the IBC messaging utility vxibc" on page 477.

About the instant snapshot feature

The instant snapshot feature enables you to take instant full snapshots, instant space-optimized snapshots, or instant plex-breakoff snapshots.

Compared to the tradition method, the instant snapshot feature has the following advantages:

- the plexes or snapshot volumes do not require synchronization before taking the snapshots.
- the snapshots are instantly available.

The instant snapshot feature provided by VVR can be used either from the Primary or the Secondary. VVR also provides you with the option to take space-optimized snapshots.

You can create the types of instant snapshots as described in the following sections:

- See "About instant full snapshots" on page 56.
- See "About Instant space-optimized snapshots" on page 56.
- See "About Instant plex-breakoff snapshots" on page 57.

About instant full snapshots

The instant full snapshot feature of VVR enables you to create a full snapshot of all the data volumes in an RVG without any delay. In this case, the snapshot plexes do not require synchronization before creating the snapshot. Therefore, the required data is available instantly after the snapshot is created. However, this method requires the snapshot volumes to be created with the appropriate naming convention, before you proceed with the snapshots.

See "About instant full snapshots" on page 247.

About Instant space-optimized snapshots

VVR also enables you to take instant space-optimized snapshots. Unlike instant full snapshots, the instant space-optimized snapshots require less storage space

than the original volume because space-optimized snapshots store only the data that has changed between the original volume and the snapshot. Typically, the data that has changed between the original volume and the snapshot volume over the lifetime of the snapshot is minimal compared to the total data on the volume. Thus, the space-optimization achieved can be significant.

The snapshot data is managed by VVR using a cache object which functions as a space-optimized persistent store. You must create the cache object before you take the instant space-optimized snapshots or specify the size of the cache object. Multiple snapshots can be created on the same cache object. The cache object can be created with an *autogrow* option set to `on` to allow it to grow automatically if the cache volume size is not enough for the specified writes. When preparing the RVG volumes for the snapshot operation, create the cache object.

See "Creating instant space-optimized snapshots" on page 254.

See "Preparing the RVG volumes for snapshot operation" on page 253.

About Instant plex-breakoff snapshots

Similar to the traditional plex-breakoff snapshot feature, this method also requires the plexes to be attached to the source volume before taking the snapshots. Although the synchronization of the plexes may still take a long time, the major difference between the traditional snapshots and the instant plex-breakoff snapshots is that you can instantly start performing the operations such as refresh, restore, and snapback on the instant plex-breakoff snapshots.

The instant plex-breakoff snapshots operation requires the plexes to be named using the `plexprefix` attribute if you want to use specific plexes. Otherwise, VVR uses the plexes that are in the snapdone state.

See "About instant plex-breakoff snapshots" on page 257.

How VVR creates instant space-optimized snapshots

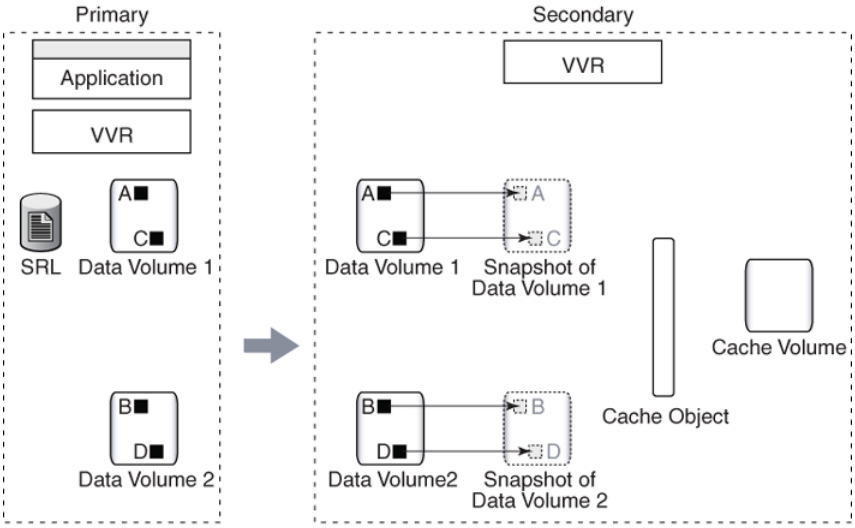
This section explains how VVR creates and manages the space-optimized snapshots.

In the following illustration, the Primary has two data volumes: Data Volume 1 and Data Volume 2. For this example we have indicated two specific blocks, namely, A and C in Data Volume 1 and B and D in Data Volume 2.

The Secondary has two data volumes, Data Volume 1 and Data Volume 2 that have all the replicated data from the Primary, including the specified blocks. The Secondary illustrates an instant space-optimized snapshot in which the data resides on the original volume itself. A Read operation to the snapshots will be redirected to the source volumes and writes will result in a copy-on-write operation. The data

will be copied to the snapshots only if there is a write to the original data. Because the snapshots are space-optimized the data will actually get written to the cache object only if there is a write to the original data.

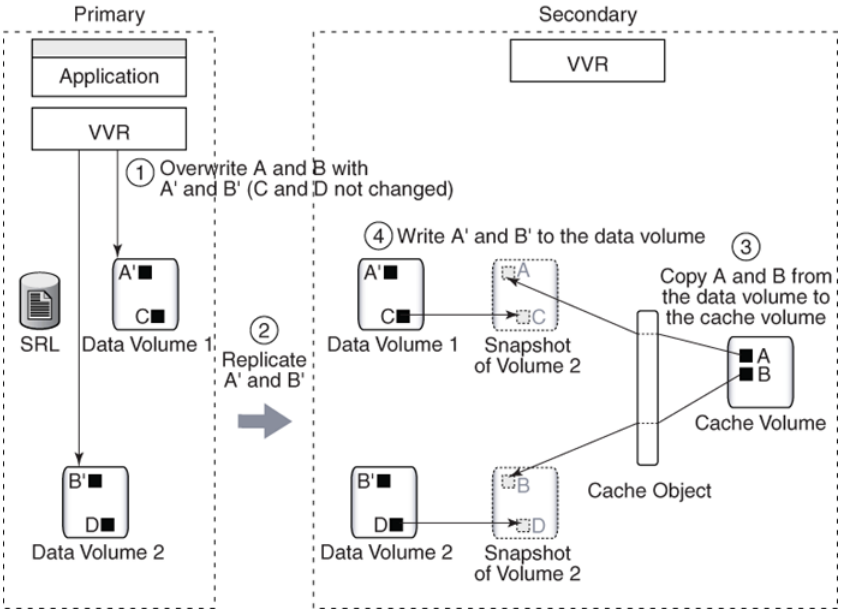
Figure 2-13 Example 1—How VVR creates instant space-optimized snapshots



The following illustration indicates the scenario where the Primary receives some updates to the blocks A and B. These are now represented as A’ and B’.

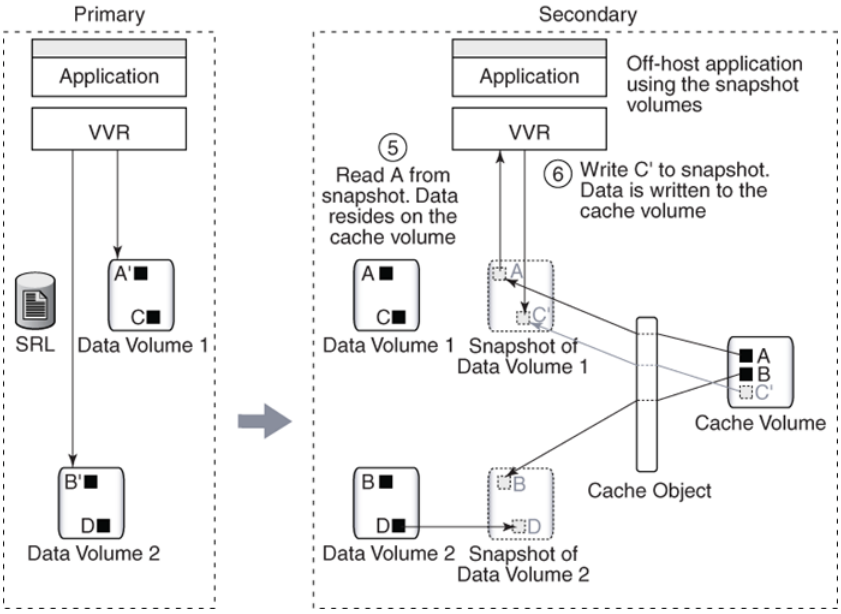
The Secondary also receives the updates A’ and B’. The write to the data volumes first results in a copy-on-write on to the space-optimized snapshot. A space-optimized snapshot is created on a cache object that holds all the data for the snapshots. Hence during a copy-on-write, the blocks A and B get written onto the cache object, before the changed blocks are written to the original volumes, Data Volume 1 and Data Volume 2. The cache object stores the blocks persistently in the cache volume after remapping the original offsets.

Figure 2-14 Example 2—How VVR writes to the instant space-optimized snapshots



The following illustration indicates the scenario when there is a write from the application to the block C on the snapshot. This block in now indicated as C'. The changed information for the block C is preserved on the cache object using the copy-on-write mechanism described earlier. If there is a read then the changed block C' will be read from the cache object. Similarly, if there is a request for reading block A from the snapshot volume, it will be fetched from the cache object, where it has been copied earlier.

Figure 2-15 Example 3—How VVR reads the changed block from the cache object



Comparing VVR snapshot methods based on different features

Table 2-2 provides a comparison of the different snapshot methods by feature.

Table 2-2 Comparison of snapshot methods

Snapshot Features	Traditional Snapshot	Instant Full Snapshot	Instant Space-Optimized Snapshot	Instant Plex Breakoff Snapshot
Requires full storage	Yes	Yes	No	Yes
Requires initialize synchronizing of plexes	Yes	No	No	Yes
Allows snapshots plexes to be reattached (snapback) to the source volume	Yes	Yes	No	Yes
Snapshots can be refreshed	No	Yes	Yes	Yes

Table 2-2 Comparison of snapshot methods *(continued)*

Snapshot Features	Traditional Snapshot	Instant Full Snapshot	Instant Space-Optimized Snapshot	Instant Plex Breakoff Snapshot
Snapshot volumes can be moved into a separate disk group	Yes	Yes	No	Yes
Can be used to restore an RVG to an earlier stable state	No	Yes	Yes	Yes
Can be used as independent volumes	Yes	Yes	No	Yes
Background synchronization	No	Yes	No	No

About VVR compression

The compression feature enables VVR to send data over the network in a compressed format from a Primary to one or more Secondary hosts. It reduces network bandwidth consumption by VVR. This feature is particularly useful in scenarios where there is low available bandwidth or where the bandwidth is shared among several applications. Purchasing an external compression software or hardware for data transfer can prove costly. Hence, compression feature in VVR is a cost-effective alternative in such scenarios.

VVR uses the zlib library for performing compression and decompression. For more information on the zlib compression library, see http://www.zlib.net

---

Note: The compression option can be enabled on a per system or per Secondary basis using the CLI. If compression is enabled, the Primary will generate the compressed data during the sending of any update. At any time, the user can disable or enable the compression feature without stopping replication. Compression is disabled by default on each system and in the VVR engine, for new VVR configurations as well as older VVR configurations upgraded from previous releases. The feature is enabled only when all hosts are running 6.0 or higher and the disk group version is 170 or above.

General functionality considerations for VVR compression

Compression results in unavoidable CPU overhead, both at the Primary and Secondary sites. In a CVM configuration, the CPU overhead will only be seen on the logowner nodes of the respective CVM clusters.

VVR will not send the data in compressed format in following cases:

- If either the Primary or Secondary hosts are not running 6.0 or higher, and the disk group version is not 170 or above
- If the primary RLINK does not have compression enabled
- If the compressed data size is greater than the uncompressed data size
- If no memory is available for allocation of memory buffers from the NMCOM memory pool on the Primary site

Chapter 3

Planning and configuring replication

This chapter includes the following topics:

- Introduction to planning and configuring replication
- Before you begin configuring
- Choosing the mode of volume replication
- Choosing latency and SRL protection
- Planning the network
- Sizing the SRL
- Best practices for setting up replication
- How the agents for hybrid applications work

Introduction to planning and configuring replication

To set up an efficient Volume Replicator (VVR) configuration, it is necessary to understand how the various VVR components interact with each other. This chapter explains the interactions and presents the decisions you must make when setting up a VVR configuration.

This document assumes that you understand the concepts of VVR.

for a discription of VVR concepts.

In an ideal configuration, data is replicated at the speed at which it is generated by the application. As a result, all Secondary hosts remain up to date. A write to a data

volume in the Primary flows through various components and across the network until it reaches the Secondary data volume. For the data on the Secondary to be up to date, each component in the configuration must be able to keep up with the incoming writes. The goal when configuring replication is that VVR be able to handle temporary bottlenecks, such as occasional surges of writes, or occasional network problems.

If one of the components cannot keep up with the write rate over the long term, the application could slow down because of increased write latency, the Secondary could fall behind, or the SRL might overflow. If a component on the path that completes the write on the Primary cannot keep up, latency might be added to each write, which leads to poor application performance. If other components, which are not on this path, cannot keep up with the write rate, it is likely that the writes on the Primary proceed at their normal pace but accumulate in the SRL. As a result, the Secondary falls behind and the SRL eventually overflows. Therefore, it is important to examine each component to ensure that it can support the expected application write rate.

In this document, the term, application, refers to the program that writes directly to the data volume. If a database is using a file system mounted on a data volume, the file system is the application; if the database writes directly to a data volume, then it is considered the application.

Before you begin configuring

Before you begin configuring VVR, you must understand the characteristics of the application writes that are to be replicated. You must also understand the needs of the business for which VVR is being deployed.

Understanding business needs

To satisfy the needs of your business, you must consider the following:

- The amount of data that can be lost if a disaster occurs and yet continue the business successfully
- The amount of time acceptable to recover the data after the disaster and continue the business successfully

In a traditional tape backup scheme, the amount of data lost in a disaster can be large, depending on the frequency of backup and tape vaulting. Also, the recovery time from a tape backup can be significant. In a VVR environment, recovery time is negligible and the amount of data lost depends on the following factors:

- Mode of replication
- Network bandwidth

- Network latency between the Primary and the Secondary
- Ability of the Secondary data volumes to keep up with the write rate

If the data on the Secondary must be as up to date as possible, we recommend that you use synchronous mode and provide the same bandwidth as the peak rate at which the application writes on the Primary. However, if the Secondary can be allowed to lag behind, we recommend that you use asynchronous mode and provide the same bandwidth as the average rate at which the application writes on the Primary. These decisions are determined by your business needs.

Understanding application characteristics

Before you configure an RDS, you must know the data throughput that must be supported, that is, the rate at which the application can be expected to write data. Only write operations are of concern; read operations do not affect replication. To perform the analyses described in later sections, a profile of application write rate is required. For an application with relatively constant write rate, the profile could take the form of certain values, such as:

- Average application write rate
- Peak application write rate
- Period of peak application write rate

For a more volatile application, a table of measured usages over specified intervals may be needed. Because matching application write rate to disk capacity is not an issue unique to replication, it is not discussed here. It is assumed that an application is already running, and that Veritas Volume Manager (VxVM) has been used to configure data volumes to support the write rate needs of the application. In this case, the application write rate characteristics may already have been measured.

If the application characteristics are not known, they can be measured by running the application and using a tool to measure data written to all the volumes to be replicated. If the application is writing to a file system rather than a raw data volume, be careful to include in the measurement all the metadata written by the file system as well. This can add a substantial amount to the total amount of replicated data. For example, if a database is using a file system mounted on a replicated volume, a tool such as `vxstat` (see vxstat(1M)) correctly measures the total data written to the volume, while a tool that monitors the database and measures its requests fails to include those made by the underlying file system.

It is also important to consider both peak and average write rates of the application. These numbers can be used to determine the type of network connection needed. For Secondary hosts replicating in synchronous mode, the network must support the peak application write rate. For Secondary hosts replicating in asynchronous

mode that are not required to keep pace with the Primary, the network only needs to support the average application write rate.

Finally, once the measurements are made, the numbers calculated as the peak and average write rates should be close to the largest obtained over the measurement period, not the averages or medians. For example, assume that measurements are made over a 30-day period, yielding 30 daily peaks and 30 daily averages, and then the average of each of these is chosen as the application peak and average respectively. If the network is sized based on these values, then for half the time there will be insufficient network capacity to keep up with the application. Instead, the numbers chosen should be close to the highest obtained over the period, unless there is reason to doubt that they are valid or typical.

Choosing the mode of volume replication

The decision to use asynchronous or synchronous mode must be made with a complete understanding of the effects of this choice on application and replication performance. The relative merits of using asynchronous or synchronous mode become apparent when you understand the underlying process of replication.

See "Asynchronous volume replication versus synchronous volume replication" on page 69.

Asynchronous mode replication considerations

Asynchronous mode of replication avoids adding the network latency to each write by sending the data to the Secondary after the write is completed to the application. The obvious disadvantage of this is that there is no immediate guarantee that a write that appears complete to the application has actually been replicated. A more subtle effect of asynchronous mode is that while application throughput remains mostly unaffected, overall replication performance may slow down.

In asynchronous mode, the Primary kernel memory buffer fills up if the network bandwidth or the Secondary cannot keep up with the incoming write rate. For VVR to provide memory for incoming writes and continue their processing, it must free the memory held by writes that have been written to the Primary data volume but not yet sent to the Secondary. When VVR is ready to send the unsent writes that were freed, the writes must first be read back from the SRL. Hence, in synchronous mode the data is always available in memory, while in asynchronous mode VVR might have to frequently read back the data from the SRL. Consequently, replication performance might suffer because of the delay of the additional read operation. VVR does not need to read back from the SRL if the network bandwidth and the Secondary always keep up with the incoming write rate, or if the Secondary only falls behind for short periods during which the accumulated writes are small enough

to fit in the VVR kernel buffer. In a shared environment, VVR always reads back from the SRL when replicating in asynchronous mode. You can tune the size of kernel buffers for VVR and VxVM to meet your requirements.

If VVR reads back from the SRL frequently, striping the SRL over several disks using mid-sized stripes (for example, 10 times the average write size), could improve performance. To determine whether VVR is reading back from the SRL, use the `vxstat` command. In the output, note the number of read operations on the SRL.

Synchronous mode replication considerations

Synchronous mode has the advantage that all writes are guaranteed to reach the Secondary before completing. For some businesses, this may simply be a requirement that cannot be circumvented – in this case, performance is not a factor in the decision. For applications where the choice is not so clear, however, this section discusses some of the performance implications of choosing synchronous operations.

All write requests first result in a write to the SRL.

See Figure 2-2 on page 32.

It is only after this write completes that data is sent to the Secondary. Because synchronous mode requires that the data reach the Secondary and be acknowledged before the write completes, this makes the latency for a write equal to:

```
SRL latency + Network round trip latency
```

Thus, synchronous mode can significantly decrease application performance by adding the network round trip to the latency of each write request.

If synchronous mode is to be used, another factor to consider is that if the synchronous attribute is set to fail, synchronous mode throttles incoming write request while catching up from a Storage Checkpoint. this means that if a Secondary takes ten hours to catch up, any application waiting for writes to complete will hang for ten hours if the Secondary is replicating in this mode. For all practical purposes, it is necessary to either shut down the application or temporarily set the mode of replication to asynchronous or set the synchronous attribute to override until the Secondary catches up after a Storage Checkpoint.

If you choose synchronous mode, you must consider what VVR should do if there is a network interruption. In synchronous mode, the `synchronous` attribute enables you to specify what action is taken when the Secondary is unreachable. The `synchronous` attribute can be set to `override` or `fail`. When the `synchronous` attribute is set to `override`, synchronous mode converts to asynchronous during a temporary outage. In this case, after the outage passes and the Secondary catches up, replication reverts to synchronous.

When the `synchronous` attribute is set to `fail`, the application receives a failure for writes issued while the Secondary is unreachable. The application is likely to fail or become unavailable, and hence this setting must be chosen only if such a failure is preferable to the Secondary being out of date.

We recommend setting the `synchronous` attribute to `override`, as this behavior is suitable for most applications. Setting the `synchronous` attribute to `fail` is suitable only for a special class of applications that cannot have even a single write difference between the Primary and Secondary data volumes. In other words, this mode of operation must be used only if you want an application write to fail if the write cannot be replicated immediately. It is imperative that the network connection between hosts using this option must be highly reliable to avert unnecessary application downtime as network outage could cause an application outage.

Synchronous mode considerations when the synchronous attribute is set to fail

When the `synchronous` attribute is set to `fail`, VVR ensures that writes do not succeed if they do not reach the Secondary. If the RLINK is disconnected, the writes fail and are not written either to the SRL or the data volumes. However, if the RLINK was connected but disconnects during the process of sending the writes to the Secondary, it is possible that the writes are written into the SRL and applied to the data volumes even though the application correctly receives failure for these writes. This happens because the data volume writes are asynchronous regardless of the mode of replication.

Note: An SRL volume is required when the `synchronous` attribute is set to `fail`.

See "How VVR uses kernel buffers for replication" on page 32.

The state of the running application on the Primary at this time is no different from that of the application brought up on the Secondary after changing its role to Primary. However, the actual contents of the Primary data volumes and the Secondary data volumes differ, and the Primary data volumes are ahead by these last writes.

Note that as soon as the synchronous RLINK connects, these writes will reach the Secondary, and then the data volumes on the Primary and the Secondary have the same contents. Also, note that at no time is the data consistency being compromised.

If the application is stopped or crashes at this point and is restarted, it recovers using the updated contents of the data volumes. The behavior of the application on the Primary could be different from the behavior of the application when it is brought up on the Secondary after changing its role of the Secondary to Primary, while the RLINK was still disconnected.

In the case of a database application, these writes might be the ones that commit a transaction. If the application tries to recover using the data volumes on the Primary, it will roll forward the transaction because the commit of the transaction is already on the data volume. However, if the application recovers using the data volumes on the Secondary after changing its role to Primary, it will roll back the transaction.

This case is no different from that of an application directly writing to a disk that fails just as it completes part of a write. Part of the write physically reaches the disk but the application receives a failure for the entire write. If the part of the write that reached the disk is the part that is useful to the application to determine whether to roll back or roll forward a transaction, then the transaction would succeed on recovery even though the transaction was failed earlier.

It could also happen that a write was started by the application and the RLINK disconnected and now before the next write is started, the RLINK reconnects. In this case, the application receives a failure for the first write but the second write succeeds.

Different applications, such as file systems and databases, deal with these intermittent failures in different ways. The Veritas File System handles the failure without disabling the file or the file system.

When the synchronous attribute is set to fail, application writes may fail if the RLINK is disconnected. Because auto synchronization or resychronizing requires the RLINK to disconnect in order to completely drain the SRL, to avoid application errors note the following:

- when failing back after takeover, do not start the application on the Primary until the DCM replay is complete, or change the replication mode to asynchronous mode temporarily until the DCM replay completes.
- when synchronizing a Secondary using autosync or with DCM replay, change the replication mode to asynchronous mode temporarily until the synchronization completes.

Asynchronous volume replication versus synchronous volume replication

The decision to use synchronous or asynchronous replication depends on the requirements of your business and the capabilities of your network.

Note: If you have multiple Secondaries, you can have some replicating in asynchronous mode and some in synchronous mode.

See “How data flows in an RDS containing multiple Secondary hosts” on page 39.

Table 3-1 summarizes the main considerations for choosing a mode of replication.

Table 3-1 Comparison of synchronous and asynchronous modes

Considerations	Synchronous mode	Asynchronous mode
Need for Secondary to be up-to-date	Ensures that the Secondary is always current. If the synchronous attribute is set to override, the Secondary is current, except in the case of a network outage.	Ensures that the Secondary reflects the state of the Primary at some point in time. However, the Secondary may not be current. The Primary may have committed transactions that have not been written to the Secondary.
Requirements for managing latency of data	Works best for low volume of writes. Does not require latency protection (because the Secondary is always current).	Could result in data latency on the Secondary. You need to consider whether or not it is acceptable to lose committed transactions if a disaster strikes the Primary, and if so, how many. VVR enables you to manage latency protection, by specifying how many outstanding writes are acceptable, and what action to take if that limit is exceeded.
Characteristics of your network: bandwidth, latency, reliability	Works best in high bandwidth/low latency situations. If the network cannot keep up, the application may be impacted. Network capacity should meet or exceed the write rate of the application at all times.	Handles bursts of I/O or congestion on the network by using the SRL. This minimizes impact on application performance from network bandwidth fluctuations. The average network bandwidth must be adequate for the average write rate of the application. Asynchronous replication does not compensate for a slow network.
Requirements for application performance, such as response time.	Has potential for greater impact on application performance because the I/O does not complete until the network acknowledgment is received from the Secondary.	Minimizes impact on application performance because the I/O completes without waiting for the network acknowledgment from the Secondary.

Choosing latency and SRL protection

The replication parameters `latencyprot` and `srlprot` provide a compromise between synchronous and asynchronous characteristics. These parameters allow the Secondary to fall behind, but limit the extent to which it does so.

When `latencyprot` is enabled, the Secondary is only allowed to fall behind by a predefined number of requests, a latency high mark. After this user-defined latency

high mark is reached, throttling is triggered. This forces all incoming requests to be delayed until the Secondary catches up to within another predefined number of requests, the latency low mark. Thus, the average write latency seen by the application increases. A large difference between the latency high mark and latency low mark causes occasional long delays in write requests, which may appear to be application hangs, as the SRL drains down to the latency low mark. A smaller range spreads the delays more evenly over writes, resulting in smaller but more frequent delays. For most cases, a smaller difference is probably preferable.

The `latencyprot` parameter can be effectively used to achieve the required Recovery Point Objective (RPO). Before setting the `latencyprot` parameter, consider the factors that affect the latency high mark and latency low mark values:

- RPO in writes
- Average write rate
- Average available network bandwidth
- Average write size
- Maximum time required by the SRL to drain from the latency high mark to the latency low mark. This is the timeout value of the application which is the most sensitive, i.e., the application with the LOWEST timeout value among all using volumes from the RVG.
- Number of writes already logged in the SRL

Based on specific requirements, set the user-defined latency high mark to an acceptable RPO value, in terms of number of writes. Thus, the value that should be set for the latency high mark is calculated as RPO in writes divided by average write size.

Set the latency low mark value such that the stalling of writes does not affect any application. Thus, assuming that the average network rate is greater than or equal to the average write rate calculate the effective SRL drain rate as average network rate - average write rate. Once this value is obtained the latency low mark value is calculated as:

```
latency high mark -(Effective SRL drain rate * lowest timeout)/
average write size
```

The replication parameter srlprot can be used to prevent the SRL from overflowing and has an effect similar to latencyprot. However, the srlprot attribute is set to autodcm by default, which allows the SRL to overflow and convert to dcm_logging mode. As a result, there is no effect on write performance, but the Secondary is allowed to fall behind and is inconsistent while it resynchronizes.

See “Protecting against SRL overflow” on page 90.

Planning the network

This section describes the available network protocols for replication in VVR. It also explains how bandwidth requirement depends on the mode of replication—synchronous or asynchronous.

Choosing the network bandwidth

To determine the network bandwidth required for VVR, consider the following factors:

- Bandwidth of the available network connection
- How network performance depends on mode of replication

Bandwidth of the available network connection

The type of connection determines the maximum bandwidth available between the two locations, for example, a T3 line provides 45 megabits/second. However, the important factor to consider is whether the available connection is to be used by any other applications or is exclusively reserved for replicating to a single Secondary. If other applications are using the same line, it is important to be aware of the bandwidth requirements of these applications and subtract them from the total network bandwidth. If any applications sharing the line have variations in their usage pattern, it is also necessary to consider whether their times of peak usage are likely to coincide with peak network usage by VVR. Additionally, overhead added by VVR and the various underlying network protocols reduces effective bandwidth by a small amount, typically 3% to 5%.

How network performance depends on the mode of replication

All replicated write requests must eventually travel over the network to one or more Secondary nodes. Whether or not this trip is on the critical path depends on the mode of replication.

Because replicating in synchronous mode requires that data reach the Secondary node before the write can complete, the network is always part of the critical path for synchronous mode. This means that for any period during which application write rate exceeds network capacity, write latency increases.

Conversely, replicating in asynchronous mode does not impose this requirement, so write requests are not delayed if network capacity is insufficient. Instead, excess requests accumulate on the SRL, as long as the SRL is large enough to hold them. If there is a persistent shortfall in network capacity, the SRL eventually overflows. However, this setup does allow the SRL to be used as a buffer to handle temporary shortfalls in network capacity, such as periods of peak usage, provided that these

periods are followed by periods during which the Secondary can catch up as the SRL drains. If a configuration is planned with this functionality in mind, you must be aware that Secondary sites may be frequently out of date. You can use the `bandwidth_limit` attribute to set the maximum network bandwidth (in bits per second) that can be used during replication.

See "Controlling the network bandwidth used for replication" on page 95.

Several parameters can change the asynchronous mode behavior described above by placing the network round-trip on the critical path in certain situations. The `latencyprot` and `srlprot` features, when enabled, can both have this effect.

See "Choosing latency and SRL protection" on page 70.

To avoid problems caused by insufficient network bandwidth, apply the following principles:

- Apply VVR compression to reduce network bandwidth consumption.
  See "About VVR compression" on page 61.
- If synchronous mode is used, the network bandwidth must at least match the application write rate during its peak usage period; otherwise, the application is throttled. However, this leaves excess capacity during non-peak periods, which is useful to allow synchronization of new volumes using Storage Checkpoints.
  See "Peak usage constraint for sizing the SRL" on page 78.
- If only asynchronous mode is used, and you have the option of allowing the Secondary to fall behind during peak usage, then the network bandwidth only needs to match the overall average application write rate. This might require the application to be shut down during synchronization procedures, because there is no excess network capacity to handle the extra traffic generated by the synchronization.
- If asynchronous mode is used with `latencyprot` enabled to avoid falling too far behind, the requirements depend on how far the Secondary is allowed to fall behind. If the latency high mark is small, replication will be similar to synchronous mode and therefore must have a network bandwidth sufficient to match the application write rate during its peak usage period. If the latency high mark is large, the Secondary can fall behind by several hours. Thus, the bandwidth only has to match the average application write rate. However, the RPO may not be met.

Choosing the network protocol

VVR exchanges two types of messages between the Primary and the Secondary: heartbeat messages and data messages. The heartbeat messages are transmitted

using the UDP transport protocol. VVR can use either the TCP transport protocol or the UDP transport protocol to exchange data messages.

The choice of protocol to use for the data messages is based on the network characteristics. TCP has been found to perform better than UDP on networks that lose packets. However, you must experiment with both protocols to determine the one that performs better in your network environment.

When using the TCP protocol, VVR creates multiple connections, if required, to use the available bandwidth. This is especially useful if there are many out of order packets.

Note: You must specify the same protocol for the Primary and Secondary; otherwise, the nodes cannot communicate and the RLINKs do not connect. This also applies to all nodes in a cluster environment.

VVR uses the TCP transport protocol by default.

See "Setting the network transport protocol for a Secondary" on page 149. for information on how to set the network protocol.

Note: If you specify TCP as your protocol, then by default, VVR does not calculate the checksum for each data packet it replicates. VVR relies on the TCP checksum mechanism. Also, if a node in a replicated data set is using a version of VVR earlier than 5.1 SP1, VVR calculates the checksum regardless of the network protocol.

Choosing the network ports used by VVR

VVR uses the UDP and TCP transport protocols to communicate between the Primary and Secondary. This section lists the default ports used by VVR.

Table 3-2 lists the default ports that VVR uses when replicating data using UDP.

Table 3-2 VVR network ports

Port Numbers	Description
UDP 4145	IANA approved port for heartbeat communication between the Primary and Secondary.
TCP 8199	IANA approved port for communication between the `vradmind` daemons on the Primary and the Secondary.
TCP 8989	Communication between the `in.vxrsyncd` daemons, which are used for differences-based synchronization.

Table 3-2 VVR network ports *(continued)*

Port Numbers	Description
UDP Anonymous ports (OS dependent)	Ports used for each Primary-Secondary connection for data replication between the Primary and the Secondary. One data port is required on each host.

Table 3-3 lists the ports that VVR uses when replicating data using TCP.

Table 3-3 VVR ports using TCP

Port Numbers	Description
UDP 4145	IANA approved port for heartbeat communication between the Primary and Secondary.
TCP 4145	IANA approved port for TCP Listener port.
TCP 8199	IANA approved port for communication between the `vradmind` daemons on the Primary and the Secondary.
TCP 8989	Communication between the `in.vxrsyncd` daemons, which are used for differences-based synchronization.
TCP Anonymous ports	Ports used for each Primary-Secondary connection for data replication between the Primary and the Secondary. One data port is required on each host.

The `vrport` command enables you to view and change the port numbers used by VVR

See "Displaying and changing the ports used by VVR" on page 237.

Configuring VVR in a firewall environment

This section explains how to configure VVR to work in a firewall environment.

VVR uses default port numbers depending on the protocol.

See "Choosing the network ports used by VVR" on page 74.

Additional considerations apply for a Network Address Translation (NAT) based firewall.

See "VVR and network address translation firewall" on page 393.

To configure VVR in a firewall environment when using TCP

- In the firewall, enable the following ports:

- the port used for heartbeats
- the port used by the `vradmind` daemon
- the port used by the `in.vxrsyncd` daemon.

A Primary RVG with one Secondary must have a minimum of one outgoing port in the firewall. For high latency networks, Veritas recommends that you have up to 16 outgoing ports per Primary RVG to allow for multiple connections.

To configure VVR in a firewall environment when using UDP

1. In the firewall, enable the following ports:
   - the port used for heartbeats
   - the port used by the `vradmind` daemon and
   - the port used by the `in.vxrsyncd` daemon.

   Use the `vrport` command to display information about the ports and to change the ports being used by VVR.
2. Set a restricted number of ports to replicate data between the Primary and the Secondary. The operating system assigns anonymous port numbers by default. Most operating systems assign anonymous port numbers between 32768 and 65535. For each Primary-Secondary connection, one data port is required. Use the `vrport` command to specify a list of ports or range of ports to use for VVR.
3. In the firewall, enable the ports that have been set in step 2.

Choosing the packet size

If you have selected the UDP transport protocol for replication, the UDP packet size used by VVR to communicate between hosts could be an important factor in the replication performance. By default, VVR uses a UDP packet size of 8400 bytes. In certain network environments, such as those that do not support fragmented IP packets, it may be necessary to decrease the packet size.

If the network you are using loses many packets, the effective bandwidth available for replication is reduced. You can tell that this is happening if you run `vxrlink stats` on the RLINK, and see many timeout errors.

In this case, network performance may be improved by reducing the packet size. If the network is losing many packets, it may simply be that each time a large packet is lost, a large retransmission has to take place. In this case, try reducing the packet size until the problem is ameliorated.

If some element in the network, such as IPSEC or VPN hardware, is adding to the packets, reduce the packet size so that there is space for the additional bytes in

the packet, and the MTU is not exceeded. Otherwise, each packet is broken into two.

See "Setting the packet size for a Secondary" on page 149.

Choosing the network maximum transmission unit

The UDP packets or TCP packets transmitted by VVR that are of size greater than the network Maximum Transmission Unit (MTU) are broken up into IP packets of MTU size by the IP module of the operating system. There may be losses on the network because the packets are going through routers that do not support IP fragmentation and have a smaller MTU than your network device. In this case, make the MTU size the same as the MTU size of the router with the smallest MTU in the network.

Sizing the SRL

The size of the SRL is critical to the performance of replication. This section describes some of the considerations in determining the size of the SRL.

You can use the Volume Replicator Advisor (`VRAdvisor`) tool to help determine the appropriate SRL size.

See " How VRAdvisor works" on page 433.

If the SRL overflows and SRL protection is not enabled, the RLINK is marked STALE,. Otherwise, if SRL protection is set to `autodcm` or `dcm`, the RLINK is disconnected and it goes into DCM mode. Because resynchronization is a time-consuming process and during this time the data on the Secondary cannot be used, it is important to avoid SRL overflows. The SRL size needs to be large enough to satisfy four constraints:

- It must not overflow for asynchronous RLINKs during periods of peak usage when replication over the RLINK may fall far behind the application.
- It must not overflow while a Secondary RVG is being synchronized.
- It must not overflow while a Secondary RVG is being restored.
- It must not overflow during extended outages (network or Secondary node).

Note: Regardless of the replication mode, the SRL size must be at least 110 MB. If the size that you have specified for the SRL is less than 110 MB, VVR displays an error message which prompts you to specify a value that is equal to or greater then 110 MB.

To determine the size of the SRL, you must determine the size required to satisfy each of these constraints individually. Then, choose a value at least equal to the maximum so that all constraints are satisfied. The information needed to perform this analysis, presented below, includes:

- The maximum expected downtime for Secondary nodes
- The maximum expected downtime for the network connection
- The method for synchronizing Secondary data volumes with data from Primary data volumes. If the application is shut down to perform the synchronization, the SRL is not used and the method is not important. Otherwise, this information could include: the time required to copy the data over a network, or the time required to copy it to a tape or disk, to send the copy to the Secondary site, and to load the data onto the Secondary data volumes.

Note: If the Automatic Synchronization option is used to synchronize the Secondary, the previous paragraph is not a concern.

If you are going to perform Secondary backup to avoid complete resynchronization in case of Secondary data volume failure, the information needed also includes:

- The frequency of Secondary backups
- The maximum expected delay to detect and repair a failed Secondary data volume
- The expected time to reload backups onto the repaired Secondary data volume

Peak usage constraint for sizing the SRL

For some configurations, it might be common for replication to fall behind the application during some periods and catch up during others. For example, an RLINK might fall behind during business hours and catch up overnight if its peak bandwidth requirements exceed the network bandwidth. Of course, for synchronous RLINKs, this does not apply, as a shortfall in network capacity would cause each application write to be delayed, so the application would run more slowly, but would not get ahead of replication.

For asynchronous RLINKs, the only limit to how far replication can fall behind is the size of the SRL. If it is known that the peak write rate requirements of the application exceed the available network bandwidth, then it becomes important to consider this factor when sizing the SRL.

Assuming that data is available providing the typical application write rate over a series of intervals of equal length, it is simple to calculate the SRL size needed to support this usage pattern.

To calculate the SRL size needed

1. Calculate the network capacity over the given interval (BW_N).
2. For each interval *n*, calculate SRL log volume usage (LU_n), as the excess of application write rate (BW_{AP}) over network bandwidth ($LU_n = BW_{AP(n)} - BW_N$).

   In a shared environment, you must consider the write rates on all the nodes in the cluster. The application write rate (BW_{AP}) should reflect the aggregate of the write rates on each node.
3. For each interval, accumulate all the SRL usage values to find the cumulative SRL log size (LS):

$$LS_n = \sum_{i=1 \ldots n} LU_i$$

   The largest value obtained for any LS_n is the value that should be used for SRL size as determined by the peak usage constraint.

Synchronization period constraint for sizing the SRL

When a new Secondary is added to an RDS, its data volumes must be synchronized with those of the Primary unless the Primary and the Secondary data volumes have been zero initialized and the application has not yet been started. You also need to synchronize the Secondary after a Secondary data volume failure, in case of SRL overflow, or after replication is stopped.

This section applies if you choose not to use the automatic synchronization method to synchronize the Secondary. Also, this constraint does not apply if you choose to use a method other than automatic synchronization and if the application on the Primary can be shut down while the data is copied to the Secondary. However, in most cases, it might be necessary to synchronize the Secondary data volumes with the Primary data volumes while the application is still running on the Primary.

During the synchronization period, the application is running and data is accumulating in the SRL. If the SRL overflows during the process of synchronization, the synchronization process must be restarted. Thus, to ensure that the SRL does not overflow during this period, it is necessary that the SRL be sized to hold as much data as the application writes during the synchronization period. After starting replication, this data is replicated and the Secondary eventually catches up with the Primary.

Depending on your needs, it may or may not be possible to schedule the synchronization during periods of low application write activity. If it is possible to complete the synchronization process during a period of low application write activity, then you must ensure that the SRL is sized such that it can hold all the incoming writes during this period. Otherwise, the SRL may overflow.

If however there is an increase in the application write activity then you may need to resize the SRL even when the synchronization is in progress.

If it is not possible to complete the synchronization process during periods of low application write activity, then size the SRL such that it uses either the average value, or to be safer, the peak value.

See "Understanding application characteristics" on page 65.

Secondary backup constraint for sizing the SRL

VVR provides a mechanism to perform periodic backups of the Secondary data volumes. In case of a problem that would otherwise require a complete resynchronization, a Secondary backup, if available, can be used to bring the Secondary online much more quickly.

A Secondary backup is made by creating a Secondary Storage Checkpoint and then making a raw copy of all the Secondary data volumes. Should a failure occur, the Secondary data volumes are restored from this local copy, and then replication proceeds from the Storage Checkpoint, thus replaying all the data from the Storage Checkpoint to the present. The constraint introduced by this process is that the Primary SRL must be large enough to hold all the data logged in the Primary SRL after the creation of the Storage Checkpoint corresponding to the most recent backup.

In this case, the constraint for the SRL depends largely on the following factors:

- The application write rate.
- The frequency of Secondary backups.

Thus, given an application write rate and frequency of Secondary backups, it is possible to come up with a minimal SRL size. Realistically, an extra margin should be added to an estimate arrived at using these figures to cover other possible delays, including:

- Maximum delay before a data volume failure is detected by a system administrator.
- Maximum delay to repair or replace the failed drive.
- Delay to reload disk with data from the backup tape.

To arrive at an estimate of the SRL size needed to support this constraint, first determine the total time period the SRL needs to support by adding the period planned between Secondary backups to the time expected for the three factors mentioned above. Then, use the application write rate data to determine, for the worst case, the amount of data the application could generate over this time period.

Note: Even if only one volume failed, all volumes must be restored.

Secondary downtime constraint for sizing the SRL

When the network connection to a Secondary node, or the Secondary node itself, goes down, the RLINK on the Primary node detects the broken connection and responds. If the RLINK has its `synchronous` attribute set to `fail`, the response is to fail all subsequent write requests until the connection is restored. In this case, the SRL does not grow, so the downtime constraint is irrelevant. For all other types of RLINKs, incoming write requests accumulate in the SRL until the connection is restored. Thus, the SRL must be large enough to hold the maximum output that the application could be expected to generate over the maximum possible downtime.

Maximum downtimes may be difficult to estimate. In some cases, the vendor may guarantee that failed hardware or network connections will be repaired within some period. Of course, if the repair is not completed within the guaranteed period, the SRL overflows despite any guarantee, so it is a good idea to add a safety margin to any such estimate.

To arrive at an estimate of the SRL size needed to support this constraint, first obtain estimates for the maximum downtimes which the Secondary node and network connections could reasonably be expected to incur. Then, use the application write rate data to determine, for the worst case, the amount of data the application could generate over this time period. With the introduction of the `autodcm` mode of SRL overflow protection, sizing the SRL for downtime is not essential to prevent SRL overflow because the changed blocks are no longer stored in the SRL. However, note that the Secondary is inconsistent during the replay of the DCM, and hence it is still important for the SRL to be large enough to cover most eventualities.

Additional factors for sizing the SRL

Once estimates of required SRL size have been obtained under each of the constraints described above, several additional factors must be considered.

For the synchronization period, downtime and Secondary backup constraints, it is not unlikely that any of these situations could be immediately followed by a period of peak usage. In this case, the Secondary could continue to fall further behind

rather than catching up during the peak usage period. As a result, it might be necessary to add the size obtained from the peak usage constraint to the maximum size obtained using the other constraints. Note that this applies even for synchronous RLINKs, which are not normally affected by the peak usage constraint, because after a disconnect, they act as asynchronous RLINKs until caught up.

Of course, it is also possible that other situations could occur requiring additions to constraints. For example, a synchronization period could be immediately followed by a long network failure, or a network failure could be followed by a Secondary node failure. Whether and to what degree to plan for unlikely occurrences requires weighing the cost of additional storage against the cost of additional downtime caused by SRL overflow.

Once an estimate has been computed, one more adjustment must be made to account for the fact that all data written to the SRL also includes some header information. This adjustment must take into account the typical size of write requests. Each request uses at least one additional disk block for header information.

Table 3-4 provides the SLR adjustment percentages required.

Table 3-4 SRL adjustments

If average write size is:	Add this percentage to SRL size:
512 bytes	100%
1K	50%
2K	25%
4K	15%
8K	7%
10K	5%
16K	4%
32K or more	2%

Example - Calculating SRL size for a VVR configuration

This section shows how to calculate the SRL size for a VVR configuration after you collect the site parameters.

Table 3-5 provides the relevant parameters for SLR size calculations.

Table 3-5 Parameters for calculating the SRL size

Parameter	Value
Application peak write rate	1 gigabyte/hour
Duration of peak	8 am - 8 pm
Application off-peak write rate	250 megabytes/hour
Average write size	2 kilobytes
Number of Secondary sites	1
Type of RLINK	synchronous=override
Synchronization Period:	
- application shutdown	no
- copy data to tape	3 hours
- send tapes to Secondary site	4 hours
- load data	3 hours
- Total	10 hours
Maximum downtime for Secondary node	4 hours
Maximum downtime for network	24 hours
Secondary backup	not used

Because synchronous RLINKs are to be used, the network bandwidth must be sized to handle the peak application write rate to prevent the write latency from growing. Thus, the peak usage constraint is not an issue, and the largest constraint is that the network could be out for 24 hours. The amount of data accumulating in the SRL over this period would be:

(Application peak write rate x Duration of peak) + (Application off-peak write rate x Duration of off peak).

In this case, the calculation would appear as follows:

```
1 GB/hour x 12 hours+ 1/4 GB/hour x 12 = 15 GB
```

An adjustment of 25% is made to handle header information. Since the 24-hour downtime is already an extreme case, no additional adjustments are needed to handle other constraints. The result shows that the SRL should be at least 18.75 gigabytes.

Best practices for setting up replication

Set up replication according to the following best practices:

- Create one RVG for each application, rather than for each server. For example, if a server is running three separate databases that are being replicated, create three separate RVGs for each database. Creating three separate RVGs helps to avoid write-order dependency between the applications and provides three separate SRLs for maximum performance per application.
- Create one RVG per disk group. Creating one RVG per disk group enables you to efficiently implement application clustering for high availability, where only one RVG needs to be failed over by the service group. If the disk group contains more than one RVG, the applications using the other RVGs would have to be stopped to facilitate the failover. You can use the Disk Group Split feature to migrate application volumes to their own disk groups before associating the volumes to the RVG.
- Plan the size and layout of the data volumes based on the requirement of your application.
- Plan the size of the network between the Primary and each Secondary host.
- Lay out the SRL appropriately to support the performance characteristics needed by the application. Because all writes to the data volumes in an RVG are first written to the SRL, the total write performance of an RVG is bound by the total write performance of the SRL. For example, dedicate separate disks to SRLs and if possible dedicate separate controllers to the SRL.
- Size the SRL appropriately to avoid overflow.
  See “Sizing the SRL” on page 77.
  The Volume Replicator Advisor (VRAdvisor), a tool to collect and analyze samples of data, can help you determine the optimal size of the SRL.
  See “Overview of VRAdvisor” on page 432.
- Include all the data volumes used by the application in the same RVG. This is mandatory.
- Provide dedicated bandwidth for VVR over a separate network. The RLINK replicates data critical to the survival of the business. Compromising the RLINK compromises the business recovery plan.
- Use the same names for the data volumes on the Primary and Secondary nodes. If the data volumes on the Primary and Secondary have different names, you must map the name of the Secondary data volume to the appropriate Primary data volume.
  See “Mapping the name of a Secondary data volume to a differently named Primary data volume” on page 214.

- Use the same name and size for the SRLs on the Primary and Secondary nodes because the Secondary SRL becomes the Primary SRL when the Primary role is transferred.
- Mirror all data volumes and SRLs. This is optional if you use hardware-based mirroring.
- The `vradmin` utility creates corresponding RVGs on the Secondary of the same name as the Primary. If you choose to use the `vxmake` command to create RVGs, use the same names for corresponding RVGs on the Primary and Secondary nodes.
- Associate a DCM to each data volume on the Primary and the Secondary if the DCMs had been removed for some reason. By default, the `vradmin createpri` and `vradmin addsec` commands add DCMs if they do not exist.
- In a shared disk group environment, currently only the cvm master node should be assigned the logowner role. It is recommended to enable the PreOnline trigger for the RVGLogOwner agent, so that the VVR logowner always resides on the CVM master node.
- In a shared disk group environment, currently only the cvm master node should be assigned the logowner role.
- The on-board write cache should not be used with VVR. The application must also store data to disk rather than maintaining it in memory. The takeover system, which could be a peer primary node in case of clustered configurations or the secondary site, must be capable of accessing all required information. This requirement precludes the use of anything inside a single system inaccessible by the peer. NVRAM accelerator boards and other disk caching mechanisms for performance are acceptable, but must be done on the external array and not on the local host.

How the agents for hybrid applications work

The agents for hybrid applications include the following:

- RVG agent
- RVGPrimary

A hybrid configuration is for Replicated Data Clusters (RDCs) and is a combination of the failover and parallel service groups. A hybrid service group behaves like a failover group within a system zone and like a parallel group across system zones. It cannot fail over across system zones. A switch operation on a hybrid service group is allowed only between systems within the same system zone.

These sections give information about the entry points, state definitions, and attributes for the RVG agent and the RVGPrimary agent. In addition, the following attribute must be set for the RVG agent and the RVGPrimary agent while configuring RDCs:

For more information about the RVG agent and RVG Primary agent, see the *Cluster Server Bundled Agents Reference Guide*.

Table 3-6 Attribute for RDCs

Optional attributes	Type and dimension	Definition
SystemZones	integer-association	Indicates failover zone.

An RDC uses VVR as opposed to shared storage to provide access to data at the Secondary. An RDC exists within a single VCS cluster. The application group, which is configured as a failover group, can be online only on the Primary host. In the case of the failure of the Primary site, the Secondary is promoted to a Primary and the application is brought online on the new Primary host.

An RDC configuration is appropriate in configurations lacking shared storage or SAN interconnection between the Primary site and Secondary site, but where dual dedicated LLT links are available between the Primary site and the Secondary site.

Chapter 4

Understanding replication settings for a Secondary

This chapter includes the following topics:

- About replication settings for a Secondary
- Modes of VVR replication
- Protecting against SRL overflow
- Setting up latency protection
- Controlling the network bandwidth used for replication
- Choosing the VVR compression mode

About replication settings for a Secondary

The VVR replication settings determine the replication behavior between the Primary RVG and a specific Secondary RVG. VVR behaves differently based on the settings for mode of replication, SRL overflow protection, and latency protection, depending on whether the Secondary is connected or disconnected. To use the replication settings effectively in your environment, it is important to understand how each replication setting affects replication when the Primary and Secondary are connected and disconnected. A Secondary is said to be disconnected from the Primary if the RLINK becomes inactive because of a network outage or administrative action.

VVR enables you to set up the replication mode, latency protection, and SRL protection using the replication attributes. Each attribute setting could affect replication and must be set up with care.

Modes of VVR replication

VVR replicates in synchronous and asynchronous modes. In synchronous mode, a write must be recorded in the Primary SRL and posted to the Secondary before the write completes at the application level. When replicating in asynchronous mode, an update to the Primary volume is complete when it has been recorded in the Primary SRL. The decision to use synchronous or asynchronous mode must be made with an understanding of the effects of this choice on the replication process and the application performance.

You can set the mode of replication between the Primary and each Secondary depending on your requirement. You can replicate to the Secondary hosts of an RDS in different replication modes.

In each mode, VVR replicates and completes the application writes differently. Each mode deals differently with network conditions. The following sections provide a brief description of synchronous and asynchronous modes of replication and discuss some of the issues involved in choosing between the two.

See "Choosing the mode of volume replication" on page 66.

About asynchronous volume replication

Asynchronous mode is useful when it is acceptable for the Secondary not to be up-to-date. When replicating in asynchronous mode, an update to the Primary volume is complete when it has been recorded in the Primary SRL. In asynchronous mode, all completed updates to the Primary volumes are guaranteed to be made on the Secondary data volumes with some delay. This is true despite failures in communication or system crashes on any of the participating hosts.

The application is informed that the write request is complete and the write is queued persistently to be sent to the Secondary. This queue may grow when there is a surge in the write rate, while the queue is being continuously drained. When the surge subsides, the queue drains faster than it grows enabling the Secondary to catch up with the Primary. Because VVR persistently records the queue of ongoing writes and holds them at the Primary for later transmission, it can deal with temporary outages of the network or the Secondary host without affecting the performance of the application. However, asynchronous mode has the disadvantage that if a disaster occurs, it is likely that the most recent writes have not reached the Secondary and therefore the data at a Secondary is not up-to-date when a failover occurs.

See "Asynchronous mode replication considerations" on page 66.

About synchronous volume replication

Synchronous mode ensures that a write has been recorded in the Primary SRL and posted to the Secondary before the write completes at the application level. In synchronous mode, the data on the Secondary is completely up-to-date and if a disaster occurs at the Primary, data can be recovered from any surviving Secondary without any loss. If the Secondary must reflect all writes that have successfully completed on the Primary in the case of a disaster, synchronous mode is the correct choice.

Synchronous replication keeps the Secondary up-to-date with the Primary by waiting for each write to reach the Secondary before the application sees the successful completion of the write on the Primary.

Synchronous replication keeps the Secondary up-to-date with the Primary, but it can impact application performance in high latency or limited bandwidth environments. When using synchronous replication, the response time experienced by the application is affected because the write has to wait for the Secondary to acknowledge it before the write can complete on the Primary.

Synchronous replication is most effective in application environments with low update rates. However, it can be deployed in write-intensive environments where high bandwidth, low latency network connections are available.

The performance of synchronous replication could degrade significantly if the peak application write rate exceeds the available network bandwidth. The amount of degradation can be reduced by increasing network bandwidth and reducing network latency between the Primary and Secondary.

See "How network performance depends on the mode of replication" on page 72.

About the synchronous attribute

You can set up VVR to replicate to a Secondary in synchronous or asynchronous mode by setting the `synchronous` attribute of the RLINK to `override`, `off`, or `fail`.

Table 4-1 summarizes how the state of the RLINK affects the mode of replication.

Table 4-1 Mode of replication and the state of the RLINK

Value of `synchronous` Attribute	When RLINK is Connected	When RLINK is Disconnected
`override`	Synchronous	Asynchronous
`off`	Asynchronous	Asynchronous
`fail`	Synchronous	I/O error to application

synchronous=off

By default, VVR sets the `synchronous` attribute to `off`. Setting the attribute of an RLINK to `synchronous=off` sets the replication between the Primary and the Secondary to asynchronous mode.

synchronous=override

Setting the `synchronous` attribute to `override` puts the RLINK in synchronous mode and specifies override behavior if the RLINK is disconnected. During normal operation, VVR replicates in synchronous mode. If the RLINK is disconnected, VVR switches temporarily to asynchronous mode and continues to receive writes from the application and logs them in the SRL. After the connection is restored and the RLINK is up-to-date, the RLINK automatically switches back to synchronous mode. Most system administrators would prefer to set the `synchronous` attribute to `override` rather than setting it to `fail`.

Warning: If you use the `synchronous=fail` mode, you must read the section about synchronous mode considerations.

See "Synchronous mode replication considerations" on page 67.

Setting the `synchronous` attribute to `fail` puts the RLINK in synchronous mode and specifies the behavior if the RLINK is disconnected. During normal operation, VVR replicates in synchronous mode. If the RLINK is disconnected, VVR fails incoming writes to the Primary.

Protecting against SRL overflow

The Primary SRL maintains writes until they are written to the Secondary. A write is removed from the Primary SRL when the Primary receives the data acknowledgment from all of the Secondary RVGs. If the network is down or the Secondary is unavailable, the number of writes in the SRL waiting to be sent to the Secondary could increase until the SRL fills up. When the SRL cannot accommodate a new write without overwriting an existing one, the condition is called SRL overflow. At this point, the new writes are held up or the RLINK overflows depending on the mode of SRL overflow protection.

Several situations could cause the SRL to overflow, including the following circumstances:

- A temporary burst of writes or a temporary congestion in the network causing the current update rate to exceed the currently available bandwidth between the Primary and the Secondary.

- A temporary failure of the Secondary node or the network connection between the Secondary and the Primary.
- Replication is paused by an administrator.
- The network bandwidth is unable, on a sustained basis, to keep up with the update rate at the Primary. This is not a temporary condition and can be corrected only by increasing the network bandwidth or reducing the application update rate, if possible.

If the SRL overflows, the Secondary becomes out-of-date and must be completely synchronized to bring it up-to-date with the Primary. The SRL Protection feature of VVR enables you to either prevent the SRL from overflowing or tracks the writes using the Data Change Map (DCM) if the SRL overflows. You must weigh the trade-off between allowing the overflow or affecting the application. You can prevent SRL overflow using the `srlprot` attribute.

If there are multiple Secondaries, each Secondary receives data at its own rate. The point of SRL overflow is specific to each Secondary, and the `srlprot` attribute can be set for each Secondary.

About the srlprot attribute

VVR provides the following modes of SRL overflow protection: `autodcm`, `dcm`, `fail`, or `override`. VVR activates these modes only when the SRL overflows. You can set up SRL protection by setting the `srlprot` attribute of the corresponding RLINKs to `autodcm`, `dcm`, `fail`, or `override`. By default, the `srlprot` attribute is set to `autodcm`.To disable SRL protection, set the `srlprot` attribute to `off`.

Table 4-2 summarizes how the state of the RLINK affects SRL protection when the SRL is about to overflow.

Table 4-2 SRL protection and the state of the RLINK

Value of the `srlprot` Attribute	When RLINK is Connected	When RLINK is Disconnected
`autodcm`	Convert to DCM logging	Convert to DCM logging
`dcm`	Protect Protects by stalling application writes until SRL drains 5% to 95% full or drains 20 megabytes, whichever is smaller.	Convert to DCM logging

Table 4-2 SRL protection and the state of the RLINK *(continued)*

Value of the `srlprot` Attribute	When RLINK is Connected	When RLINK is Disconnected
`fail`	SRL protection is enabled.	Write requests are failed with an EIO error
`off`	SRL protection is disabled.	SRL protection is disabled.
`override`	Protect Protects by stalling application writes until SRL drains 5% to 95% full or drains 20 megabytes, whichever is smaller.	Overflow

If the SRL overflow protection is set to `autodcm`, `dcm`, `fail`, or `override`, SRL overflow protection is enabled. The replication setting for the Secondary and the state of the connection between the Primary and the Secondary determines how VVR works when the SRL is about to overflow.

srlprot=autodcm

VVR activates the DCM irrespective of whether the Primary and Secondary are connected or disconnected. Each data volume in the RVG must have a DCM; note that VVR does not stall writes when `srlprot` is set to `autodcm`.

srlprot=dcm

If the Primary and Secondary are connected, new writes are stalled in the operating system of the Primary host until a predetermined amount of space, that is 5% or 20 MB, whichever is less, becomes available in the SRL.

If the Primary and Secondary are disconnected, DCM protection is activated and writes are written to the DCM. Each data volume in the RVG must have a DCM.

srlprot=fail

If the Primary and Secondary are connected, SRL protection is enabled. If the RLINK becomes inactive for any reason, and SRL overflow is imminent, subsequent write requests are failed with an EIO error.

srlprot=override

If the Primary and Secondary are connected, new writes are stalled in the operating system of the Primary host until a predetermined amount of space, that is 5% or 20 MB, whichever is less, becomes available in the SRL.

srlprot=off

SRL overflow protection is disabled, regardless of whether the Primary and Secondary are connected or disconnected.

If the Primary and Secondary are disconnected, VVR disables SRL protection and lets the SRL overflow.

See "Setting the SRL overflow protection for a Secondary" on page 148.

VVR in a Flexible Storage Sharing environment

In a Flexible Storage Sharing (FSS) environment, the DCM log may not get automatically mirrored across the host failure domains. To ensure that the DCM log is mirrored across hosts, allocation attributes such as daname may have to be specified manually using the `vxassist` command.

Setting up latency protection

Excessive lag between the Primary and the Secondary could be a liability in asynchronous replication. The Latency Protection feature of VVR protects the Secondary host from falling too far behind in updating its copy of data when replicating in asynchronous mode. This feature limits the number of outstanding writes lost in a disaster enabling automatic control of excessive lag between Primary and Secondary hosts when you replicate in asynchronous mode.

When replicating in asynchronous mode, it is normal for the SRL to have writes waiting to be sent to the Secondary. If your network has been sized based on the average update rate of the application on the Primary node, the number of writes waiting in the Primary SRL is likely to be within an acceptable range.

The number of writes in the SRL would grow under the following circumstances:

- A temporary burst of writes or a temporary congestion in the network, which causes the current update rate to exceed the currently available bandwidth between the Primary and the Secondary.
- A temporary failure of the Secondary node or the network connection between the Secondary and the Primary.
- Replication is paused by an administrator.
- The network bandwidth is unable, on a sustained basis, to keep up with the write rate at the Primary. This is not a temporary condition and can be corrected only by increasing the network bandwidth or reducing the application write rate if possible.

If the Primary SRL has a large number of writes waiting to be transferred to the Secondary, the Secondary data is considerably behind the Primary. If a disaster strikes the Primary and the Secondary takes over, the Secondary would not contain

all the data in the Primary SRL. In this case, the data on the Secondary would be consistent but significantly out of date when the Secondary takes over. To prevent the Secondary from being too far behind the Primary in this scenario, you can limit the number of writes waiting in the Primary SRL for transmission to the Secondary by setting up latency protection.

About the latencyprot attribute

Latency protection has two components, its mode, and the `latency_high_mark` and `latency_low_mark` which specify when the protection is active or inactive. The `latency_high_mark` specifies the maximum number of waiting updates in the SRL before the protection becomes active and writes stall or fail, depending on the mode of latency protection.

The `latency_low_mark` must be a number lower than the `latency_high_mark`; the `latency_low_mark` is the number of writes in the SRL when the protection becomes inactive and writes succeed. You can set up latency protection by setting the `latencyprot` attribute to either `override` or `fail`. Setting the attribute to `latencyprot=off`, which is the default, disables latency protection.

Setting the attribute to `latencyprot=fail` or `override` enables latency protection. The following sections explain how VVR controls replication depending on the setting of the `latencyprot` attribute of the RLINK when the Primary and Secondary either connected or disconnected.

Table 4-3 summarizes how the state of the RLINK affects the latency protection.

Table 4-3 Latency protection and the state of the RLINK

Value of `latencyprot` Attribute	When RLINK is Connected	When RLINK is Disconnected
`override`	Protect*	Drop protection
`off`	No protection	No protection
`fail`	Protect*	I/O error to application

About Latency protection when Primary and Secondary are connected

latencyprot=fail or override

Under normal operation, if the number of waiting writes increase and reach the `latency_high_mark`, following writes are stalled in the operating system of the

Primary until the SRL drains sufficiently to bring the number of waiting writes below the `latency_low_mark`.

About Latency protection when Primary and Secondary are disconnected

Primary and Secondary are said to be disconnected when they are in the paused state or are disconnected because of a network outage, or an outage of the Secondary node.

latencyprot=override

VVR allows the number of writes in the SRL to exceed the `latency_high_mark`. In this case, VVR causes latency protection to be overridden and allows incoming writes from the application whose data is being replicated. VVR does not stall incoming writes because the SRL is currently not draining and incoming writes may be stalled indefinitely. Stalling of incoming writes is undesirable for the writing application. Most system administrators set `latencyprot=override`.

If replication is paused and not resumed, or if there is an extended network outage, the outstanding writes can exceed the latency high mark. When the Secondary reconnects either because replication is resumed or the network becomes available, VVR starts stalling writes until the writes in the SRL reach the latency low mark. The time required for the Primary to send the accumulated writes to the Secondary can take a long time depending on the amount of data to be sent and the bandwidth of the network. The application perceives this as VVR being unresponsive and some applications may time out resulting in application error.

latencyprot=fail

If the number of writes in the SRL reaches the `latency_high_mark` while the Primary and the Secondary are disconnected, VVR causes new writes at the Primary to fail. This prevents the Secondary from falling further behind than specified by the `latency_high_mark`.

Controlling the network bandwidth used for replication

VVR uses the network to replicate data from the Primary to the Secondary. The Bandwidth Throttling feature enables you to control the maximum network bandwidth to be used by VVR for replication. Bandwidth Throttling controls the rate at which data is sent from the Primary to the Secondary; it does not limit the rate at which the network acknowledgments are sent from the Secondary to the Primary.

You might want to control the bandwidth used by VVR depending on factors such as whether the available network connection is to be used by other applications or exclusively for VVR, the network costs, and network usage over time. For example, if the network is used for purposes other than replication, you might have to control the network bandwidth used by VVR.

Decide on the bandwidth limit for VVR depending on the bandwidth required for VVR, and the bandwidth required for other purposes. You can use the VRAdvisor to determine the bandwidth limit to specify for VVR.

See "Choosing the network bandwidth" on page 72.

See "Overview of VRAdvisor" on page 432.

VVR enables you to change the network bandwidth used for replication to a Secondary even when replication is in progress. It is not necessary to pause replication before you change the bandwidth limit for a Secondary or for an RDS.

Use the `vrstat` command to determine the network bandwidth currently used by VVR.

See "Determining VVR network bandwidth usage and compression ratio" on page 182.

Use the `bandwidth_limit` attribute of the `vradmin set` command to set the limit on the network bandwidth used to replicate from the Primary to the Secondary. For example, if `bandwidth_limit` is set to `30 mbps`, VVR uses 30 mbps for replication. If `bandwidth_limit` is set to `none`, then VVR uses the available network bandwidth. The default value is `none`.

You can also control the network bandwidth used by VVR when synchronizing volumes, which are not part of an RDS; use the `bandwidth_limit` attribute of the `vradmin syncvol` command to specify the limit.

Note: This value of `bandwidth_limit` specified in the `vradmin syncvol` command is in addition to the bandwidth limit set for replication.

For example, if `bandwidth_limit` is set to `30 mbps` for replication between a Primary and Secondary in an RDS, and the bandwidth limit to be used when synchronizing volumes that are not part of an RDS using the `vradmin syncvol` command is specified as 10 mbps, then together VVR will use a maximum of 40 mbps.

Choosing the VVR compression mode

This chapter describes how to administer VVR compression using the CLI.

The VVR compression features enables you to send data in a compressed format from a Primary to one or more Secondary hosts. The compression mode can be enabled on a per system or per Secondary basis. If the per Secondary setting is not set, then the per system basis default settings are used. At any time, you can enable or disable compression online without stopping replication.

The VVR compression feature parameters can also be tuned.

See the *Storage Foundation and High Availability Solutions Tuning Guide* for more information on tuning replication.

See "About VVR compression" on page 61.

Enabling compression

The `vradmin set` command allows you to enable compression on a per secondary host basis.

To enable compression

1. Enable compression using the following command:

   ```
   # vradmin -g diskgroup set local_rvgname [sec_hostname]
   compression=on
   ```

   The argument *local_rvgname* is the name of the rvg on the local host and represents the RDS.

   The argument *sec_hostname* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command. This argument is optional when only one secondary is configured.

2. Verify that compression has been enabled using the following command:

   ```
   # vradmin -g diskgroup -l repstatus local_rvgname
   ```

The output resembles:

```
Replicated Data Set: hr_rvg
Primary:
  Host name:                 seattle
  RVG name:                  hr_rvg
  DG name:                   hrdg
  RVG state:                 enabled for I/O
  Data volumes:              4
  Vsets:                     1
  SRL name:                  hr_srl
```

```
  SRL size:                   4.00 GB
  Total secondaries:          1

Secondary:
  Host name:                  london
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Rlink from Primary:         rlk_london_hr_rvg
  Rlink to Primary:           rlk_seattle_hr_rvg
  Configured mode:            asynchronous
  Latency protection:         off
  SRL protection:             autodcm
  Data status:                consistent, up-to-date
  Replication status:         replicating (connected)
  Current mode:               asynchronous
  Logging to:                 SRL
  Timestamp Information:      behind by 0h 0m 0s
  Bandwidth Limit:            N/A
  Compression Mode:           On
```

Disabling compression

The `vradmin set` command allows you to disable compression on a per secondary host basis.

To disable compression

1. Disable compression using the following command:

   ```
   # vradmin -g diskgroup set local_rvgname [sec_hostname] compression=off
   ```

   The argument *local_rvgname* is the name of the rvg on the local host and represents the RDS.

   The argument *sec_hostname* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command. This argument is optional when only one secondary is configured.

2. Verify that compression has been disabled using the following command:

   ```
   # vradmin -g diskgroup -l repstatus local_rvgname
   ```

The output resembles:

```
Replicated Data Set: hr_rvg
Primary:
```

```
  Host name:                  seattle
  RVG name:                   hr_rvg
  DG name:                    hrdg
  RVG state:                  enabled for I/O
  Data volumes:               4
  Vsets:                      1
  SRL name:                   hr_srl
  SRL size:                   4.00 GB
  Total secondaries:          1

Secondary:
  Host name:                  london
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Rlink from Primary:         rlk_london_hr_rvg
  Rlink to Primary:           rlk_seattle_hr_rvg
  Configured mode:            asynchronous
  Latency protection:         off
  SRL protection:             autodcm
  Data status:                consistent, up-to-date
  Replication status:         replicating (connected)
  Current mode:               asynchronous
  Logging to:                 SRL
  Timestamp Information:      behind by 0h 0m 0s
  Bandwidth Limit:            N/A
  Compression Mode:           Off
```

Chapter 5

Configuring VVR in a VCS environment

This chapter includes the following topics:

- Overview of how to configure VVR in a VCS environment
- Using the primary-elect feature to choose the primary site after a site disaster or network disruption
- Requirements for configuring VVR in a VCS environment
- Generic VVR setup in a VCS environment
- Example VVR configuration in a VCS environment
- Example RVG configuration for a failover application
- Example RVG configuration for a parallel application
- Example setting up VVR in a VCS environment
- Configuring the agents for a bunker replication configuration
- Administering VCS service groups

Overview of how to configure VVR in a VCS environment

This section gives an overview of how to configure VVR in a VCS environment for high availability of the application that is involved in replication.

To configure VVR in a VCS environment, you must perform the following tasks in the following order:

- Set up a VVR configuration, which involves creating a Replicated Data Set (RDS).
- Create service groups for the VVR agents and add the resource and group dependencies appropriately.
- Add these application service groups to the composite service group for global failover.

Using the primary-elect feature to choose the primary site after a site disaster or network disruption

VCS global clustering monitors and manages the replication jobs and clusters at each site. In the event of a site outage, global clustering controls the shift of replication roles to the Secondary site, bring up the critical applications and redirects client traffic from one cluster to the other.

Before Release 5.1SP1, if there was a disaster at the Primary site or a network disruption, the applications were taken offline on the original Primary and failed over to the Secondary. When the original Primary returned or the network disruption was corrected, you had the following options:

- Manually resynchronize the original Primary with the data from the new Primary, once the original Primary comes back up. The applications are only active on the new Primary site.
- Automatically resynchronize the original Primary with the data from the new Primary, once the original Primary comes back up. The applications are only active on the new Primary site.

Beginning in Release 5.1SP1, you have a third option. Applications can be active on both the original Primary and Secondary sites. After the original Primary returns or the network disruption is corrected, you have the option of specifying which site is the Primary going forward. This option is called the primary-elect feature, and it is enabled through the VCS global clustering.

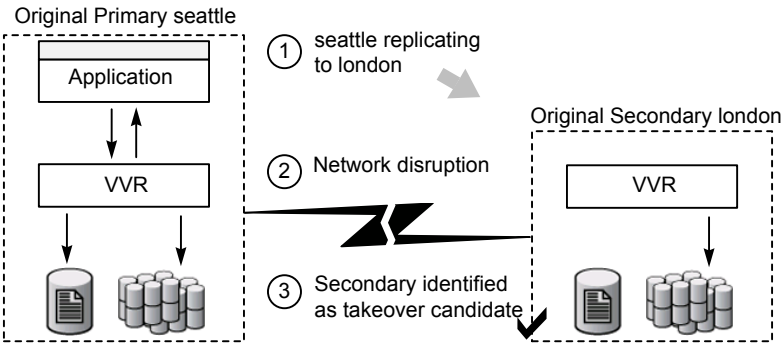
The key difference between the primary-elect feature and the other options is that if a network disruption occurs, applications continue to run on the Primary site and they are also failed over to the Secondary. This feature lets you maintain application availability on both sites while the network is down.

Note: You cannot use the primary-elect feature and the automated bunker replay feature in the same environment. If you set the `AutoResync` attribute to `2` (to enable the primary-elect feature), the value of the `BunkerSyncTimeOut` attribute must be `0` to disable the automated bunker replay feature. Similarly, if you set the `BunkerSyncTimeOut` attribute to a non-zero value, the `AutoResync` attribute cannot be `2`.

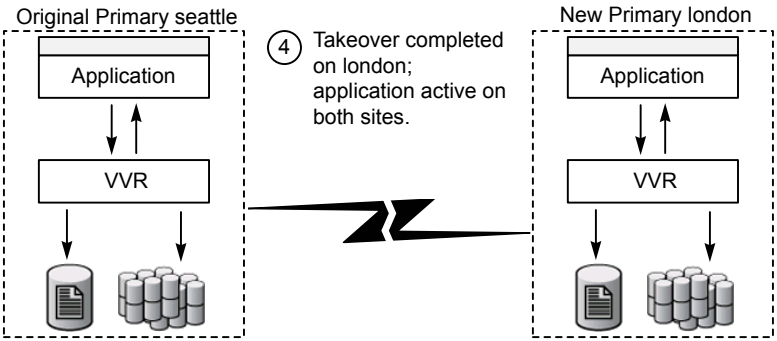
Application availability in the case of a network disruption using the primary-elect feature

The following diagrams illustrate the primary-elect feature. This feature is most useful in the case of the network disruption. It ensures application availability on both sites, even though the network is down.

In this example, the Primary site (`seattle`) is replicating data to the Secondary host (`london`) when a network disruption occurs. `london` has been identified as the Secondary for takeover.



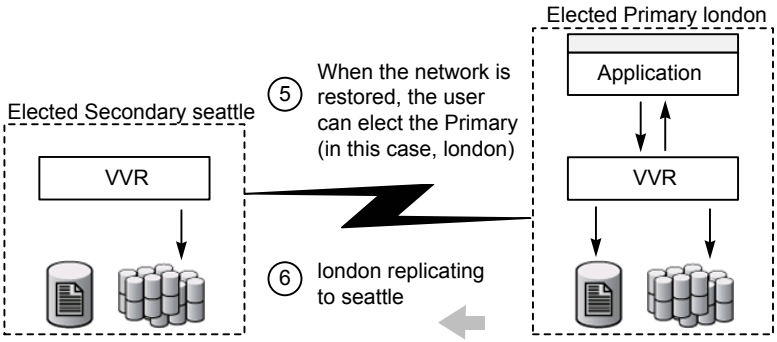
After the takeover, `london` becomes the new Primary. The applications come online on `london`.



Because this event is a network disruption and not a site failure, the original Primary site, `seattle`, is still active. In the primary-elect feature, the applications on the original Primary site are not taken offline. Instead, application data is now being written to both sites, `seattle` and `london`.

Note: In the case of a Primary site failure, the applications are taken offline, even if you choose the primary-elect feature. However, this feature allows you to online the applications on the original Primary outside of VCS control.

When the network connection is restored, you can elect which of the two sites continues as the Primary. In this example the elected Primary is `london` and the elected Secondary is `seattle`.



Any data that was written to the elected Secondary (`seattle`) during the network disruption is lost. Any data that was written to the elected Primary (`london`) is intact.

Configuring VCS global clustering so you can choose the Primary site

To configure VCS global clustering so you can choose the Primary site

1. Make the configuration writable and set the value of the `AutoResync` attribute to `2` for the RVGPrimary resource.

```
# haconf -makerw
# hares -modify RVGPrimary_resource AutoResync 2
```

2. Make sure that the `BunkerSyncTimeout` attribute of the RVGPrimary resource is set to `0` to disable the automated bunker replay feature. You cannot use the automated bunker replay feature and the primary-elect feature in the same environment.
3. Check the value and save the configuration.

```
# hares -value RVGPrimary_resource AutoResync
# haconf -dump -makero
```

Choosing the Primary site after a site disaster or network disruption

After a site disaster or network disruption is resolved, you have the following options:

- Choose the original Primary site as the Primary site going forward.
- Choose the original Secondary site as the Primary site going forward.

This section shows the procedure for each option. In these examples,`seattle` is the Primary site and replicates data to the Secondary site `london`. When `seattle` goes down (or there is a network disruption), `london` takes over as the new Primary. After the problem is resolved and `seattle` is back up, you can elect which site (`seattle` or `london`) is the Primary going forward.

Note: Before you try these procedures, make sure that the AutoResync attribute value is set to 2 to enable the primary-elect feature.

Example – Choosing the original Primary as the Primary going forward

In the following example, the original Primary site, `seattle`, will be chosen as the Primary site going forward. `london` is the elected Secondary site.

To choose the original Primary site as the Primary site going forward

1. On the original Primary, bring the applications in the application service group online so that applications are active on both sites during the primary-elect phase. Bringing the applications online is performed outside of VCS control.
2. On the elected Secondary site, offline the applications in the application service group (*app_grp*). In this example, `london` is the new Secondary site.

   ```
   hagrp -offline app_grp -sys london
   ```

   If you have multiple application service groups, you must repeat this step for each one.
3. On the elected Secondary, specify the original Primary site as the elected Primary site with the `-actionargs` argument of the `hares -action` command. In this example, `seattle` is the name of the global cluster that is associated with the elected Primary site, `seattle`.

   ```
   hares -action RVGPrimary_resource \
   ElectPrimary -actionargs seattle -sys london
   ```

4. Bring the service groups online that you took offline earlier.

Note: If the problem was caused by a network disruption, any application data that was written to `london` during the disruption is lost.

`seattle` is now the Primary site and `london` is the Secondary site. If necessary, bring the applications on `seattle` back online. `seattle` now replicates data to `london`.

Example – Choosing the original Secondary as the Primary going forward

In the following example, the original Secondary site, `london`, will be chosen as the Primary site going forward. `seattle` will be the elected Secondary site.

To choose the original Secondary site as the Primary site going forward

1. On the original Primary, bring the applications in the application service group online so that applications are active on both sites during the primary-elect phase. Bringing the applications online is performed outside of VCS control.
2. On the elected Secondary site, offline the applications in the application service group (*app_grp*). In this example, `seattle` is the new Secondary site.

   ```
   hagrp -offline app_grp -sys seattle
   ```

   If you have multiple application service groups, you must repeat this step for each one.
3. On the elected Secondary, specify the original Secondary site as the elected Primary site with the `-actionargs` argument of the `hares -action` command. In this example, `london` is the name of the global cluster that is associated with the elected Primary site, `london`.

   ```
   hares -action RVGPrimary_resource \
   ElectPrimary -actionargs london -sys london
   ```

4. Bring the service groups online that you took offline earlier.

Note: Any updates made on the elected Secondary during the primary-elect phase will be lost.

`london` is now the Primary site and `seattle` is the Secondary site. If necessary, bring the applications on `london` back online.`london` now replicates data to `seattle`.

Troubleshooting the primary-elect feature

You can troubleshoot the RVGPrimary agent's online agent function, or the VVR ElectPrimary command.

See “Troubleshooting failures in the RVGPrimary online agent function” on page 106.

See “ Troubleshooting failures in VVR ElectPrimary command” on page 107.

Troubleshooting failures in the RVGPrimary online agent function

Use the following information to troubleshoot the online agent function for the primary-elect feature:

- Did not prepare the data volumes for space-optimized snapshot operations.

Check the reason for the failure in the VCS engine log. Manually prepare all the data volumes inside the RVG using the `vxsnap -g dg prepare vol` command. Clear the application service group and bring it back online manually.

- Did not create space-optimized snapshots due to lack of space in the disk group, or any other reason.
  Check the reason for the failure in the VCS engine log. Ascertain and fix the cause for the failure. For example, if you do not have enough disk storage, provision more space in the disk group. Clear the application service group and bring it back online manually. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.
- Did not set the value of the autogrow option to on for the cache objects.
  Check the reason for the failure in the VCS engine log. Manually set 'autogrow' to 'on' for all the cache objects in the RVG, use the `vxcache -g dg set autogrow=on cacheobject` command. Clear the application service group and bring it back online manually.
- Did not convert the secondary RVG to a primary using the `vxrvg -E -F -r makeprimary rvg` command.
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Clear the application service group and bring it back online manually. If the cause of the failure cannot be determined, note the error message and the error code of the failing command from VCS engine logs. Contact Veritas technical support for assistance.

Note: The `-E` option of the vxrvg command is internal and is not intended for customer use.

Troubleshooting failures in VVR ElectPrimary command

Use the following information to troubleshoot failures in VVR ElectPrimary command:

- Did not offline the RVG resource
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Try to manually offline the RVG using the `/opt/VRTS/bin/hares -offline RVG_resource -sys system` command. Re-run the ElectPrimary command.
- Failure for the RVG resource to go offline in one minute
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Ensure that the RVG resource goes offline using the `/opt/VRTS/bin/hares -wait RVG_resource State OFFLINE -sys system`

command. Re-run the ElectPrimary command. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.

- Did not restore the data volumes of the RVG from the space-optimized snapshots
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Re-run the ElectPrimary command. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.
- Did not destroy the space-optimized snapshots
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Manually run the `vxrvg -g dg -P snap_prefix snapdestroy rvg` command to destroy the space-optimized snapshots. The primary election is complete and you do not need to run the ElectPrimary command again. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.

Requirements for configuring VVR in a VCS environment

The requirements for configuring VVR in a VCS environment are as follows:

- Follow the best practices for setting up replication with VVR.
- Each node that is part of a particular service group involved in replication must use the same port number for replication. You may need to change this number on some nodes before configuring VVR.
- If a node has more than one network interface card on the same physical network being used for replication, each network interface card must have a different MAC address. This is true for all the nodes at the Primary and Secondary sites.

Best practices for setting up VVR agents

The following list gives the best practices for setting up the agents:

- Only one DiskGroup and one RVG resource must be present in a service group.
- If a disk group is configured as a DiskGroup resource, then all the RVGs in this disk group must be configured as RVG resources.
  If a disk group is configured as a CVMVolDG resource, then all the RVGs must be configured as RVGShared resources.
- When configuring failover applications, use the RVG and RVGPrimary agents.

- When configuring parallel applications, use the RVGShared and RVGSharedPri agents.
- When configuring parallel applications, use the RVGShared, RVGSharedPri, and RVGLogowner agents. If the configuration has multiple RVGLogowner resources, we recommend that you alternate the order of hosts in the AutoStartList attributes for the service groups containing the RVGLogowner resources. VCS then brings online the RVGLogowner resources on different nodes in the cluster, which facilitates load-balancing. For example, the first service group containing an RVGLogowner resource would appear as:

  ```
  AutoStartList = { seattle1, seattle2 }
  ```

  whereas the next service group would have:

  `AutoStartList = { seattle2, seattle1 }` and so on.
- Do not configure the RVGShared resource in the cvm group. Configure the RVGShared resource in a separate group which contains the RVGShared resource and the CVMVolDg resource.
- In a global clustering environment, you must also use the ClusterList attribute.

  ```
  ClusterList = { gco_primclus=1, gco_secclus=2 }
  ```
- If a volume set is fully associated to an RVG, that is, if all its component volumes are associated to the RVG, you can add the volume set to the agent configuration in the same way that a volume is added. Specify the volume set in the Mount resource instead of the component volume names.

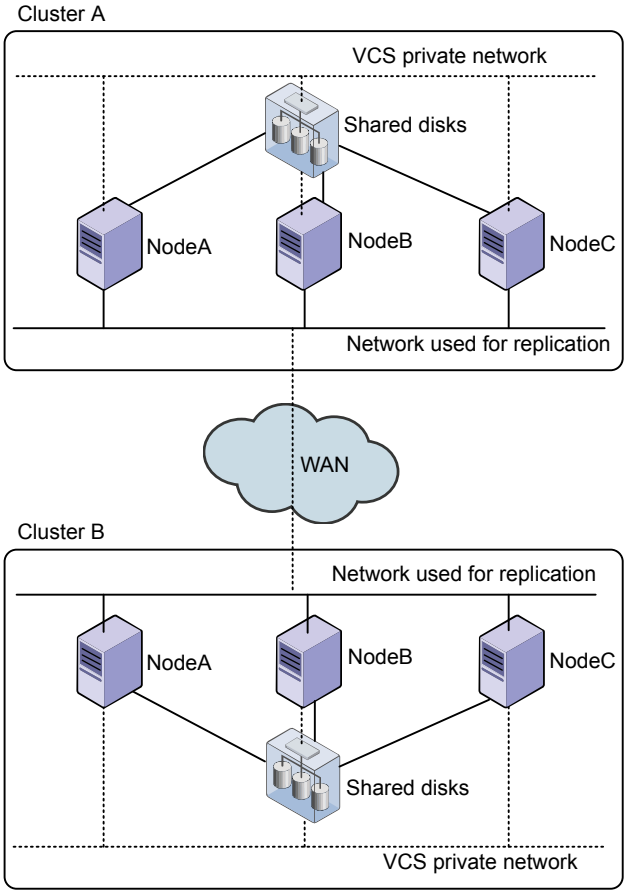
  See "Example setting up VVR in a VCS environment" on page 114.

Note: The agents do not support mounting a volume set that is partially associated to an RVG, that is, if one or more of its component volumes are not associated to the RVG.

See "Volume sets in VVR" on page 51.

Generic VVR setup in a VCS environment

The following illustration shows how VVR replicates in a VCS environment given a two-cluster environment.

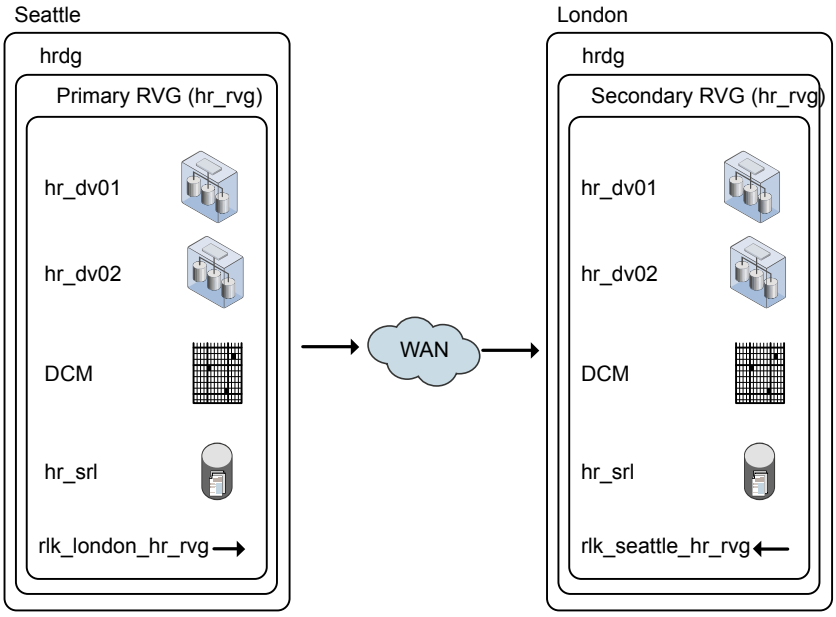


Example VVR configuration in a VCS environment

In the following example, two clusters reside at separate sites. VVR uses a WAN to replicate data between the sites.

The first cluster (Seattle) is in Seattle. The Seattle cluster consists of two nodes: seattle1 and seattle2. The second cluster is in London and is named London.The London cluster also consists of two nodes: london1 and london2. The nodes located in the cluster Seattle contain the Primary RVG. The nodes located in the cluster London contain the Secondary RVG. Note that the following illustration shows the names of the VVR components used by the RVG agent.

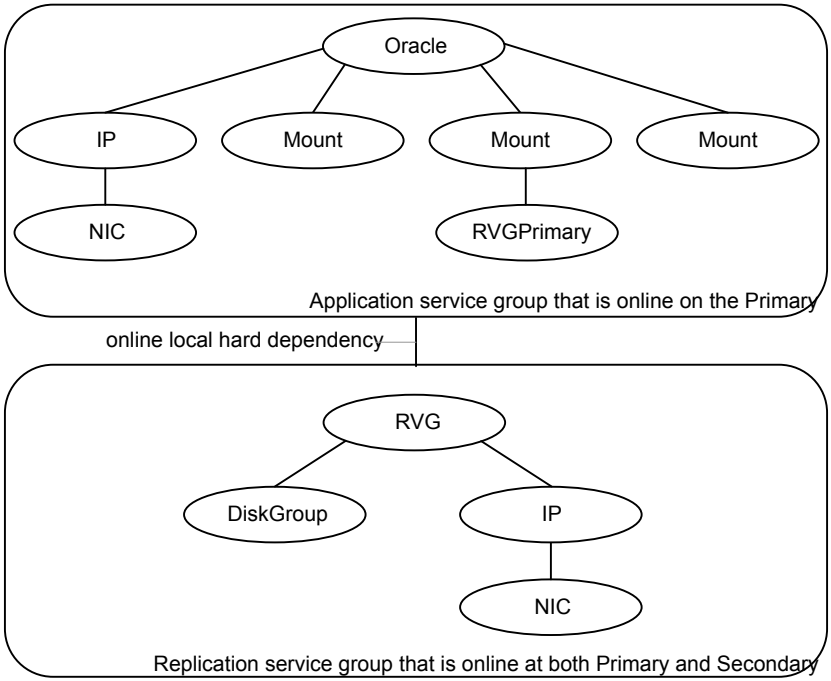
Figure 5-1 Example—VVR configuration in a VCS environment



Example RVG configuration for a failover application

In the following example, a failover application that uses an RVG is made highly available across two clusters. The application service group contains the following resources: application, Mount, NIC, IP, and RVGPrimary. The replication group contains the RVG, IP, NIC, and DiskGroup resources. The application group has an online local hard dependency on the replication group.

Figure 5-2 RVG and RVGPrimary agents—service groups and resource dependencies

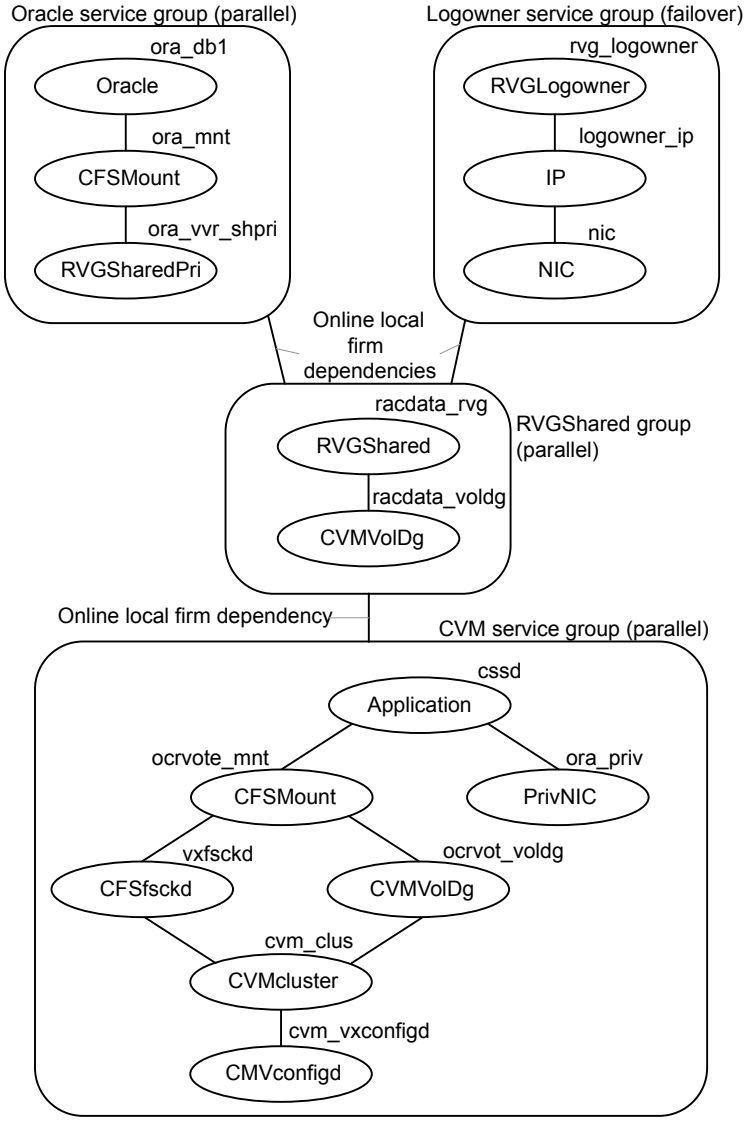


Example RVG configuration for a parallel application

In the following example, a parallel application that uses an RVG is made highly available across two clusters. The Oracle service group is the application group and contains the CFSMount resource. The Logowner service group is a failover group, which manages the logowner. The service group RVGShared manages the RVG used by the application. The Oracle and CVM groups are configured as parallel groups.

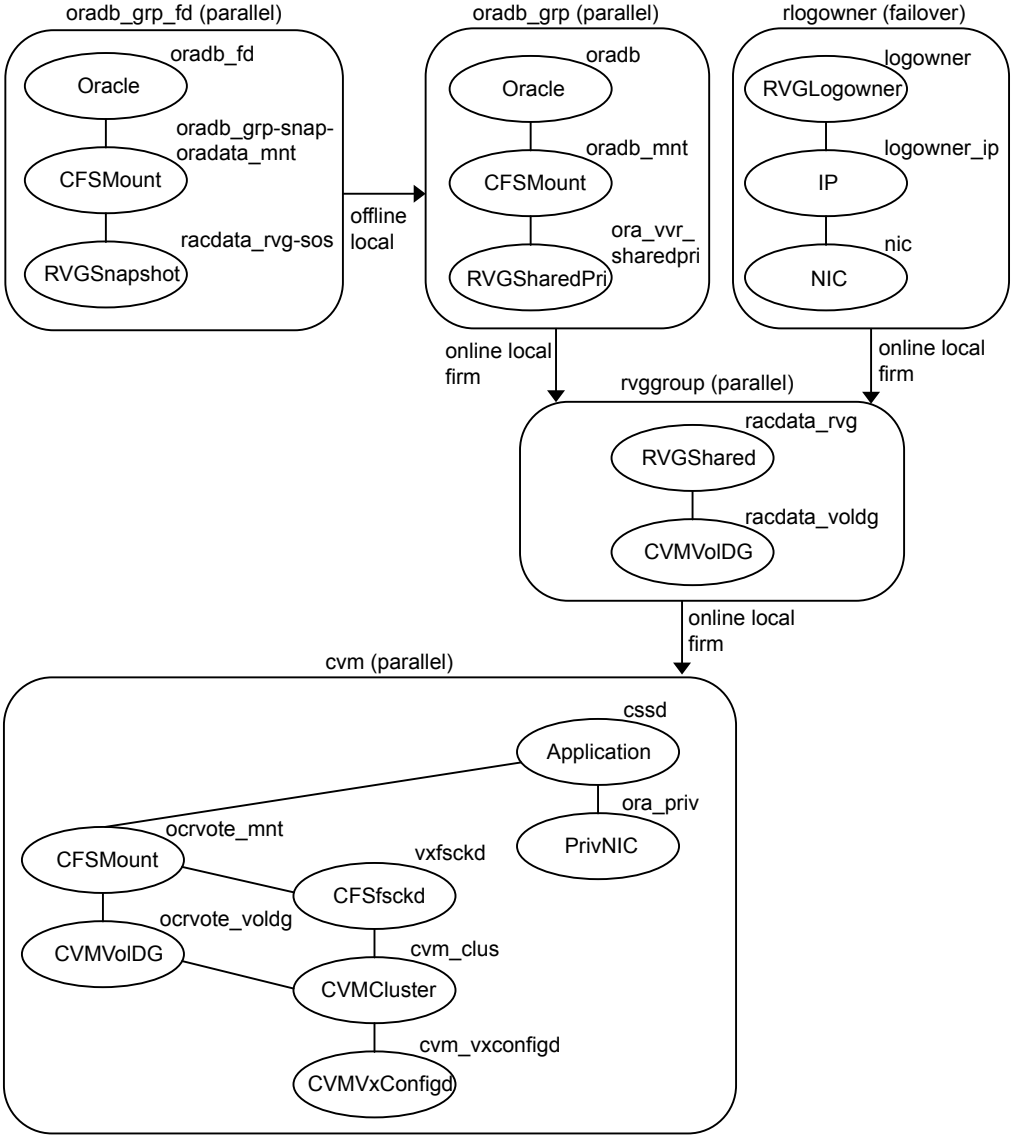
The service groups Logowner and Oracle are dependent on the service group RVGShared. The RVGShared manages the RVG in a shared environment; therefore, it is dependent on the cvm service group.

Figure 5-3 RVGShared, RVGLogowner, and RVGSharedPri agents—Service Groups and Resource Dependencies



The following diagram shows the service group dependencies for a sample FireDrill service group.

Figure 5-4 Example configuration of FireDrill for parallel application



Example setting up VVR in a VCS environment

Configuring VVR with VCS requires the completion of several tasks, each of which must be performed in the following order:

- See "Setting up the VVR configuration" on page 115.
- See "Verifying the VVR replication state" on page 118.
- See "Configuring the VVR agents for failover applications" on page 118.
- See "Configuring the VVR agents for parallel applications" on page 125.

Before setting up the VVR configuration, verify whether all the nodes in the cluster that have VVR installed use the same port number for replication. To verify and change the port numbers, use the vrport command. If the port number is the same on all nodes, add the VVR agents to the VCS configuration.

See "Displaying and changing the ports used by VVR" on page 237.

Setting up the VVR configuration

This section shows how to set up a sample VVR configuration. The VVR configuration in this example applies to the RVG Agent.

It uses the names that are used in the sample configuration file of the RVG agent. The procedure to configure VVR is the same for all the VVR agents. Use the sample configuration files located in /etc/VRTSvcs/conf/sample_vvr directory to configure the other agents.

The example uses the names listed in the following table.

Name of Cluster: Seattle

Disk group	hrdg
Primary RVG	hr_rvg
Primary RLINK to london1	rlk_london_hr_rvg
Primary data volume #1	hr_dv01
Primary data volume #2	hr_dv02
Primary volume set (with data volumes hr_dv03, hr_dv04)	hr_vset01
Primary SRL for hr_rvg	hr_srl
Cluster IP address	10.216.144.160

Name of Cluster: London

Disk group	hrdg

Secondary RVG	hr_rvg
Secondary RLINK to seattle	rlk_seattle_hr_rvg
Secondary data volume #1	hr_dv01
Secondary data volume #2	hr_dv02
Secondary volume set (with data volumes hr_dv03, hr_dv04)	hr_vset01
Secondary SRL for hr_rvg	hr_srl
Cluster IP address	10.216.144.162

In this example, each of the hosts (seattle1 and london1) has a disk group named hrdg with enough free space to create the VVR objects.

Set up the VVR configuration on seattle1 and london1 to include the objects used in the sample configuration files, main.cf.seattle and main.cf.london, located in the /etc/VRTSvcs/conf/sample_vvr/RVG directory.

See "Example VVR configuration in a VCS environment" on page 110.

To set up the VVR configuration

1 On london1:

- Create the Secondary data volumes.

```
# vxassist -g hrdg make hr_dv01 100M \
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv02 100M \
    layout=mirror logtype=dcm mirror=2
```

- Create the data volumes for the volume set on the Secondary and create the volume set.

```
# vxassist -g hrdg make hr_dv03 100M \
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv04 100M \
    layout=mirror logtype=dcm mirror=2
# vxmake -g hrdg vset hr_vset01 \
    appvols=hr_dv03,hr_dv04
```

- Create the Secondary SRL.

```
# vxassist -g hrdg make hr_srl 200M mirror=2
```

2 On seattle1:

- Create the Primary data volumes.

```
# vxassist -g hrdg make hr_dv01 100M \
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv02 100M \
    layout=mirror logtype=dcm mirror=2
```

- Create the data volumes for the volume set on the Primary and create the volume set.

```
# vxassist -g hrdg make hr_dv03 100M \
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv04 100M \
    layout=mirror logtype=dcm mirror=2
# vxmake -g hrdg vset hr_vset01 \
    appvols=hr_dv03,hr_dv04
```

- Create the Primary SRL.

```
# vxassist -g hrdg make hr_srl 200M mirror=2
```

- Create the Primary RVG.

```
# vradmin -g hrdg createpri hr_rvg \
    hr_dv01,hr_dv02,hr_vset01 hr_srl
```

- Uncomment the plus symbol from /etc/vx/vras/.rdg file from the nodes in the secondary site, or configure security accordingly.
- Determine the virtual IP address to be used for replication. Get the IP up using the OS-specific command. This IP address that is to be used for replication must be configured as the IP resource for this RVG service group.
- Create the Secondary RVG.

```
# vradmin -g hrdg addsec hr_rvg 10.216.144.160 \
    10.216.144.162 prlink=rlk_london_hr_rvg \
    srlink=rlk_seattle_hr_rvg
```

Note: The RLINKs must point to the virtual IP address for failovers to succeed. The virtual IP address 10.216.144.160 must be able to ping virtual IP address 10.216.144.162 and vice versa. IPv6 addresses are supported.

- Start replication.

```
# vradmin -g hrdg -f startrep hr_rvg
```

3 Create the following directories on seattle1 and seattle2. These directories will be used as mount points for volumes hr_dv01 and hr_dv02 and the volume set hr_vset01 on the seattle site.

```
# mkdir /hr_mount01
# mkdir /hr_mount02
# mkdir /hr_mount03
```

4 On seattle1, create file systems on the volumes hr_dv01 and hr_dv02 and on the volume set hr_vset01.

Verifying the VVR replication state

Test the replication state between seattle1 and london1 to verify that VVR is configured correctly.

To verify the replication state

1 Type the following command on each node:

```
# vxprint -g hrdg hr_rvg
```

2 In the output, verify the following:

- State of the RVG is `ENABLED/ACTIVE`.
- State of the RLINK is `CONNECT/ACTIVE`.

Configuring the VVR agents for failover applications

This section explains how to configure the VVR agents for failover applications.

See "Configuring the VVR agents for parallel applications" on page 125.

Configure the RVG agent and RVGPrimary agent either when VCS is stopped or running.

Sample configuration files, main.cf.seattle and main.cf.london, are located in the /etc/VRTSvcs/conf/sample_vvr/RVG and /etc/VRTSvcs/conf/sample_vvr/RVGPrimary directories respectively, and can be used for reference.

Use one of the following procedures to add the RVG resource to the VCS configuration:

- See "Configuring the agents when VCS is running" on page 119.
- See "Configuring the agents when VCS is stopped" on page 124.

Configuring the agents when VCS is running

The example in this section explains how to configure the RVG and RVGPrimary agents when VCS is running.

See "Example RVG configuration for a failover application" on page 111.

Note: Use this example as a reference when creating or changing your resources and attributes.

To add the agent resources to your existing VCS configuration when VCS is running, perform the following procedures:

- Create the replication service group
- Create the application service group
- Create an online local hard dependency between the application service group and the replication service group
- Configure the application service group for global failover

Perform the following steps on the client system seattle1 in the Primary cluster Seattle, and then repeat the steps (with minor changes as noted) on the client system london1 in Secondary cluster London:

To create the replication service group

1. Log in as root.
2. Set the VCS configuration mode to read/write by issuing the following command:

```
# haconf -makerw
```

3 Add the replication service group, `VVRGrp`, to the cluster. This group will contain all the storage and replication resources. Modify the attributes `SystemList` and `AutoStartList` of the service group to populate `SystemList` and `AutoStartList`:

```
# hagrp -add VVRGrp
# hagrp -modify VVRGrp SystemList seattle1 0 seattle2 1
# hagrp -modify VVRGrp AutoStartList seattle1 seattle2
```

On the Secondary cluster, replace seattle1 and seattle2 with london1 and london2

4 Add the DiskGroup resource `Hr_Dg` to the service group `VVRGrp` and modify the attributes of the resource:

```
# hares -add Hr_Dg DiskGroup VVRGrp
# hares -modify Hr_Dg DiskGroup hrdg
```

5 Add a NIC resource `vvrnic` to the service group `VVRGrp` and modify the attributes of the resource:

```
# hares -add vvrnic NIC VVRGrp
# hares -modify vvrnic Device eth3
```

6 Add the IP resource `vvrip` to the service group `VVRGrp` and modify the attributes of the resource:

```
# hares -add vvrip IP VVRGrp
# hares -modify vvrip Device eth3
# hares -modify vvrip Address 192.168.40.20
# hares -modify vvrip NetMask "255.255.248.0"
```

On the Secondary cluster, use the appropriate IP for the Address. For example:

```
# hares -modify vvrip Address 192.168.40.21
```

7 Add the RVG resource Hr_Rvg in the VVRGrp and modify the attributes of the resource.

```
# hares -add Hr_Rvg RVG VVRGrp
# hares -modify Hr_Rvg RVG hr_rvg
# hares -modify Hr_Rvg DiskGroup hrdg
```

8 Specify resource dependencies for the resources you added in the previous steps:

```
# hares -link Hr_Rvg vvrip
# hares -link Hr_Rvg Hr_Dg
# hares -link vvrip vvrnic
```

9 Enable all resources in `VVRGrp`

```
# hagrp -enableresources VVRGrp
```

10 Save and close the VCS configuration

```
# haconf -dump -makero
```

Perform the following steps on the system seattle1 in the Primary cluster Seattle, and then repeat the steps (with minor changes as noted) on the system london1 in Secondary cluster London:

To create the application service group

1 Log in as root.

2 Set the VCS configuration mode to read/write by issuing the following command:

```
# haconf -makerw
```

3 Add a service group, `ORAGrp`, to the cluster `Seattle`. This group will contain all the application specific resources. Populate the attributes `SystemList`, `AutoStartList` and `ClusterList` of the service group.

```
# hagrp -add ORAGrp
# hagrp -modify ORAGrp SystemList seattle1 0 seattle2 1
# hagrp -modify ORAGrp AutoStartList seattle1 seattle2
# hagrp -modify ORAGrp ClusterList  Seattle 0 London 1
```

On the Secondary , replace seattle1 and seattle2 with london1 and london2, as follows:

```
# hagrp -add ORAGrp
# hagrp -modify ORAGrp SystemList london1 0 london2 1
# hagrp -modify ORAGrp AutoStartList london1 london2
# hagrp -modify ORAGrp ClusterList  Seattle 0 London 1
```

4 Add a NIC resource `oranic` to the service group `ORAGrp` and modify the attributes of the resource:

```
# hares -add oranic NIC ORAGrp
# hares -modify oranic Device eth0
```

5 Add an IP resource `oraip` to the service group `ORAGrp` and modify the attributes of the resource:

```
# hares -add oraip IP ORAGrp
# hares -modify oraip Device eth0
# hares -modify oraip Address 192.168.40.1
# hares -modify oraip NetMask "255.255.248.0"
```

On the Secondary, modify the Address attribute for the IP resource appropriately.

6 Add the Mount resource `Hr_Mount01` to mount the volume `hr_dv01` in the RVG resource `Hr_Rvg`:

```
# hares -add Hr_Mount01 Mount ORAGrp
# hares -modify Hr_Mount01 MountPoint /hr_mount01
# hares -modify Hr_Mount01 BlockDevice /dev/vx/dsk/Hr_Dg/hr_dv01
# hares -modify Hr_Mount01 FSType vxfs
# hares -modify Hr_Mount01 FsckOpt %-n
```

7 Add the Mount resource `Hr_Mount02` to mount the volume `hr_dv02` in the RVG resource `Hr_Rvg`:

```
# hares -add Hr_Mount02 Mount ORAGrp
# hares -modify Hr_Mount02 MountPoint /hr_mount02
# hares -modify Hr_Mount02 BlockDevice /dev/vx/dsk/Hr_Dg/hr_dv02
# hares -modify Hr_Mount02 FSType vxfs
# hares -modify Hr_Mount02 FsckOpt %-n
```

8 Add the Mount resource Hr_Mount03 to mount the volume set hr_vset01 in the RVG resource Hr_Rvg:

```
# hares -add Hr_Mount03 Mount ORAGrp
# hares -modify Hr_Mount03 MountPoint /hr_mount03
# hares -modify Hr_Mount03 BlockDevice /dev/vx/dsk/ Hr_Dg/hr_vset01
# hares -modify Hr_Mount03 FSType vxfs
# hares -modify Hr_Mount03 FsckOpt %-n
```

9 Add the Oracle resource `Hr_Oracle`

```
# hares -add Hr_Oracle Oracle ORAGrp
# hares -modify Hr_Oracle Sid hr1
# hares -modify Hr_Oracle Owner oracle
# hares -modify Hr_Oracle Home "/hr_mount01/OraHome1"
# hares -modify Hr_Oracle Pfile "inithr1.ora"
# hares -modify Hr_Oracle User dbtest
# hares -modify Hr_Oracle Pword dbtest
# hares -modify Hr_Oracle Table oratest
# hares -modify Hr_Oracle MonScript "./bin/Oracle/SqlTest.pl"
# hares -modify Hr_Oracle StartUpOpt STARTUP
# hares -modify Hr_Oracle ShutDownOpt IMMEDIATE
# hares -modify Hr_Oracle AutoEndBkup 1
```

10 Add the Oracle listener resource `LISTENER`

```
# hares -add LISTENER Netlsnr ORAGrp
# hares -modify LISTENER Owner oracle
# hares -modify LISTENER Home "/hr_mount01/OraHome1"
# hares -modify LISTENER Listener LISTENER
# hares -modify LISTENER EnvFile "/oracle/.profile"
# hares -modify LISTENER MonScript "./bin/Netlsnr/LsnrTest.pl"
```

11 Add the RVGPrimary resource `Hr_RvgPri`

```
# hares -add Hr_RvgPri RVGPrimary ORAGrp
# hares -modify Hr_RvgPri RvgResourceName Hr_Rvg
```

12 Specify resource dependencies for the resources you added in the previous steps:

```
# hares -link LISTENER Hr_Oracle
# hares -link LISTENER oraip
# hares -link Hr_Oracle Hr_Mount01
# hares -link Hr_Oracle Hr_Mount02
# hares -link Hr_Mount01 rvg-pri
# hares -link Hr_Mount02 rvg-pri
# hares -link Hr_Mount03 rvg-pri
# hares -link oraip oranic
```

13 The application service group and the replication service group must both exist before doing this step. If you have not yet created the replication service group, do so now.

See "Configuring the agents when VCS is running" on page 119.

After you have created the application service group and the replication service group, specify an online local hard group dependency between `ORAGrp` and `VVRGrp`.

```
# hagrp -link ORAGrp VVRGrp online local hard
```

14 Enable all resources in `ORAGrp`

```
# hagrp -enableresources ORAGrp
```

15 Save and close the VCS configuration

```
# haconf -dump -makero
```

16 Bring the service groups online, if not already online.

```
# hagrp -online VVRGrp -sys seattle1
# hagrp -online ORAGrp -sys seattle1
```

17 Verify that the service group `ORAGrp` is ONLINE on the system `seattle1` by issuing the following command:

```
# hagrp -state ORAGrp
```

Refer to the /var/VRTSvcs/engine.log file to identify configuration errors and to take actions accordingly.

Configuring the agents when VCS is stopped

Perform the following steps to configure the RVG agent using the sample configuration file on the first node in the Primary cluster and Secondary cluster. In the example in this guide, seattle1 is the first Primary node and london1 is the first Secondary node.

To configure the agents when VCS is stopped

1. Log in as root.
2. Ensure that all changes to the existing configuration have been saved and that further changes are prevented while you modify main.cf:

   If the VCS cluster is currently writeable, run the following command:

   ```
   # haconf -dump -makero
   ```

   If the VCS cluster is already read only, run the following command:

   ```
   # haconf -dump
   ```

3. Do not edit the configuration files while VCS is started. The following command will stop the had daemon on all systems and leave resources available:

   ```
   # hastop -all -force
   ```

4. Make a backup copy of the main.cf file:

   ```
   # cd /etc/VRTSvcs/conf/config
   # cp main.cf main.cf.orig
   ```

5. Edit the main.cf files for the Primary and Secondary clusters. The files main.cf.seattle and main.cf.london located in the /etc/VRTSvcs/conf/sample_vvr/RVGPrimary directory can be used for reference for the primary cluster and the secondary cluster respectively.
6. Save and close the file.
7. Verify the syntax of the file /etc/VRTSvcs/conf/config/main.cf:

   ```
   # hacf -verify /etc/VRTSvcs/conf/config
   ```

8. Start VCS on all systems in both clusters.
9. Administer the service groups.

   See "Administering VCS service groups" on page 132.

Configuring the VVR agents for parallel applications

Use the RVGShared, RVGSharedPri, and the RVGLogowner agents to manage and monitor RVGs used by parallel applications in a shared environment.

Note: Determine the node that is performing the most writes by running the vxstat command on each node for a suitable period of time; after you set up replication, specify this node as the logowner.

The prerequisites for configuring the agents are as follows:

- You must have replication set up between the Primary and Secondary sites.
  See "Replication in a shared disk group environment" on page 41.
- The sites must be configured in a global cluster and the application service must be configured as a global service group.
  For more information about configuring global clusters, see the *Cluster Server Administrator's Guide*.

Sample configuration files are located in the /etc/VRTSvcs/conf/sample_rac/ directory and include CVR in the filename. These sample files are installed as part of the VRTSdbac package, and can be used as a guide when creating your configuration. You can configure agents from the command line or from the VCS Java and Web consoles.

See the *Cluster Server Administrator's Guide* for more information.

To modify the VCS configuration on the Primary cluster

1. Define two new service groups: A logowner group that includes the RVGLogowner resource, and an RVG group that includes the RVGShared resource replication objects.
2. In the logowner group, define IP and NIC resources, used by the RLINKs for the RVG, and the RVGLogowner resource, for which the RVG and its associated disk group are defined as attributes.
3. In the RVG service group, set up the RVGShared agent to monitor the RVG resource. Because it is shared, the RVG must be configured to depend on the CVMVolDg resource, which defines the shared disk group and its activation mode.

   Define the RVGShared and CVMVolDg resources within a parallel service group so that the service group may be online at the same time on all cluster nodes.
4. Add the RVGSharedPri resource to the existing application service group and define the service group to be a global group.

   See the *Cluster Server Administrator's Guide* for instructions on how to create global groups.

5 Move the CVMVolDg resource from the existing application service group to the newly created RVGShared service group.

6 Set the following service group dependencies:

- The RVG logowner service group has an “online local firm” dependency on the service group containing the RVG.
- The RVG service group has an “online local firm” dependency on the CVM service group.
- The application service group has an “online local firm” dependency on the RVG service group.

To modify the VCS configuration on the Secondary cluster

1 Log on to a node in the secondary cluster as root.

2 Ensure that all changes to the existing configuration have been saved and that further changes are prevented while you modify main.cf:

If the VCS cluster is currently writeable, run the following command:

```
# haconf -dump -makero
```

If the VCS cluster is already read only, run the following command:

```
# haconf -dump
```

3 Ensure VCS is not running while you edit main.cf by using the `hastop` command to stop the VCS engine on all systems and leave the resources available:

```
# hastop -all -force
```

4 Make a backup copy of the main.cf file:

```
# cd /etc/VRTSvcs/conf/config
# cp main.cf main.orig
```

5 Use vi or another text editor to edit the main.cf file, making the following changes:

- Edit the CVM group on the secondary cluster. Use the CVM group on the primary as your guide.
- Add the logowner group and the RVG service groups.
- Add an application service group. Use the application service group on the primary cluster as a pattern for the service group on the secondary cluster.

- Since the service group is a global group, assign it the same name as the group on the primary cluster.
- Define the ClusterList and ClusterFailOverPolicy cluster attributes.
- Include the RVGSharedPri resource.

6 Save and close the main.cf file.

7 Verify the syntax of the file /etc/VRTSvcs/conf/config/main.cf:

```
# hacf -verify /etc/VRTSvcs/conf/config
```

8 Start VCS on all systems in both clusters.

The application group should be online on both systems of the primary cluster.

The application service group should not be online on the secondary cluster, but the CVM, RVG logowner, and RVG groups should be online.

Configuring the agents for a bunker replication configuration

This section describes how to set up the VCS agents for a bunker replication configuration, that is, an RDS that includes a bunker site. A bunker can be set up using the STORAGE protocol, or using IP.

See "Introduction to replication using a bunker site" on page 334.

Refer to one of the following sections to configure the VCS agents:

- See "VCS configuration for a bunker using the STORAGE protocol" on page 128.
- See "VCS configuration for a bunker using IP" on page 130.

VCS configuration for a bunker using the STORAGE protocol

When a bunker is set up using the STORAGE protocol, the disk group containing the bunker RVG is imported on the Primary node. If the Primary RVG is in a VCS cluster, the bunker RVG must remain online on the same node on which the Primary RVG is online.

Note: Bunker replication is not supported for shared disk group environments.

This section describes how to configure the agents to automate the failover of the bunker RVG.

In a private disk group environment, the RVG resource handles the failover process. If the node on which the RVG resource is online fails, the RVG resource fails over to another node within the cluster. The RVG resource ensures that the bunker RVG also fails over, so that the bunker RVG continues to be on the same node with the Primary RVG.

Table 5-1 Attributes for configuring bunker failover

Attribute	Description
StorageDG	The name of the bunker disk group.
StorageRVG	The name of the bunker RVG.
StorageHostIds	Hostid of the bunker node or, if the bunker is clustered, a space-separated list of the hostids of each node in the bunker cluster.

The bunker failover attributes described in this section are the only specific attributes that differ for an RDS containing a bunker. The rest of the configuration for the VCSAgent is the same as for any other RDS.

See "Example setting up VVR in a VCS environment" on page 114.

Sample configuration file for VCS agents in a bunker replication environment

The following example shows a sample configuration file when the bunker Secondary is connected to the Primary using the STORAGE protocol.

This example uses the following names:

- seattle: primary cluster node
- london: bunker node
- bdg : bunker disk group name
- brvg: bunker RVG name

Sample configuration file (failover application)

The following sample file shows the configuration for the VCS agent on the Primary. The RVG agent includes attributes for a STORAGE bunker, to enable the bunker disk group to failover together with the parent RVG.

If the Secondary for the RDS has a bunker associated to it, the RVG agent on the Secondary similarly would include the StorageRVG, StorageDG, and StorageHostIds attributes.

```
group AppSG (
        ClusterList = { cluster_london = 0 }
        SystemList = { seattle = 0, london = 1 }
        Authority = 1
        AutoStartList = { seattle }
        ClusterFailOverPolicy = Manual
        )
        RVG RVG-1 (
     RVG = vcsrvg
     DiskGroup = pdg
     Primary = true
     StorageRVG = brvg
     StorageDG = bdg
     StorageHostIds = "portland"
                    )
...
```

VCS configuration for a bunker using IP

The configuration for the VCS agents for a bunker over IP is the same as for any other Secondary.

To set up a bunker configuration

1. The Primary and Secondary configurations are the same as for any other RDS using VCS agents.

   See "Example setting up VVR in a VCS environment" on page 114.

2. Add the bunker to the RDS with the `vradmin addbunker` command.

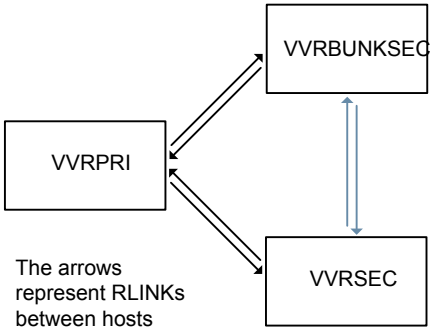
   See "Adding a bunker to an RDS" on page 340.

3. Configure the VCS agents on the secondary in the same way as the configuration for any other Secondary. There is no special configuration that needs to be done for a bunker over IP.

How the RVGPrimary works in a bunker setup

Under normal operating conditions, the Primary cluster (VVRPRI) site replicates data to the bunker's Secondary (VVRSEC) site in the synchronous override mode. It replicates the data in the synchronous mode to ensure that it is up-to-date.

Figure 5-5 RLINKs among Primary, Secondary, and bunker hosts, where the arrows represent RLINKs between hosts



If a disaster occurs at the VVRPRI, the RVGPrimary agent on VVRSEC activates the Bunker node and starts replay from the Bunker Replicator Log to VVRSEC. During this replay, the agent converts the Bunker node to a Primary and the agent uses the data in the Replicator Log to bring the Secondary up-to-date.

When the replay completes or the BunkerSyncTimeOut attribute's timeout limit has elapsed, the Secondary takes over the Primary's role and deactivates the Bunker node.

For a Storage Bunker configuration, if a disaster occurs at the Primary then the RVGPrimary agent comes online on the secondary node VVRSEC. The agent first imports the disk group on the Bunker node and then it activates the Bunker node to start replay to the Secondary.

When the original Primary becomes available, you can migrate the Primary role back to the original site. If you had performed the takeover with auto failback then you have enabled failback logging at takeover.

If the original Primary becomes available again, the agent converts it to the Secondary. The new Primary writes to the original Primary to bring it up-to-date. After a successful migration or takeover, the agent tries to start the replication to the Bunker if any bunker sites are associated with the new Primary. If this operation fails, an VCS logs an error message in its engine log. After you review the error message, you can then start the replication to bunker outside of VCS.

About bunker SRL overflow

The bunker replay does not occur if the bunker SRL has overflowed or the bunker SRL is not in useful state. In the case of an overflowed bunker or one that is in a useless state, the RVGPrimary agent logs an error message while attempting to initiate the bunker replay and then faults the resource. Since the bunker replay cannot happen in this situation, Veritas recommends that you set the

BunkerSyncTimeOut value to 0. A value of 0 means that you do not choose to use the bunker replay and that the online operation can be completed successfully without bunker replay.

About global clustering with bunker replay

When the value of the ClusterFailoverPolicy attribute is Auto and the Application service group is configured on some nodes of the primary cluster, global clustering immediately detects any system fault at the primary site and fails over the Application service group to the remote site. VVR might take longer to detect the fault at the primary site and to complete its configuration changes to reflect the fault. To ensure that global clustering successfully initiates a bunker replay, Veritas recommends that you set the value of the OnlineRetryLimit attribute to a non-zero value for the RVGPrimary resource when the primary site has a bunker configured.

Administering VCS service groups

This section explains how to administer a VCS service group for cluster Seattle from the command line. Note that you can also use the VCS Java and Web consoles to administer service groups.

To administer a VCS service group

1. Start the VCS engine on seattle1:

```
# hastart
```

2. Verify that all the service groups that contain RVG resource type are brought online:

```
# hagrp -display
```

3. Take the service group offline and verify that all resources are stopped:

```
# hagrp -offline hr_grp -sys seattle1
# hagrp -display
```

4. Bring the service group online again and verify that all resources are available:

```
# hagrp -online hr_grp -sys seattle1
# hagrp -display
```

5. Start the VCS engine on seattle2:

```
# hastart
```

6 Switch the VVR service group to seattle2:

```
# hagrp -switch hr_grp -to seattle2
```

7 Verify that all the service groups that contain RVG resource type are brought online on seattle2:

```
# hagrp -display
```

8 Repeat step 1 through step 7 for the cluster London.

9 If required, check the following log files on any system for the status or any errors:

/var/VRTSvcs/log/engine_A.log

/var/VRTSvcs/log/RVG_A.log

Section 2

Setting up and administering VVR

- Chapter 6. Setting up replication
- Chapter 7. Displaying configuration information
- Chapter 8. Administering Volume Replicator
- Chapter 9. Using VVR for off-host processing
- Chapter 10. Transferring the Primary role
- Chapter 11. Replication using a bunker site
- Chapter 12. Troubleshooting VVR
- Chapter 13. Tuning replication performance

Chapter 6

Setting up replication

This chapter includes the following topics:

- About configuring VVR replication
- Creating a Replicated Data Set
- Synchronizing the Secondary and starting replication
- Starting replication when the data volumes are zero initialized

About configuring VVR replication

You can configure and administer Volume Replicator (VVR) using one of the following interfaces:

Command line interface (CLI)	You can use the command line interface of VVR to configure, administer, and monitor VVR in a distributed environment. This guide gives instructions on configuring, administering, and monitoring VVR using the Veritas product installer.

This topic explains how to set up a Replicated Data Set (RDS) using the command-line interface. VVR enables you to set up replication either when the data volumes are zero initialized or contain valid data. Make sure you follow the best practices or recommendations described to ensure successful configuration of VVR.

Detailed examples are also available on how to configure and set up a simple VVR configuration. Read this information before you start setting up replication.

See "Examples for setting up a simple Volume Replicator configuration" on page 511.

Before setting up a Replicated Data Set, decide how you plan to lay out your VVR configuration.

To configure and set up replication, perform the following tasks in the order presented below.

See "Creating a Replicated Data Set" on page 136.

See "Synchronizing the Secondary and starting replication" on page 152.

Note: The procedure to set up replication is the same either when the application is running or stopped, unless noted otherwise.

Creating a Replicated Data Set

To create a Replicated Data Set (RDS), perform the following tasks in the order presented below:

- Create a Primary Replicated Volume Group (RVG) of the RDS
  You can also associate volume-set component volumes to an RDS.
  See "Associating a volume set to an RDS" on page 201.
- Add a Secondary to the RDS
- Change the Replication Settings for the Secondary

In a shared disk group environment, the `vradmin`, `vrstat`, and `vrnotify` commands can be issued on any node in the cluster. However, the `vradmin createpri`, `vxibc`, `vxrlink` (other than informational commands), and `vxrvg` commands must be issued from the CVM master node.

Creating a Primary RVG of an RDS

The first step in creating an RDS is creating its Primary RVG. VVR enables you to create a Primary RVG of an RDS using the `vradmin createpri` command.

The `vradmin createpri` command enables you to associate existing data volumes and the Storage Replicator Log (SRL) to the Primary RVG.

The `vradmin createpri` command performs the following operations:

- Creates the Primary RVG on the host on which the command is issued.
- Enables or starts the Primary RVG.
- Associates DCMs to the data volumes in the RVG.
- Associates the specified data volumes and SRL to the RVG.

- Associates the specified volume sets (if any) to the RVG.

Note: Specify the volume set name in the command, not the names of each component volume. Specifying the component volume name causes the command to fail.

VVR does not support RAID-5 volumes, that is, volumes with usage type `raid5` are not supported. Data volumes must be of usage type `gen` or `fsgen`. However, data volumes can be configured on hardware-based RAID-5 disks.

Dirty Region Logs (DRLs) are not needed with VVR because VVR uses the SRL to recover volumes, not the DRLs. If any of the data volumes or the SRL has a DRL, the `vradmin createpri` command removes the DRL before the data volume is associated to the RVG.

By default, the `vradmin createpri` command adds DCMs to the data volumes, if they have not already been added. The `vradmin createpri` command creates the DCM of an appropriate default size based on the size of the volume and mirrors the DCM by default. To create and add a DCM of a size that is different from the default, associate the DCM of the required size to the data volumes before running the `vradmin createpri` command.

Note: The `vradmin createpri` command will fail if there are not enough drives to mirror the DCM. You may need to associate the DCM to an appropriately sized data volume.

See "Associating a Data Change Map to a data volume" on page 205.

The `-nodcm` option when used with the `vradmin createpri` command associates data volumes to the RVG but does not add DCMs to the data volumes.

If you want to associate additional volumes to the RVG after creating the RVG, use the `vradmin addvol` command.

See "Associating a volume to a Replicated Data Set" on page 196.

Prerequisites for creating a Primary RVG of an RDS

Before creating a Primary RVG of an RDS, the following prerequisites must be met:

- The data volumes and SRL must exist on the Primary. If the data volumes and SRL do not exist on the Primary, create them. To associate a volume set to the RVG, the volume set must exist on the Primary.
- The SRL cannot be a volume set or a component volume of a volume set.

- The data volumes and SRL must be started. If the data volumes and SRL are not started, start them. When a data volume is started, its state is active.
- The data volumes used by the application must exist in the same RVG. Include the data volumes used by the application in the same RVG.
- Make sure you include the appropriate loopback address(es) in the `/etc/hosts file`.
  - If your environment only uses IPv4, you must include an IPv4 loopback address in the `/etc/hosts` file. The following is a sample entry:

    ```
    127.0.0.1       localhost       loopback
    ```

  - If your environment only uses IPv6, you must include an IPv6 loopback address in the `/etc/hosts` file.

    ```
    ::1             localhost6.localdomain6      localhost6     loopback
    ```

  - If your environment uses both IPv4 and IPv6, the `/etc/hosts` file must include both loopback addresses.

    ```
    127.0.0.1       localhost       loopback
    ::1             localhost6.localdomain6      localhost6     loopback
    ```

To create a Primary RVG of an RDS

Issue the following command on the host on which you want to create the Primary RVG:

```
# vradmin -g diskgroup createpri rvgname \
      dv01_name,dv02_name... srl_name
```

The argument *rvgname* is the name of the RVG to be created.

The argument *dv01_name,dv02_name,...* is a comma-separated list of the names of the data volumes to be associated to the RVG. Each item can be an independent data volume name, or the name of a volume set. To associate a volume set to the RVG, specify the name of the volume set, not the names of the individual component volumes.

Note: In previous releases, component volumes could be associated directly to an RVG. Beginning in Release 5.0, the volume set itself is associated to the RVG, enabling VVR to verify consistency between the volume sets on the Primary and the Secondary RVGs. The `vradmin createpri` command fails if a component volume of the volume set and the volume set itself are each specified for an RVG.

The argument *srl_name* is the name of the SRL to be associated to the RVG.

Use -nodcm option if you do not want DCMs to be added to the data volumes. By default, DCMs are added automatically.

Example - Creating a Primary RVG containing a data volume

This example shows how to create a Primary RVG hr_rvg in the disk group hrdg, which contains the data volumes hr_dv01 and hr_dv02, and the volume hr_srl that is to be used as the SRL. This example automatically adds DCMs to the data volumes.

```
# vradmin -g hrdg createpri hr_rvg hr_dv01,hr_dv02 hr_srl
```

Example - Creating a Primary RVG containing a volume set

This example shows how to create a Primary RVG hr_rvg in the disk group hrdg, which contains the volume set hr_vset, the data volumes hr_dv01 and hr_dv02, and the volume hr_srl that is to be used as the SRL.

```
# vradmin -g hrdg createpri hr_rvg hr_dv01,hr_dv02,hr_vset \
hr_srl
```

If the volume set includes the component volumes hr_vsetdv01 and hr_vsetdv02, these volumes are associated to the RVG hr_rvg. This example automatically adds DCMs to the data volumes, including the component volumes hr_vsetdv01 and hr_vsetdv02.

Adding a Secondary to an RDS

After creating the Primary RVG of the RDS, go on to adding a Secondary. Use the vradmin addsec command to add a Secondary RVG to an RDS. This command can also be used to add additional Secondary RVGs. The vradmin addsec command can be issued from any host that is already in the RDS.

Note: Run the vradmin addsec from the Primary node. If you run this command from the node being added as the Secondary, the command fails.

The vradmin addsec command performs the following operations by default:

- Creates and adds a Secondary RVG of the same name as the Primary RVG to the specified RDS on the Secondary host. By default, the Secondary RVG is

added to the disk group with the same name as the Primary disk group. Use the option `-sdg` with the `vradmin addsec` command to specify a different disk group on the Secondary.

- If any of the data volumes or the SRL on the Secondary has a DRL, the DRL is removed before the data volume is associated to the RVG. DRLs are not needed with VVR because VVR uses the SRL to recover volumes, not the DRLs.
- Automatically adds DCMs to the Primary and Secondary data volumes if they do not have DCMs. Use the `-nodcm` option to specify that DCMs are not to be added to the data volumes.

  The `vradmin addsec` command creates the DCM of an appropriate default size based on the size of the volume and mirrors the DCM by default. To create and add a DCM of a size that is different from the default, associate the DCM of the required size to the data volumes before running the `vradmin addsec` command.

  See "Associating a Data Change Map to a data volume" on page 205.
- Associates to the Secondary RVG, existing data volumes of the same names and sizes as the Primary data volumes; it also associates an existing volume with the same name as the Primary SRL, as the Secondary SRL.
- If the Primary RVG includes a volume set, the `vradmin addsec` command associates the corresponding volume set to the Secondary, if the volume set exists on the Secondary. The volume set on the Secondary must include volumes of the same name, lengths and indices as the component volumes on the Primary. If the volume set exists on the Secondary and the volume set configuration is correct except that it does not include all of the component volumes corresponding to those in the volume set on the Primary, the `vradmin addsec` command attempts to add the remaining component volumes to the volume set on the Secondary and then associate the volume set to the Secondary RVG. This command succeeds if all of the remaining component volumes exist on the Secondary with the same names, lengths, and indices as the component volumes on the Primary. However, if any of the component volumes do not exist on the Secondary or have a mismatched name, length, or index, the `vradmin addsec` command fails with the appropriate error message.

  If the volume set does not exist on the Secondary, but the component volumes exist with the same names, lengths, and indices, the `vradmin addsec` command creates the volume set on the Secondary and then associates it to the Secondary RVG.
- Creates and associates to the Primary and Secondary RVGs respectively, the Primary and Secondary RLINKs with default RLINK names *`rlk_remotehost_rvgname`*. If you choose to use names other than the default, use the `prlink` and `srlink` attributes of the `vradmin addsec` command to specify the Primary and Secondary RLINK names.

See "Example - Creating a Primary RVG containing a volume set" on page 139.

Note: For replication in asynchronous mode with secondary logging, the SRL size on both the Primary and Secondary must be the same.

Best practices for adding a Secondary to an RDS

When you add a Secondary to an RDS, we recommend the following best practices:

- Determine the network and IP addresses to use. Add all participating system names and IP addresses to the `/etc/hosts` files on each system or to the name server database of your name service. Make sure the IP addresses are available (that is, plumbed and up) on the appropriate hosts for your configuration.
- Plan ahead for application clustering by configuring the IP addresses used for replication as virtual IP addresses. For each replicated data set, the Primary and the Secondary cluster should each have one unique virtual IP address to use as the address for the RLINK. If you do this, you can place VVR under cluster control without having to modify the IP address of the RLINK later. Changing the IP address of an RLINK requires pausing replication.
- Plan the bandwidth of the network based on your requirement. You can choose to use either the UDP protocol or TCP protocol for network communication between the Primary and Secondary. Also, plan to operate in a firewall environment.
  See "Planning the network" on page 72.
- We recommend that you use the following naming conventions for RLINKs. By default, VVR follows the following naming conventions for RLINKs:
  Primary RLINK: *`rlk_remotehost_rvgname`*. For example:
  `rlk_london_hr_rvg`
  Secondary RLINK: *`rlk_remotehost_rvgname`*. For example:
  `rlk_seattle_hr_rvg`
- If you have multiple secondaries in your RDS setup, VVR automatically creates RLINKs between ever pair of secondaries. By doing this, the additional secondaries will be automatically added to the RDS after the migrate operation has completed successfully.
- Associate a DCM to each data volume on the Primary and the Secondary to use the SRL Protection and Failback Logging features.

Prerequisites for adding a Secondary to an RDS

On the Secondary to be added, do the following:

- Create a disk group with the same name as the Primary disk group.
- Create data volumes of the same names and lengths as the Primary data volumes.
- Create an SRL of the same name as the Primary SRL. Note that the SRL cannot be a volume set or a component volume of a volume set.
- If the Primary RVG includes a volume set, make sure that the component volumes on the Secondary to be added have identical names, lengths, and indices as the component volumes on the Primary.
- Make sure the `/etc/vx/vras/.rdg` file on the Secondary host to be added to the RDS contains the Primary disk group ID. Ensure that each disk group ID entry in the `.rdg` file is on a separate line.

  Refer to the `.rdg` file for the sample format for the disk group ID entry.

  The `vradmin addsec` command checks whether the Primary RVG is authorized to create a corresponding Secondary RVG on the specified Secondary host. A Primary is determined as authorized if the `/etc/vx/vras/.rdg` file on the specified Secondary host contains the Primary disk group ID. If the Primary contains multiple RVGs in the same disk group, only one entry is required. A plus (+) sign in the `/etc/vx/vras/.rdg` file on the Secondary host indicates that all Primary RVGs on all hosts are authorized to create a Secondary RVG on the specified Secondary host.

  The `/etc/vx/vras/.rdg` file on the Secondary host is only used for authorization checking when a Secondary is added, or when remote data volumes are synchronized or verified. To perform these operations after a Secondary takes over from the Primary, the original Primary host should also have an `/etc/vx/vras/.rdg` file containing the disk group ID for the new Primary host.

  To display the Primary disk group ID, issue the following command on the Primary host:

  ```
  # vxprint -l diskgroup
  ```

  For example, to enable host `seattle` to create an RVG on Secondary host `london` the `.rdg` file on the host `london` must have the following entries, each on a new line.

  ```
  1083007373.10.seattle
  ```

To add a Secondary to an RDS

```
# vradmin -g local_diskgroup addsec local_rvgname pri_hostname \
      sec_hostname
```

The argument *`local_diskgroup`* is the name of the disk group on the local host.

The argument *`local_rvgname`* is the name of the RVG on the local host.

The arguments *pri_hostname* and *sec_hostname* are either resolvable hostnames or IP addresses for the Primary and the Secondary hosts. These names are used as `local_host` and `remote_host` attributes while creating RLINKs. The `local_host` and `remote_host` specify the network connection to use for the Primary and Secondary RLINKs.

Use the `-nodcm` option if you do not want to add DCMs to the data volumes. By default, DCMs are automatically added unless the `-nodcm` option is specified.

Note: By default, SRL protection on the new Primary and Secondary RLINKs is set to `autodcm`. If you specify the `-nodcm` option, the `vradmin addsec` command disables SRL protection.

Note that the Secondary RVG is added to the disk group with the same name as the Primary disk group, unless specified otherwise using the `-sdg` option.

Example 1:

This example shows how to add a Secondary host `london_priv` to the RDS, which contains the RVG `hr_rvg`. For replication, this example uses a private network with the Primary hostname `seattle_priv`, Secondary hostname `london_priv`. On the Secondary, the RVG is added to the same disk group as the Primary, that is, `hrdg`. This example automatically adds DCMs to the data volumes.

```
# vradmin -g hrdg addsec hr_rvg seattle_priv london_priv
```

Example 2:

This example shows how to add the Secondary host `london_priv` to the RDS, which contains the RVG `hr_rvg`. It creates the Secondary with the specific Primary and Secondary RLINK names `to_london` and `to_seattle`. The RLINK connects the Primary host `seattle_priv` and the Secondary host `london_priv`. On the Secondary, the RVG is added to the same disk group as the Primary, that is, `hrdg`.

```
# vradmin -g hrdg addsec hr_rvg seattle_priv london_priv \
prlink=to_london srlink=to_seattle
```

Example 3:

This example shows how to add a Secondary host `london-v6_priv` to the RDS, which contains the RVG `hr_rvg`. For replication, this example uses a private IPv6 network with the Primary hostname `seattle-v6_priv`, Secondary hostname `london-v6_priv`. Both hostnames `london-v6_priv` and `seattle-v6_priv` resolve to IPv6 addresses belonging to the private IPv6 network. On the Secondary, the

RVG is added to the same disk group as the Primary, that is, `hrdg`. This example automatically adds DCMs to the data volumes.

```
# vradmin -g hrdg addsec hr_rvg seattle-v6_priv london-v6_priv
```

Example 4:

This example shows how to add a Secondary host `london-v6` to the RDS, which contains the RVG `hr_rvg`. It creates the Secondary with the specific Primary and Secondary RLINK names `to_london-v6` and `to_seattle-v6`. The RLINK connects the Primary host `seattle-v6` and the Secondary host `london-v6`, which resolve to IPv6 addresses `aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh and pppp:qqqq:rrrr:ssss:wwww:xxxx:yyyy:zzzz` respectively. On the Secondary, the RVG is added to the same disk group as the Primary, that is, `hrdg`. This example also automatically adds DCMs to the data volumes.

```
# vradmin -g hrdg addsec hr_rvg aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh \
pppp:qqqq:rrrr:ssss:wwww:xxxx:yyyy:zzzz prlink=to_london-v6 \
srlink=to_seattle-v6
```

Changing the replication settings for a Secondary

When you add a Secondary to an RDS, the default replication attributes of the Secondary are set to `synchronous=off`, `latencyprot=off`, `srlprot=autodcm`, `packet_size=8400` and `bandwidth_limit=none`.

To display the default replication settings for the Secondary, use the following form of the `vxprint` command:

```
vxprint -g diskgroup -Pl
```

If you are using the UDP protocol, this form of the `vxprint` command also shows the default packet size.

You can set up the replication mode, latency protection, SRL protection, transport protocol, packet size, and the bandwidth used by VVR using the replication attributes, such as `synchronous`, `latencyprot`, and `srlprot`. These attributes are of the form `attribute=value`. Each attribute setting could affect replication and must be set up with care.

The `vradmin set` command enables you to change the replication settings between the Primary and a Secondary. This command can be issued from any host in the RDS. It enables you to perform the following tasks:

- See "Setting the mode of replication for a Secondary" on page 145.
- See "Setting the latency protection for a Secondary" on page 146.

- See "Setting the SRL overflow protection for a Secondary" on page 148.
- See "Setting the network transport protocol for a Secondary" on page 149.
- See "Setting the packet size for a Secondary" on page 149.
- See "Setting the bandwidth limit for a Secondary" on page 150.

The `vradmin set` command changes the corresponding attributes on both the Primary and Secondary RLINK. The attributes `synchronous`, `latencyprot`, and `srlprot` are only active on the Primary RLINK; however, the Secondary attributes are already set up and ready for use if the Primary role is transferred to the Secondary.

Setting the mode of replication for a Secondary

You can set up VVR to replicate to a Secondary in synchronous or asynchronous mode by setting the `synchronous` attribute of the RLINK to `override`, or `off` respectively.

Setting the `synchronous` attribute to `override` puts the RLINK in synchronous mode. During normal operation, VVR replicates in synchronous mode, but if the RLINK becomes inactive due to a disconnection or administrative action, VVR switches temporarily to asynchronous mode and continues to receive updates from the application and store them in the SRL. After the connection is restored and the SRL is completely drained, the RLINK automatically switches back to synchronous mode. Most system administrators set the `synchronous` attribute to `override`.

The `vradmin` command does not allow you to set the `synchronous` attribute to `fail`. Use the `vxedit` command to set the attribute `synchronous=fail`. For more information on using the `vxedit` command, refer to the `vxedit` manual page.

Caution: See "Synchronous mode replication considerations" on page 67. if you use the `synchronous=fail` mode.

To enable asynchronous mode of replication

To set the replication to asynchronous mode, set the `synchronous` attribute to `off`.

```
# vradmin -g diskgroup set local_rvgname sec_hostname
synchronous=off
```

The argument *local_rvgname* is the name of the RVG on the local host and represents its RDS.

The argument *sec_hostname* is the name of the Secondary host displayed in the output of the *vradmin printrvg* command. If the RDS contains only one Secondary, the argument *sec_hostname* is optional.

Example - Setting the mode of replication to asynchronous for an RDS

To set the mode of replication to asynchronous for the RDS `hr_rvg` between the Primary `seattle` and the Secondary `london`, issue the following command on any host in the RDS:

```
# vradmin -g hrdg set hr_rvg london synchronous=off
```

To enable synchronous mode of replication

To set the synchronous attribute of the RLINK to `override`, use the following command:

```
# vradmin -g diskgroup set local_rvgname sec_hostname \
        synchronous=override
```

Example - Setting the mode of replication to synchronous for an RDS

To set the mode of replication to synchronous for the RDS `hr_rvg` between the Primary `seattle` and the Secondary `london`, issue the following command on any host in the RDS:

```
# vradmin -g hrdg set hr_rvg london synchronous=override
```

Setting the latency protection for a Secondary

The `vradmin set` command enables you to set the `latencyprot` attribute to `override`, `fail`, or `off`; it also enables you to specify a `latency_high_mark` and a `latency_low_mark`, which indicate when the protection becomes active or inactive.

Set the `latencyprot` attribute to enable latency protection between a Primary and a Secondary.

See “About the latencyprot attribute” on page 94.

Note: Before enabling latency protection, be sure you understand how latency protection works when the Primary and Secondary are connected or disconnected.

See “Setting up latency protection” on page 93.

To enable latency protection

1 Set the `latencyprot` attribute of the corresponding RLINKs on the Primary and Secondary.

To set the `latencyprot` attribute to `override`:

```
# vradmin -g diskgroup set local_rvgname sec_hostname \
        latencyprot=override
```

To set the `latencyprot` attribute to `fail`:

```
#    vradmin -g diskgroup set local_rvgname sec_hostname \
       latencyprot=fail
```

2 Set the `latency_high_mark` and the `latency_low_mark` attributes:

```
# vradmin -g diskgroup set local_rvgname sec_hostname \
        latency_high_mark=high_mark

# vradmin -g diskgroup set local_rvgname sec_hostname \
        latency_low_mark=low_mark
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents the RDS.

The argument *`sec_hostname`* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command.

Note that the value of `latency_high_mark` must be greater than the value of `latency_low_mark`. We recommend that the difference between the value of `latency_high_mark` and the value of `latency_low_mark` be a small number, for example, 50.

To disable latency protection

◆ Setting the `latencyprot` attribute to `off` disables latency protection. This does not limit the number of waiting updates in the SRL.

To set the `latencyprot` attribute to `off`:

```
# vradmin -g diskgroup set local_rvgname sec_hostname
    latencyprot=off
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents the RDS.

The argument *`sec_hostname`* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command.

Setting the SRL overflow protection for a Secondary

VVR provides the following modes of SRL overflow protection: `autodcm`, `dcm`, `override`, `fail`, and `off`.

See "About the srlprot attribute" on page 91.

To enable SRL overflow protection

- Set the `srlprot` attribute of the corresponding RLINK to either `autodcm`, `dcm`, `override`, or `fail`.
  - To set the `srlprot` attribute to `autodcm`, use the following command:

    ```
    # vradmin -g diskgroup set local_rvgname sec_hostname \
            srlprot=autodcm
    ```

  - To set the `srlprot` attribute to `dcm`, use the following command:

    ```
    # vradmin -g diskgroup set local_rvgname sec_hostname \
            srlprot=dcm
    ```

  - To set the `srlprot` attribute to `override`, use the following command:

    ```
    # vradmin -g diskgroup set local_rvgname sec_hostname \
            srlprot=override
    ```

  - To set the `srlprot` attribute to `fail`, use the following command:

    ```
    # vradmin -g diskgroup set local_rvgname sec_hostname \
            srlprot=fail
    ```

  - To set the `srlprot` attribute to `off`, use the following command:

    ```
    # vradmin -g diskgroup set local_rvgname sec_hostname \
            srlprot=off
    ```

  The argument *`local_rvgname`* is the name of the RVG on the local host and represents the RDS.

  The argument *`sec_hostname`* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command.

Setting the network transport protocol for a Secondary

The value specified for the protocol attribute determines the protocol that will be used to communicate between the hosts. You can specify one of the following values for the protocol attribute.

- UDP—The hosts communicate using the UDP/IP protocol.
  VVR automatically calculates the checksum for each data packet it replicates.
- TCP—The hosts communicate using the TCP/IP protocol, which is the default. If a protocol is not specified, then TCP is used as the protocol of communication between hosts.
  If you specify TCP, the VVR checksum is automatically disabled. VVR relies on the TCP checksum mechanism instead. Also, if a node in a replicated data set is using a version of VVR earlier than 5.1 SP1, VVR calculates the checksum regardless of the network protocol.
- STORAGE—Used for bunker replication. The Primary host and the bunker SRL communicate using STORAGE protocol. If the storage is directly accessible by the Primary, for example, DAS or NAS, set the protocol to STORAGE. If the bunker is replicating over IP, the protocol can be set to UDP or TCP.
  See "Introduction to replication using a bunker site" on page 334.

Note: UDP, TCP, and STORAGE are case sensitive.

To set the network protocol

◆ To set the protocol for RDSs in disk group of version 110 or above, the following `vradmin` command can be used:

```
# vradmin -g diskgroup set local_rvgname sec_hostname \
        protocol=protocol_name
```

The argument *`protocol_name`* is the name of the protocol that the Primary will use to replicate to the Secondary. The protocol can be set to either TCP or UDP.

Setting the packet size for a Secondary

The packet size determines the number of bytes in a packet that are sent to the Secondary host. The packet size can be changed using the `packet_size` attribute for UDP mode only. If the protocol is set to TCP, the data is sent using the TCP stream.

See "Choosing the packet size" on page 76. for more information on the `packet_size` attribute.

To set the packet size

◆
```
# vradmin -g diskgroup set local_rvgname
   sec_hostname \
        packet_size=n
```

The argument *local_rvgname* is the name of the RVG on the local host and represents the RDS.

The argument *sec_hostname* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command.

The argument *n* represents the packet size in bytes.

The minimum value for the `packet_size` is 1300 bytes.

The maximum value of the `packet_size` is 65464 bytes.

Example - Setting the packet size between the Primary and Secondary

To set the packet size between the Primary host `seattle` and the Secondary host `london` to 1400 bytes, issue the following command on any host in the RDS:

```
# vradmin -g hrdg set hr_rvg london packet_size=1400
```

Setting the bandwidth limit for a Secondary

Use the `bandwidth_limit` attribute of the `vradmin set` command to set the limit on the network bandwidth used to replicate from the Primary to the Secondary. If `bandwidth_limit` is set to `none`, then VVR uses the available network bandwidth. The default value is `none`. To limit the network bandwidth used by VVR when synchronizing volumes that are not part of an RDS, use the `bandwidth_limit` attribute of the `vradmin syncvol` command.

See "Controlling the network bandwidth used for replication" on page 95.

To control the network bandwidth used for replication

◆ To limit the bandwidth used for replication between the Primary and a Secondary in an RDS, issue the following command on any host in the RDS. In the command, you can either use the units of bandwidth `kbps`, `mbps`, or `gbps`, or abbreviate the units of bandwidth to `k`, `m`, `g`, respectively. The default unit of bandwidth is bits per second (bps).

```
# vradmin -g diskgroup set local_rvgname sec_hostname \
        bandwidth_limit=value
```

The argument *local_rvgname* is the name of the RVG on the local host and represents the RDS.

The argument *sec_hostname* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command.

Example: Limiting network bandwidth between the Primary and the Secondary

To limit the bandwidth to 30 mbps for the RDS `hr_rvg` between the Primary `seattle` and the Secondary `london`, issue the following command on any host in the RDS:

```
# vradmin -g hrdg set hr_rvg london bandwidth_limit=30mbps
```

To disable Bandwidth Throttling for a Secondary

- To disable Bandwidth Throttling for a Secondary in an RDS, issue the following command on any host in the RDS:

```
# vradmin -g diskgroup set local_rvgname sec_hostname \
     bandwidth_limit=none
```

The argument *local_rvgname* is the name of the RVG on the local host and represents the RDS.

The argument *sec_hostname* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command.

Example: Disabling Bandwidth Throttling between the Primary and the Secondary

To disable Bandwidth Throttling for replication between the Primary `seattle` and the Secondary `london` of RDS `hr_rvg`, issue the following command on any host in the RDS:

```
# vradmin -g hrdg set hr_rvg london bandwidth_limit=none
```

To control the network bandwidth used to synchronize volumes

- To limit the network bandwidth used by VVR when synchronizing volumes that are not part of an RDS, issue the following command:

```
# vradmin -g diskgroup syncvol local_vols_list \
        remote_hostname.... bandwidth_limit=value
```

The argument *local_vols_list* is a comma-separated list of volumes on the local host. The names of the volumes on the local and remote hosts are assumed to be the same.

The argument *remote_hostname* is a space-separated list of names of the remote hosts on which the volumes to be resynchronized reside. It must be possible for IP to resolve the remote host names.

Example: Limiting network bandwidth used by VVR when using full synchronization

This example shows how to limit the network bandwidth used by VVR when using full synchronization to synchronize the remote volumes on host `london` with the local volumes `hr_dv01`, `hr_dv02`, `hr_dv03` in the disk group `hrdg` on the local host `seattle`. The names of the disk group and the volumes on the remote host are the same as the names of the disk group and volumes on the local host.

```
# vradmin -g hrdg -full syncvol hr_dv01,hr_dv02,hr_dv03 london  \
        bandwidth_limit=10mbps
```

Synchronizing the Secondary and starting replication

This section explains how to synchronize the Secondary and start replication.

Methods to synchronize the Secondary

You can synchronize the Secondary using the network, using block-level tape backup or by physically moving disks to the Secondary. Automatic synchronization is the recommended method to synchronize the Secondary. Use one of the following methods to synchronize the Secondary depending on your environment:

- Using the network
  - Automatic synchronization
  - Full synchronization with Storage Checkpoint
  - Difference-based synchronization with Storage Checkpoint
- Using block-level tape backup
  - Block-level tape backup and checkpointing
- Moving disks physically
  - Disk Group Split and Join

The following tables explain when and how to use the different synchronization methods:

Using the network to synchronize the Secondary

You can synchronize the Secondary over the network either when the application is active or inactive. SmartMove is supported with each of these methods.

Table 6-1 Synchronizing the Secondary using the network

To Synchronize the Secondary:	Perform:	Using This Command:
completely	automatic synchronization and start replication See "Using the automatic synchronization feature" on page 154.	`vradmin -a startrep`
completely	full synchronization with Storage Checkpoint See "Using the full synchronization feature" on page 502.	`vradmin -full -c checkpoint syncrvg`
when there is little difference between the data on the Primary and Secondary data volumes of the RDS	difference-based synchronization with Storage Checkpoint See "Using difference-based synchronization" on page 509.	`vradmin -c checkpoint syncrvg`

Using block-level tape backup to synchronize the Secondary

Table 6-2 shows how to synchronize the Secondary using block-level tape backup.

Table 6-2 Synchronizing the Secondary using block-level tape backup

To Synchronize the Secondary:	Do the following:	Using This Command:
completely and when a large amount of data must be moved from the Primary to the Secondary	1. Start a Primary Storage Checkpoint.	`vxrvg -c checkpoint checkstart` *`rvg_name`*
	2. Perform a block-level backup of the Primary.	

Table 6-2 Synchronizing the Secondary using block-level tape backup *(continued)*

To Synchronize the Secondary:	Do the following:	Using This Command:
	3. End the Primary Storage Checkpoint.	`vxrvg -c checkpoint checkstart` *`rvg_name`*
	4. Restore the tapes on the Secondary and start replication to the Secondary using the Storage Checkpoint. See "Using block-level backup and Storage Checkpoint" on page 505.	`vradmin -c checkpoint startrep`

Moving disks physically to synchronize the Secondary

Table 6-3 shows how to synchronize the Secondary by moving disks physically.

Table 6-3 Synchronizing the Secondary by moving disks physically

To Synchronize the Secondary:	Use This Feature	Using This Command:
completely by physically moving disks from the location of the Primary host to the location of Secondary host	Disk Group Split and Join	See "Using the Disk Group Split and Join feature" on page 507.

Using the automatic synchronization feature

The Automatic Synchronization feature enables you to transfer the data on the Primary to the Secondary over the network. You can synchronize the Secondary using automatic synchronization either when the application is active or inactive.

The Automatic Synchronization procedure transfers data in the Primary data volumes to the Secondary by reading the Primary data volumes from start to finish and sending the data to the Secondary.

Note: Automatic Synchronization does not maintain the order of writes; therefore, the Secondary is inconsistent until the process is complete.

The Secondary becomes consistent after the automatic synchronization completes. To use Automatic Synchronization successfully, the network must be sized appropriately. Note that the synchronization will complete only if the Primary receives writes at a lesser rate than they can be sent to the Secondary. If the Primary receives writes at a faster rate than they can be sent to the Secondary, the synchronization might never complete, especially if the writes are dispersed widely in the volume.

This feature enables you to synchronize multiple Secondary hosts at the same time. When performing automatic synchronization to multiple Secondary hosts, synchronization proceeds at the rate of the slowest network.

VVR pauses synchronization if the Secondary fails or the network disconnects. If the Primary fails while synchronization is in progress, the synchronization continues from the point at which it had stopped when the Primary recovers.

Prerequisite for using Automatic Synchronization

- Each data volume in the Primary RVG must have a DCM associated to it. If data volumes do not have DCMs, an attempt to automatically synchronize a Secondary fails.

The `vradmin startrep` command when used with the option `-a` enables you to start replication and automatically synchronize the Secondary data volumes with the Primary data volumes in an RDS; it brings the Secondary data volumes up-to-date with the Primary data volumes. You can use this command to synchronize the Secondary when the data volumes contain data and when the application is active or inactive. Replication to another Secondary can be started only after this automatic synchronization completes.

The `vradmin startrep` command can be issued from any host in the RDS. To check the status and progress of the automatic synchronization, use the `vxrlink status` command on the Primary RLINK.

See “Displaying the status of a Secondary” on page 174.

To synchronize the Secondary and start replication using automatic synchronization, issue the following command:

```
# vradmin -g diskgroup -a startrep local_rvgname sec_hostname
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`sec_hostname`* is the name of the Secondary host displayed in the output of the `vradmin printrvg` command. If the RDS contains only one Secondary, the *`sec_hostname`* is optional.

Example—Using the Automatic Synchronization Feature

In this example, the data volumes in the Primary RVG `hr_rvg` on host `seattle` contain valid data and the application is active. To start replication and synchronize the Secondary RVG `hr_rvg` on host `london`, issue the following command:

```
# vradmin -g hrdg -a startrep hr_rvg london
```

Notes on using automatic synchronization

Observe the following notes about using automatic synchronization:

- If you associate a new volume to an RDS while automatic synchronization is in progress, VVR does not automatically synchronize the newly associated data volume.
- In an RDS containing multiple Secondaries that have SRL overflow protection set to dcm, more than one Secondary may require the use of the DCM. If one Secondary is undergoing automatic synchronization and the RLINK of another Secondary is about to overflow, the Automatic Synchronization is abandoned and the DCM becomes active for the overflowing RLINK.
- If you try to automatically synchronize a new RLINK while an existing RLINK is using the DCM mechanism, the automatic synchronization fails.
- To remove a Secondary from a DCM resynchronization process, detach the corresponding Primary RLINK.
- If you try to dissociate a DCM from a data volume while the DCM is in use, the operation fails.
- If the DCM is detached because of I/O errors while the DCM is in use, the resynchronization is abandoned and the RLINKs that are being synchronized are detached.
- You can use automatic synchronization with SmartMove. This combination lets deploy your disaster recovery site must faster because you are replicating far less data.
  See “About SmartMove for VVR” on page 157.

Example for setting up replication using automatic synchronization

This example assumes that the RDS has been created using the example procedure.

See “Creating a Replicated Data Set for the examples” on page 512.

You can synchronize the Secondary using automatic synchronization when the application is active or inactive.

To setup replication using automatic synchronization

- Start Secondary synchronization and replication using automatic synchronization by issuing the following command from any host in the RDS:

```
# vradmin -g hrdg -a startrep hr_rvg london
```

About SmartMove for VVR

The SmartMove for VVR feature enables VVR to leverage information from VxFS knowledge of the file system blocks in use to optimize the time and network bandwidth required for initial synchronization of replicated volumes. This feature is available if there is a VxFS file system mounted on top of the volume that is being synchronized.

If you use SmartMove with automatic synchronization, you can deploy the disaster recovery site faster because you are replicating far less data than the storage provisioned on the system. To use automatic synchronization with SmartMove in a cluster volume replication (CVR) environment, the file system must be mounted on the logowner.

The default behavior is to use the SmartMove for VVR feature for initial synchronization. The commands that use SmartMove during initial synchronization are `vradmin syncrvg/syncvol/startrep` and `vxrlink -a att`.

To turn off SmartMove

- Enter:

```
# vxtune usefssmartmove none
```

The `vradmin verifydata` command has also been enhanced to leverage VxFS knowledge of file system blocks in use for verification.

About thin storage reclamation and VVR

Thin storage helps you optimize your array capacity by allocating storage to applications only when it is needed. When files are created and written to in the file system, storage is allocated from a free storage pool on the array.

However, when you delete files on the host, the storage is not automatically returned to the pool. The result is large amounts of allocated storage on the array that is now unused. You must reclaim this storage manually.

See “Determining if a thin reclamation array needs reclamation” on page 158.

In a VVR environment, you can reclaim storage on volumes configured under an RVG. You can reclaim storage at the disk, disk group, or file system level.

Thin storage reclamation is only supported for LUNs that have the `thinrclm` attribute. VxVM automatically discovers LUNs that support thin reclamation from thin-capable storage arrays. On the host, you can list devices that have the `thinonly` or `thinrclm` attributes.

Thin storage reclamation is not supported on volumes in an RVG that has full instant or space-optimized snapshots that are associated to it. The reclaim command may complete without an error, but the storage space is not reclaimed. Thin storage reclamation is not supported as reclamation on the volume triggers a data transfer from the primary volume to the snapshot volume. Moving this data to the snapshot volume triggers storage allocation at the backend. If there are multiple snapshots, copies of the same data are maintained, which requires more storage than reclaimed. In the case of space-optimized snapshots, the cache object size increases as it copies all the reclaimable data to the space-optimized snapshot.

If you have used Storage Foundation thin storage reclamation in another context, the commands are identical when you use it in a VVR environment.

When you use thin reclamation with VVR, keep in mind the following:

- The VxFS file system must be mounted on the Primary site before you can perform thin reclamation on the volume.
- When you reclaim storage on the Primary site, it is automatically reclaimed on the Secondary site - unless the Primary site is in DCM mode or when autosync is running. The Primary site goes into DCM mode when its SRL overflows.
- You can reclaim storage on the Primary and Secondary sites even if the sites use different types of arrays. The arrays can have different fixed size physical storage allocation units.
- You can reclaim storage during a rolling upgrade with no impact on your systems.

For detailed information on thin storage, as well procedures for reclaiming thin storage, see *Veritas InfoScale™ Solutions Guide*.

Determining if a thin reclamation array needs reclamation

You can only perform thin storage reclamation on LUNs that have the `thinrclm` attribute. An array may be a good candidate for reclamation if:

- Your array-specific commands indicate that storage is nearing the maximum; for example, 90% is used.
- You receive an alert from the array that a certain storage level has been reached.

Even if storage capacity is reaching the maximum, that does not necessarily mean there is any data to reclaim; the data could still be needed. You should investigate further to determine if there are large blocks of deleted or unused data. If there are, run thin reclamation.

For more information reclaiming storage, see *Veritas InfoScale™ Solutions Guide*.

Starting replication when the data volumes are zero initialized

Use the option `-f` with the `vradmin startrep` command to start replication when the Primary and Secondary data volumes are zero initialized. The `vradmin startrep` command can be issued from any host in an RDS.

To start replication to a Secondary in an RDS when the data volumes are zero initialized:

```
# vradmin -g diskgroup -f startrep local_rvgname sec_hostname
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`sec_hostname`* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command. If the RDS contains only one Secondary, the sec_hostname is optional.

Example: Starting replication when the data volumes are zero initialized

To start replication from the Primary RVG `hr_rvg` on `seattle` to the Secondary RVG on host `london` when the data volumes are zero initialized, issue the following command from any host in the RDS:

```
# vradmin -g hrdg -f startrep hr_rvg london
```

Note: The `-f` option can be used to stop the initial sync of the Primary and Secondary volumes if they already contain the same data.

Chapter 7

Displaying configuration information

This chapter includes the following topics:

- Displaying RVG and RDS information
- Displaying information about data volumes and volume sets
- Displaying information about Secondaries
- Displaying a list of Storage Checkpoints
- Displaying statistics with the vrstat display commands
- Collecting consolidated statistics of the VVR components
- Displaying network performance data
- VVR event notification

Displaying RVG and RDS information

You can display replicated volume group (RVG) and replicated data set (RDS) information using either the Veritas InfoScale product installer or Veritas InfoScale Operations Manager (VOM). This section describes how to display information using the Veritas product installer.

For information on using VOM, see the *Veritas InfoScale Operations Manager User guide*.

This section describes the VVR commands that you can use to view the status of the objects that take part in replication. The `vradmin` print commands display the corresponding objects on all hosts. The `vxprint` print commands display detailed information about a specific object on the host on which the command is issued.

Displaying RDS information

Use the `vradmin printrvg` command to display information about the RDSs on a host. You can run the `vradmin printrvg` command from any host in an RDS.

To display information about a specific RDS, on a local host type:

```
# vradmin -g diskgroup printrvg local_rvgname
```

The argument *local_rvgname* is the name of the RVG on the local host. The local RVG name represents its RDS.

To display additional information about an RDS, type:

```
# vradmin -g diskgroup -l printrvg local_rvgname
```

The `-l` option with the `printrvg` command displays additional information about the RVGs in an RDS in a long format, such as the data volume count, number of volume sets, the SRL name, and the RLINK names for each RVG in the RDS. This `-l` option also displays any configuration errors.

To display information about all the RDSs in the specified disk group, type:

```
# vradmin -g diskgroup printrvg
```

To display information about the RDSs that have local RVGs of name *local_rvgname*, type:

```
# vradmin printrvg local_rvgname
```

Displaying individual RVG information

The `vxprint -Vl` command displays detailed information about the status of an individual RVG. This command is useful to determine the role of the Primary or Secondary RVG and the state of the RVG as seen by the operating system.

To display detailed information about an RVG:

```
# vxprint -g diskgroup -Vl rvg_name
```

The following table lists the output of the `vxprint -Vl` command.

Disk Group	Name of the disk group in which this RVG resides.
RVG	Name of the RVG.
info	Displays the `last_tag`, `record_id`, configuration version, and the version of the RVG.

state	The current utility and kernel states.
assoc	Data volumes, SRL, RLINKs, and volume sets associated with the RVG. If a volume set is associated to the RVG, the list of data volumes includes component volumes of the volume set.
att	The RLINKs that are attached. A Primary can have multiple associated, attached RLINKs. A Secondary can have multiple associated RLINKs, but only one attached RLINK.
flags	See "Interpreting RVG flag settings" on page 162.
logowner	The name of the logowner in the cluster when replicating in a shared disk group environment.

Interpreting RVG flag settings

The following table shows the RVG flag settings and their meanings.

open	This flag is always set on the Secondary when it is waiting to receive an IBC from the Primary.
closed	This flag is always set.
primary/secondary	Indicates the role of the RVG.
enabled/attached	I/O and IOCTLs can be performed.
disabled/detached	I/O and IOCTLs cannot be performed.

Displaying information about data volumes and volume sets

This section describes the VVR commands that you can use to view information about the data volumes and volume sets that are being replicated. These display commands can be entered from any host in an RDS.

The `vradmin` print commands display the corresponding objects on all hosts. The `vxprint` print commands display detailed information about a specific object on the host on which the command is issued.

Displaying data volumes in a Replicated Data Set

The `vradmin printvol` command displays information about the data volumes in an RDS and can be entered from any host in the RDS. For data volumes that are

associated to a volume set, the `vradmin printvol` command displays volume set information, including the name of the volume set and the index of the volume.

To display information about the data volumes in an RDS, type:

```
# vradmin -g diskgroup printvol local_rvgname
```

The argument *local_rvgname* is the name of the RVG on the local host. The local RVG name represents its RDS.

Displaying a list of data volumes

To list all data volumes in an RVG in a single column, issue the following command:

```
# vxrvg -g diskgroup [-1] getdatavols rvg_name
```

Displaying information about all failed data volumes

To display information about all failed data volumes that are associated with the specified RVG, issue the following command:

```
# vxrvg -g diskgroup getfailedvols rvg_name
```

The output displays the data volume names, the percentage of the SRL used after the volume failure, and whether the volume is recoverable. The state of the failed volume is stored in the SRL. If the SRL usage reaches 100%, that information is overwritten and therefore the failed volume cannot be recovered.

Displaying an individual data volume

Use the `vxprint -l volume_name` command to display information about a specific data volume. The output fields of special interest for VVR are shown in the following table.

assoc	Shows the RVG to which this data volume is associated (rvg=).
logging	Shows the logging type, which should always be DCM for a data volume.

Displaying a volume set

Use the `vxprint -l volume_set` command to display information about a specific volume set. The output fields of special interest for VVR are shown in the following table.

assoc	Shows the RVG to which this volume set is associated (rvg=), and the data volumes associated to this volume set (appvols=).
replicating	Shows which of the data volumes in this volume set are being replicated.
not-replicating	Shows which of the data volumes in this volume set are not being replicated.

Displaying information about Secondaries

This section describes the VVR commands that you can use to view the status of the objects that take part in replication. These display commands can be entered from any host in an RDS. The `vradmin` print commands display the corresponding objects on all hosts. The `vxprint` print commands display detailed information about a specific object on the host on which the command is issued.

Displaying consolidated replication status

The `vradmin repstatus` command displays the consolidated replication status of the specified Replicated Data Set (RDS). The `vradmin repstatus` command displays the following information about each RVG in the RDS:

- Consolidated view of the RDS
- Replication settings for all Secondary hosts in the RDS
- Status of the data on each Secondary host in the RDS
- Status of replication to each Secondary host in the RDS

To display consolidated replication information about an RDS:

```
# vradmin -g diskgroup [-l] repstatus local_rvgname
```

The argument *local_rvgname* is the name of the RVG on the local host. The local RVG name represents its RDS.

The option `-l` displays additional information, such as RLINK names, replication setting, compression mode, and so on. Similar to the `vradmin -l printrvg` command, the `vradmin repstatus` command also displays configuration errors in the RDS, if any.

Note: If the `vradmin repstatus` command is run on a Secondary that has a few configuration errors or cannot reach the Primary, the output displays the status known to the Secondary before the above condition occurred and therefore might be out of date.

Example:

When the Primary is reachable from all the Secondary hosts and the `vradmin repstatus` command is run from any host:

```
# vradmin -g hrdg -l repstatus hr_rvg
```

Output resembles:

```
Replicated Data Set: hr_rvg
Primary:
  Host name:                 seattle
  RVG name:                  hr_rvg
  DG name:                   hrdg
  RVG state:                 enabled for I/O
  Data volumes:              4
  Vsets:                     1
  SRL name:                  hr_srl
  SRL size:                  4.00 GB
  Total secondaries:         1

Secondary:
  Host name:                 london
  RVG name:                  hr_rvg
  DG name:                   hrdg
  Rlink from Primary:        rlk_london_hr_rvg
  Rlink to Primary:          rlk_seattle_hr_rvg
  Configured mode:           asynchronous
  Latency protection:        off
  SRL protection:            autodcm
  Data status:               inconsistent
  Replication status:        resync in progress (smartsync autosync)
  Current mode:              asynchronous
  Logging to:                DCM (contains 169728 Kbytes) (autosync)
  Timestamp Information:     N/A
  Bandwidth Limit:           30.00 Mbps
  Compression Mode:          Off
```

Example:

When the Primary is unreachable from the Secondary hosts and the `vradmin repstatus` command is run from the Secondary host:

```
# vradmin -g hrdg -l repstatus hr_rvg
```

Output resembles:

```
VxVM VVR vradmin INFO V-5-52-1205 Primary is unreachable or RDS has configuration
 error. Displayed status information is from Secondary and can be out-of-date.
 Replicated Data Set: hr_rvg
 Primary:
 Host name:                  seattle <unreachable>
 RVG name:                   hr_rvg
 DG name:                    hrdg
 RVG state:                  enabled for I/O
 Data volumes:               4
 Vsets:                      1
 SRL name:                   hr_srl
 SRL size:                   4.00 GB
 Total secondaries:          1

 Secondary:
 Host name:                  london
 RVG name:                   hr_rvg
 DG name:                    hrdg
 Rlink from Primary:         rlk_london_hr_rvg
 Rlink to Primary:           rlk_seattle_hr_rvg
 Configured mode:            asynchronous
 Latency protection:         off
 SRL protection:             autodcm
 Data status:                consistent, up-to-date
 Replication status:         replicating (connected)
 Current mode:               asynchronous
 Logging to:                 SRL (0 updates behind, last update ID 18533.0)
 Timestamp Information:      behind by 00:00:00 hours
 Bandwidth Limit:            30.00 Mbps
 Compression Mode:           Off
 Last Update on Primary:     Oct 10 04:32:21
 Secondary up-to-date as of: Oct 10 04:32:21

 Config Errors:
 seattle:                      Pri or Sec IP not available or vradmind not running,
                               stale information
```

The following section describes the important fields displayed by the `vradmin repstatus` command. The values and meaning of each field are listed in tables:

- **RVG state**: Displays the state of the Primary RVG. The following table lists the values for the RVG state field and their meanings.

acting_secondary	This Primary RVG is currently the acting Secondary as part of the fast failback process. Writes to the data volumes in this RVG are disabled independent of whether the RVG is started or stopped.
disabled for I/O	Primary RVG is disabled for I/O, that is, the RVG is stopped.
enabled for I/O	Primary RVG is enabled for I/O, that is, RVG is started.
needs recovery	State after an import or reboot. The `vxrvg recover rvg` command clears this state.
passthru	The Primary RVG is in `passthru` mode because the Primary SRL is detached or missing. See "About RVG PASSTHRU mode" on page 372.

- **Data status**: Shows the data status of this Secondary. The following table lists the values for the Data status field and their meanings:

consistent, behind	Secondary data is consistent but not up-to-date with the Primary data.
consistent, stale	The data on this Secondary is consistent. Replication to this Secondary has been stopped; the Primary RLINK is detached.
consistent, up-to-date	The Secondary data is consistent and is current or up-to-date with the Primary data. The Primary role can be migrated to this Secondary.
inconsistent	The data on the Secondary volumes is not consistent and the Secondary cannot take over.
needs recovery	State after an import or reboot. The `vxrlink recover` command clears this state.
N/A	Current state of the Secondary data cannot be determined. This may occur because of a configuration error on this Secondary. For information about the state, use the *`vxprint -l rlink_name`* command on the Primary and Secondary.

- **Current mode**: Displays the mode of replication, asynchronous or synchronous, that is being used to replicate data to the Secondary. This value can be different

from the configured replication setting if the configured mode is synchronous=override.

See "Changing the replication settings for a Secondary" on page 144.

- **Replication status**: Displays the status of the replication to the Secondary. The following table lists the values for the Replication status field and their meanings:

Value	Meaning
logging to DCM	DCM is active for this Secondary, that is, new updates on Primary are tracked using DCM for this Secondary. The following information may be displayed: needs dcm resynchronization—To continue replication, resynchronize the Secondary using DCM resynchronization. See "Incrementally synchronizing the Secondary after SRL overflow" on page 218. needs failback synchronization—To continue replication, start failback synchronization to this Secondary. See "Failing back using fast failback synchronization" on page 317.
needs failback synchronization	This Primary RVG is acting as Secondary as part of the fast failback process. To continue replication, start failback resynchronization on the new Primary.
not replicating	Data is not being replicated to Secondary because Primary RLINK is in `needs_recovery` state. primary needs_recovery—Primary RLINK in `needs_recovery` state and needs to be recovered before replication can resume.
paused by user	Replication to Secondary is paused because of some administrative action. This results in the following states: primary paused—Primary RLINK is paused. secondary paused—Secondary RLINK is paused.
paused due to error	Replication to Secondary is paused because of the following errors: secondary config error—Secondary has some configuration error. See "Interpreting RLINK flag settings" on page 172. secondary log error—Secondary SRL has an I/O error. See "Interpreting RLINK flag settings" on page 172.

Value	Meaning
paused due to network disconnection	Replication to Secondary is paused because of some network problem.
replicating	connected—Replication can take place if there are updates on the Primary data volumes
resync in progress	Resynchronization to the Secondary is in progress. autosync—Resynchronization type is autosync. dcm resynchronization—Resynchronization after an SRL overflow. failback resynchronization—Resynchronization using failback logging. smartsync—Resynchronization type is autosync using SmartMove.
resync paused by user	Resynchronization to Secondary is paused because of some administrative action. This results in the following states: primary paused—Primary RLINK is paused. secondary paused—Secondary RLINK is paused.
resync paused due to error	Resynchronization to Secondary is paused because of the following errors: secondary config error—Secondary has some configuration error. See “Interpreting RLINK flag settings” on page 172. secondary log error—Secondary SRL has an I/O error. See “Interpreting RLINK flag settings” on page 172.
resync paused due to network disconnection	Resynchronization to Secondary is paused because of some network problem.
stopped	Replication to Secondary is stopped because of the following: Primary detached—Primary RLINK is detached. Secondary detached—Secondary RLINK is detached.
N/A	The replication status cannot be determined. For information about the status, use the `vxprint -l rlink_name` command on the Primary and Secondary.

- **Logging to**: Indicates whether updates for this Secondary are tracked on the Primary using the SRL or DCM. The following table lists the values for the Logging to field and their meanings:

Value	Meaning
DCM (contains xxx Kbytes) (`log_type`)	DCM is active (in use) for the replication to this Secondary. `log_type` can be `autosync`, `failback logging`, or `SRL protection logging`. The yyy% value can sometimes reach beyond 100%. If synchronization is restarted and the DCM map is full, new incoming writes will cause the total yyy% to exceed 100%.
SRL (xxx Kbytes behind, yyy % full)	Updates to be transferred to Secondary are logged into the SRL and are currently occupying xxx Kbytes or yyy% of the SRL
SRL	SRL is used for logging. Check the Data status field for the status of the Secondary data.

If the `vradmin repstatus` command is run on a Secondary and the Secondary is disconnected from the Primary because of a configuration or network error, the Logging to field may show the following values:

Value	Meaning
DCM (log_type)	The last known information about logging type before the Secondary disconnected from the Primary is that it was logging to DCM.
SRL (xxx updates behind, last update ID yyy)	Before Secondary disconnected from the Primary, SRL was used for logging. Secondary was xxx updates behind the Primary, and the last update that was applied on the Secondary has update ID yyy. This information is similar to that displayed by the `vxrlink updates` command.
SRL (updates behind N/A)	Before Secondary disconnected from the Primary, SRL was used for logging. The number of updates this Secondary is behind with the Primary is not known.

- **Timestamp information, Last Update on Primary, Secondary up-to-date as of:**
  These fields are the same as the output that is displayed for the `vxrlink -T` command.
  See "Displaying the status of a Secondary" on page 174.

See "Identifying the most up-to-date Secondary " on page 189.

- **Compression Mode:**
  Displays the mode of compression. The values are on or off.

Displaying a list of RLINKs

To display all the RLINK names in a single column, issue the following command:

```
# vxrvg -g diskgroup [-1] getrlinks rvg_name
```

You can then use the RLINK name to obtain detailed information about a specific RLINK.

Displaying a specific RLINK

Use the `vxprint -Pl` command to display detailed information about the status of an RLINK. This command prints one record per RLINK. The following table lists the information displayed in the output.

To display detailed information about a specific RLINK:

```
# vxprint -g diskgroup -Pl rlink_name
```

Disk Group	Name of the disk group.
RLINK Name	Name of the RLINK.
Info	timeout, packet_size, record_id, latency high, low marks and bandwidth_limit.
State	Displays utility state of the RLINK - active, stale, etc.
synchronous, latencyprot, and srlprot	The current configuration settings for the replication mode, the latency mode, and SRL protection.
assoc	The RVG to which the RLINK is associated.
remote_host/IP_addr/port	The name of the remote host for the RLINK, its IP address, and the port number.
remote_dg	The name of the disk group on the remote system.
remote_dg_dgid	The disk group ID assigned when the disk group was created on the remote system.
remote_rvg_version	The rvg_version of the remote RVG.
remote_rlink	The name of the corresponding RLINK on the remote host.

remote_rlink_rid	The record_id of the corresponding RLINK on the remote host.
local_host/IP_addr/port	The name of the local host, its IP address, and the port number it uses for communication to the remote host.
protocol	The transport protocol used by the RLINK for communicating between the hosts. The protocol can be either UDP/IP or TCP/IP.
checkpoint	Displays the Storage Checkpoint only if the Primary RLINK is attached using the Storage Checkpoint or the Secondary RLINK has been restored using the Storage Checkpoint.
flags	See "Interpreting RLINK flag settings" on page 172.

Note: To display all the RLINK names in a single column, issue the following command:

```
# vxrvg -g diskgroup [-1] getrlinks rvg_name
```

Interpreting RLINK flag settings

The following table lists the various flags that can appear in the flags field of the `vxprint -Pl` output.

Note: The Primary and Secondary RLINKs are communicating only when the `connected` flag is on. However, replication is taking place only if the following set of flags is displayed:

```
write enabled attached consistent connected
```

In all other cases, a corrective action may be needed.

autosync	The RDS is in the process of automatic synchronization.
attached	The RLINK is attached to the RVG.
bunker_target	Indicates that this is the RLINK from the Secondary to the bunker node.
can_sync	If the inconsistent and can_sync flags are set, there is enough information in the Secondary SRL volume to make the Secondary consistent again and capable of taking over.
cant_sync	The RLINK is inconsistent, and this Secondary needs a complete resynchronization before it can take over or replicate.

compression_enabled	Indicates that compression is enabled on the RLINK.
connected	The RLINK is connected to the corresponding RLINK on the remote host and replication can take place.
consistent	The state of the data volumes on the Secondary is suitable for takeover.
dcm_logging	DCM is in use, due to either autosync, failback sync, or an SRL overflow.
detached	The RLINK is stale and not taking part in replication.
disabled	The RLINK is not attached and is not replicating.
disconnected	The two RLINKs are not connected and are not replicating.
enabled	The RLINK is attached. If the `connected` flag is displayed, replication can take place. If the `disconnected` flag is displayed, replication is not taking place.
fail	An I/O error occurred while writing to a data volume on the Secondary.
inconsistent	The data in the Secondary volumes is not consistent and the Secondary cannot take over.
needs_recovery	State after an import or reboot. The `vxrecover` command clears this state.
primary_paused	The Primary RLINK has been paused and the RLINKs are not replicating.
resync_started	The resynchronization of the Secondary has been started.
resync_paused	The resynchronization has been started but is not currently active because of some problem.
secondary_config_err	There is a mismatch between the configuration of the volumes on the Primary and the Secondary - either a volume is missing on the Secondary or its length is not the same as that of the corresponding volume on the Primary.
secondary_log_err	An I/O error has occurred on the Secondary SRL; replication cannot continue until the SRL has been dissociated and a new one associated.
secondary_paused	The Secondary RLINK has been paused and the RLINKs are not replicating.
smartsync	Automatic synchronization is being used with SmartMove.

Displaying the status of a Secondary

Use the `vxrlink status` command to determine the status of a Secondary. This command displays different information depending on what state the replication is in for that Secondary. For example, whether the Primary is currently replicating to the Secondary, synchronizing the Secondary with a Storage Checkpoint, using the DCM to resynchronize the Secondary, or using automatic synchronization for the Secondary. To determine the state of the replication, use the `vradmin repstatus` command.

See "Displaying consolidated replication status" on page 164.

If the state is replicating, the `vxrlink status` command displays whether the Secondary corresponding to the specified RLINK is up-to-date and if not, how much the Secondary is behind.

Note that outstanding writes are shown even if the Secondary is replicating in synchronous mode. Although for synchronous mode, the write is considered complete to the application when the network acknowledgment is received from the Secondary, VVR still considers the write outstanding until it is written to the data volume on the Secondary.

If automatic synchronization or DCM resynchronization is in progress, the `vxrlink status` command shows the progress of the automatic synchronization.

To show the status of a Secondary

```
# vxrlink -g diskgroup status rlink_name
```

If replication is in progress, the output resembles:

```
VxVM VVR vxrlink INFO V-5-1-4640 Rlink rlink_name has <x> outstanding
writes, occupying <y> Kbytes (17%) on the SRL
```

If automatic synchronization is in progress, the output resembles:

```
VxVM VVR vxrlink INFO V-5-1-4464 Rlink rlink_name is in AUTOSYNC.
100864 Kbytes remaining.
```

If DCM resynchronization is in progress, the output resembles:

```
VxVM VVR vxrlink INFO V-5-1-12887 DCM is in use on Rlink rlink_name.
DCM contains 7200 Kbytes (1%) of the Data Volume(s).
```

To display the Secondary status periodically, specify a time interval using the `-I` option. For example, to print the status of the Secondary every five seconds, use the command:

```
# vxrlink -g diskgroup -i5 status rlink_name
```

If replication is in progress, the output resembles:

```
VxVM VVR vxrlink INFO V-5-1-4640 Rlink rlink_name has <x>
outstanding writes, occupying <y> Kbytes (17%) on the SRL
VxVM VVR vxrlink INFO V-5-1-4640 Rlink rlink_name has <x>
outstanding writes, occupying <y> Kbytes (19%) on the SRL
```

If automatic synchronization is in progress, the output resembles:

```
VxVM VVR vxrlink INFO V-5-1-4464 Rlink rlink_name is in
AUTOSYNC. 100864 Kbytes remaining.
VxVM VVR vxrlink INFO V-5-1-4464 Rlink rlink_name is in
AUTOSYNC. 94464 Kbytes remaining.
```

If automatic synchronization is in progress and SmartMove is enabled, the output resembles:

```
VxVM VVR vxrlink INFO V-5-1-0 Rlink rlink_name is in SMART
AUTOSYNC. 2425696 Kbytes remaining.
VxVM VVR vxrlink INFO V-5-1-0 Rlink rlink_name is in SMART
AUTOSYNC. 2367840 Kbytes remaining.
```

To display the status of an RLINK with a timestamp, use the `vxrlink status` command with the `-T` option. This is useful if the Secondary is not up-to-date. The output of the command displays a timestamp in the locale's appropriate time format to indicate the time by which the Secondary is behind.

For example, if there are pending writes in the Primary SRL, use the following command to check the status of the Primary:

```
# vxrlink -g diskgroup -T status rlink_name
```

The output resembles:

```
VxVM VVR vxrlink INFO V-5-1-4640 Rlink rlink_name has <x>
outstanding writes, occupying <y> Kbytes (20%) on the SRL
VxVM VVR vxrlink INFO V-5-1-0 Rlink rlink_name is behind by 0:00:40 hours
```

The second message indicates the time by which the RLINK is behind.

Note: If the system time is reset to a value different from that of the current system time, then, the output of the `vxrlink -T status` command will appropriately show a negative or an inaccurate value, until the updates that were done before resetting the system time get replicated.

Displaying a list of Storage Checkpoints

VVR enables you to get a list of the Primary Storage Checkpoints using the `vxrvg cplist` command; the `vxrlink cplist` command enables you to get a list of the Secondary Storage Checkpoints. The `vxrvg cplist` and `vxrlink cplist` commands can be run on the Primary only. VVR supports a maximum of 46 Storage Checkpoints, and hence the list displays a maximum of 46 Storage Checkpoints. If you try to create more than the specified number of Storage Checkpoints, an error message prompts you to delete older Storage Checkpoints before creating new Storage Checkpoints.

Primary Storage Checkpoints are created using the *`vxrvg -c checkpoint_name checkstart`* command on the Primary and are associated with an RVG. Issue the `vxrvg cplist` command to display a list of the existing Primary Storage Checkpoints associated with the specified RVG. The Primary Storage Checkpoints can be deleted using the *`vxrvg -c checkpoint_name checkdelete rvg_name`* command.

Secondary Storage Checkpoints are created using the *`vxrlink -c checkpoint_name pause`* command on the Secondary and are associated with the RLINK. Issue the `vxrlink cplist` command on the Primary to display a list of the existing Secondary Storage Checkpoints associated with the specified RLINK. The Secondary Storage Checkpoint can be deleted by using the *`vxrlink -c checkpoint_name checkdelete rlink_name`* command.

Note: The `vxrlink cplist` command and the `vxrlink checkdelete` command must be run on the Primary only.

The displayed information includes details about each Storage Checkpoint, such as Storage Checkpoint name, size, percentage of SRL used, and whether the Storage Checkpoint has been started or completed. If the SRL usage reaches 100%, the Storage Checkpoint overflows and becomes unusable. In this case, VVR displays the message `Checkpoint overflowed`.

To display a list of Primary Storage Checkpoints, enter the following command on the Primary:

```
# vxrvg -g diskgroup cplist rvg_name
```

To display a list of Secondary Storage Checkpoints, enter the following command on the Primary:

```
# vxrlink -g diskgroup cplist rlink_name
```

where *rlink_name* is the name of the Primary RLINK that connects to the Secondary where the `vxrlink -c checkpoint_name pause` was issued. The output resembles:

```
Name         MBytes  % Log    Started/Completed
----         ------  ------   -----------------
a8           200        5     Completed
a9           800        20    Completed
a6           2000       40    Started
```

Displaying statistics with the vrstat display commands

This section describes the VVR commands that you can use to view statistics about the RLINKs and the volumes in an RVG, for all the hosts in an RDS. The `vrstat` command combines the output of commands such as `vxrlink stats`, `vxrlink status`, `vxstat`, and `vxmemstat` in a single command to display the statistics about the RLINKs and the volumes in an RVG, for all hosts in the RDS.

The messages are displayed at a default frequency of 10 seconds, which is the frequency at which the `vrstat` command collects the statistics. To change the frequency of the display, set the `VRAS_STATS_FREQUENCY` environment variable to a required value in the `/etc/vx/vras/vras_env` file.

After setting the environment variable to a new value, restart the `vradmind` daemon as follows:

```
# /etc/init.d/vras-vradmind.sh restart
```

Displaying the consolidated statistics

To display the consolidated statistics of the RLINKs, SRL, data volumes, and memory tunables for the RDSs on a host, use the `vrstat` command without specifying any option.

To view the consolidated statistics:

```
# vrstat
```

Displaying the RLINK information for all the hosts in the RDS

The `vrstat -R` command displays detailed statistics for the RLINKs on all the hosts in an RDS. This information can be used to assess connectivity and network problems between the hosts. The `vrstat -R` command can be executed from the

Primary and the Secondary. The output of this command is a combination of the outputs of the `vxrlink stats` and `vxrlink status` commands.

To view information about all the RLINKs in an RDS:

```
# vrstat -R [local_rvgname]
```

The argument *`local_rvgname`* is the name of the RVG on the local host and is optional. The local RVG name represents its RDS.

If you specify the *`local_rvgname`* name, the `vrstat -R` command displays the information for all the RLINKs for the specified RVG. Otherwise, the command displays the information for all the RLINKs for all the RDSs.

For an RVG in a shared disk group, the host_name is the logowner and the displayed information reflects activity on the logowner.

The output of this command resembles:

```
Mon Oct 27 15:44:21 2003
Replicated Data Set hr_rvg:

Data Status:
london: up-to-date.

Network Statistics:
Messages                Errors                              Flow Control
--------                ------                              ------------
# Blocks RT(msec)  Timeout  Stream  Memory  Delays  NW Bytes NW Delay Timeout
seattle - london
260     133120  5       1    0       0       333     178000     1        20
279     0       11      0    0       0       0       100000     1        30
Bandwidth Utilization 72908 Kbps.
Compression ratio 14.03
Bandwidth savings 92.87%
```

Note: The Compression ratio and Bandwidth savings fields are only displayed if compression is enabled.

The fields in the output are similar to those in the output of the `vxrlink stats` command.

See “Displaying network performance data” on page 186.

Displaying information about all the data volumes for all the hosts in the RDS

The `vrstat -V` command displays detailed statistics for all the data volumes associated with the specified RVG on each host in the RDS. The `vrstat -V` command can be executed from the Primary and the Secondary.

To view information about all the data volumes associated with an RVG in an RDS:

```
# vrstat -V [local_rvgname]
```

The argument *`local_rvgname`* is the name of the RVG on the local host and is optional. The local RVG name represents its RDS.

If you specify the *`local_rvgname`* name, the `vrstat -V` command displays information about all the volumes associated with the specified RVG. Otherwise, the command displays information about all the volumes in all the RDSs.

The output of this command resembles:

```
Mon Oct 27 15:49:15 2003
Replicated Data Set hr_rvg:


Data Volume-I/O Statistics:
HOST   NAME    OPERATIONS            BLOCKS          AVG TIME(ms)
               READ   WRITE       READ  WRITE     READ      WRITE
seattle hr_dv01   0       0          0      0      0.0        0.0
london hr_dv01    0     412          0 210944      0.0       12.0


Mon Oct 27 15:49:25 2003
Replicated Data Set hr_rvg:


Data Volume-I/O Statistics:
HOST   NAME    OPERATIONS            BLOCKS          AVG TIME(ms)
               READ   WRITE      READ   WRITE     READ      WRITE
seattle hr_dv01   0       0         0       0      0.0        0.0
london hr_dv01    0       0         0       0      0.0        0.0
```

The output of this command includes the following details:

- The *`host_name`* of the host for which the information is being displayed.
  For an RVG in a shared disk group, *`host_name`* is the logowner, and the displayed information reflects activity on the logowner host.

- Name of the volume for which the information is being displayed.
- The total number of read and write operations performed on a volume.
- Number of blocks that have been read from or written to the volume.
- Average time in milliseconds to complete the read and write operations.

Displaying information about the SRL volumes for all the hosts in the RDS

The `vrstat -S` command displays detailed statistics about the SRL for every host within the RDS. This command can be executed from the Primary and the Secondary.

To view information on all the SRL volumes for every host in an RDS:

```
# vrstat -S [local_rvgname]
```

The argument *`local_rvgname`* is the name of the RVG on the local host and is optional. The local RVG name represents its RDS.

If you specify the *`local_rvgname`* name, the `vrstat -S` command displays information about the SRL in the RDS. Otherwise, the command displays information about the SRL in all the RDSs.

The output of the `vrstat` command resembles:

```
Mon Oct 27 15:53:11 2003
Replicated Data Set hr_rvg:


SRL-I/O Statistics:
HOST       NAME          OPERATIONS            BLOCKS         AVG TIME(ms)
                       READ     WRITE        READ     WRITE     READ  WRITE
seattle  hr_srl           0       258           0    132885      0.0   17.6
london   hr_srl           0         0           0         0      0.0    0.0


Mon Oct 27 15:53:21 2003
Replicated Data Set hr_rvg:


SRL-I/O Statistics:
HOST        NAME          OPERATIONS            BLOCKS         AVG TIME(ms)
                        READ     WRITE        READ    WRITE     READ  WRITE
```

```
seattle  hr_srl            0       143           0    73430      0.0   17.6
london   hr_srl            0         0           0        0      0.0    0.0
```

The output of this command includes the following details:

- The *host_name* of the host for which the information is being displayed.
  For an RVG in a shared disk group, *host_name* is the logowner, and the displayed information reflects activity on the logowner host.
- Name of the SRL volume for which the information is being displayed.
- The total number of read and write operations performed on a volume.
- Number of blocks that have been read from or written to the volume.
- Average time in milliseconds to complete the read and write operations.

Displaying information about the memory tunable parameters for all the hosts in the RDS

The `vrstat -M` command displays detailed information about the memory tunable parameters. This command can be executed from the Primary and the Secondary. The output of the `vrstat -M` command is similar to the output displayed by the `vxmemstat` command.

If you specify the *local_rvgname* name with the `vrstat -M` command, it displays the information about the memory tunables for all the hosts in that RDS. Otherwise, the command displays information about the memory tunable parameters for all the hosts in all the RDSs.

To view information about the memory tunable parameters:

```
# vrstat -M [local_rvgname]
```

The argument *local_rvgname* is the name of the RVG on the local host and is optional. The local RVG name represents its RDS.

The output of this command resembles:

```
Mon Oct 27 15:57:15 2003
 Replicated Data Set hr_rvg:
 Memory-pool Statistics:
 Host        Pool        DG     Min      Max      In      Allocated      Max     Waiting
                            Size     Size     Use                    Used
 ------    ----------    ---- ------   ------   -----   ----------    -----    -------
 seattle    Voliomem      -     1024     12443       0      1024           0       no
 seattle    NMCOM-hr_rvg dg1    1024      4096       0      1024           0       no
 seattle    RViomem       -     1024     12443       0      1024           0       no
```

```
seattle    WRSHIP        -     1024      4096       0      1024           0       no
seattle    RDBK-hr_rvg   -     1024      4096       0      1024           0       no
london     Voliomem      -     1024     12441       0      1024           0       no
london     NMCOM-hr_rvg dg1    1024      4096       0      1024           0       no
london     RViomem       -     1024     12441       0      1024           0       no
london     WRSHIP        -     1024      4096       0      1024           0       no
london     RDBK-hr_rvg   -     1024      4096       0      1024           0       no
```

The output of the `vrstat` command includes the following details:

- The *`host_name`* of the host for which the information is being displayed.
  For an RVG in a shared disk group, *`host_name`* is the logowner, and the displayed information reflects activity on the logowner host.
- Name of the memory tunable parameter.
- Name of the disk group in which this RVG is present.
- Minimum and maximum size for each tunable parameter.
- The amount of the allocated space that is being used.
- Amount of space allocated for the parameter.
- Maximum space that has been used by a parameter.

Determining VVR network bandwidth usage and compression ratio

Use the `vrstat` command to determine the network bandwidth being used by VVR and the VVR compression ratio.

To view the network bandwidth currently being used by VVR and the VVR compression ratio

```
# vrstat -R local_rvgname
```

The argument *`local_rvgname`* is the name of the RVG on the local host and is optional. The local RVG name represents its RDS.

If you specify the *`local_rvgname`* name, the `vrstat -R` command displays the information about the RLINKs for the specified RVG. Otherwise, the command displays the information for all the RLINKs for all the RDSs.

Example:

To view the compression ratio and the network bandwidth used by the RDS `hr_rvg` between the Primary `seattle` and the Secondary `london`, issue the following command on any host in the RDS:

```
# vrstat -R hr_rvg
```

The output resembles:

```
Replicated Data Set hr_rvg:
Data Status:
london: DCM contains 1157888 Kbytes.
Network Statistics:
Messages              Errors                 Flow Control
--------              ------                 ------------
# Blocks RT(msec) Timeout Stream Memory Delays NW Bytes NW Delay
Timeout
seattle - london
356     182272  6      1      0       0       280    271000  1 10
339     0       15     0      0       0       0      100000  1 20
Bandwidth Utilization 72908 Kbps.
Compression ratio 14.03
Bandwidth savings 92.87%
```

Note: The Compression ratio and Bandwidth savings fields are only displayed if compression is enabled.

Collecting consolidated statistics of the VVR components

You can configure VVR to collect statistics of the VVR components. The collected statistics can be used to monitor the system and diagnose problems with the VVR setup. VVR collects the statistics generated by the VVR commands `vxrlink stats`, `vxrlink status` and `vxrvg stats` for all the imported disk groups, and the system level commands `netstat`, `vmstat`, and `vxmemstat`. The output of these commands are stored in separate files.

By default, VVR collects the statistics automatically when the `vradmind` daemon starts. Configuring VVR to collect statistics according to your requirements involves modifying the values of the environment variables in the `vras_env` file, located in the `/etc/vx/vras` directory.

Note: If the `vradmind` daemon is not running, VVR stops collecting the statistics.

To configure VVR to collect statistics automatically

1 Modify the default values for the environment variables specified in the `vras_env` located in the `/etc/vx/vras` directory file to suit your requirements. The following table provides information about the variables:

Environment Variable	Description
`VRAS_ENABLE_STATS`	Specifies whether you want the statistics collection to start automatically. Set `VRAS_ENABLE_STATS=on` to enable statistics collection. This is the default. Set `VRAS_ENABLE_STATS=off` to disable statistics collection.
`VRAS_STATS_FREQUENCY`	Specifies the frequency in seconds at which the statistics should be collected for the VVR commands, `vxrlink stats`, `vxrlink status` and `vxrvg stats`. By default, `VRAS_STATS_FREQUENCY` is set to 10 seconds.
`VRAS_NETSTAT_FREQUENCY`	Specifies the time interval in seconds over which the statistics for the different network protocols should be collected. By default, `VRAS_NETSTAT_FREQUENCY` is set to 300 seconds.
`VRAS_VMSTAT_FREQUENCY`	Specifies the time interval in seconds over which the memory and CPU utilization statistics should be collected. By default, `VRAS_VMSTAT_FREQUENCY` is set to 300 seconds.
`VRAS_STATS_DAYS_LOG`	Specifies the number of days for which the collected statistics should be preserved. After this time the earlier statistics are automatically deleted. By default, `VRAS_STATS_DAYS_LOG` is set to three days.

2 Restart the `vradmind` daemon as follows:

```
# /etc/init.d/vras-vradmind.sh restart
```

You can restart `vradmind` even while the application is active and replication is in progress.

See "Understanding how VVR stores the statistics" on page 185.

Understanding how VVR stores the statistics

VVR stores the statistics collected for each of the commands, that is, `vxmemstat`, `netstat`, `vmstat`, `vxrlink stats`, `vxrlink status` and `vxrvg stats` in separate files that are stored in the `/var/vx/vras/stats/` directory. Each file stores the statistics for a day and only those files for the period specified by the `VRAS_STATS_DAYS_LOG` variable are preserved. The earlier files are automatically deleted. VVR stores the statistics in files that are named using the following convention:

- `statsType_dgName_objectN`—stores the statistics of the VVR components collected by the commands `vxrlink stats`, `vxrlink status` and `vxrvg stats`.
- `statsType_hostname_date`—stores the system level statistics collected by the commands `netstat`, `vmstat`, and `vxmemstat`.

The data collected for the `vxmemstat` command is the same as that displayed by the `vxmemstat -e` command.

The output collected for the `vxrlink stats` command is a combination of the fields displayed by the `vxrlink -e stats` command and the `vxrlink stats` command. Using network performance data the output of `vxrlink stats` is displayed under the messages, errors, and flow control headings.

See "Displaying network performance data" on page 186.

About the VVR statistics Messages heading

The fields displayed under this heading are the same as those displayed by the `vxrlink stats` command, with the exception of an additional field `Blocks (ACKed)`. This field displays information about the number of transmitted blocks that have been acknowledged.

About the VVR statistics Errors heading

Most of the fields under this heading are similar to those displayed by the `vxrlink -e stats` command; however, some of the field names are different. The following table provides the mapping:

Table 7-1 vxrlink commands map

Fields in the vxrlink stats command when vradmind collects the statistics	Fields in the vxrlink -e stats command
Memory	No memory available
Slots	No message slots available
Pool	No memory available in nmcom pool on Secondary
Timeout	Timeout
Packet	Missing packet
Message	Missing message
Stream	Stream
Checksum	Checksum
Transaction	Unable to deliver due to transaction

About the VVR statistics Flow control heading

The fields displayed under this heading are the same as those displayed by the `vxrlink stats` command.

See "Displaying network performance data" on page 186.

Displaying network performance data

The `vxrlink stats` command reports detailed information about the state of the network. It displays network statistics that can be used to assess network problems. Use the network performance data to determine the optimum network configuration for efficient use of system resources.

The `vxrlink stats` command can be executed repeatedly at given intervals using the `-i` interval option. In this case, the displayed values indicate the change since the last interval, except for the average round-trip value, which displays a moving average. The `vxrlink stats` command can be executed from the Primary and the Secondary. The RLINK statistics are reset when the RLINK disconnects. For detailed information on available options, refer to the `vxrlink` manual page.

The output of the `vxrlink stats` command includes the following details:

- Number of messages transmitted.

- Number of 512-byte blocks transmitted.
- Average round-trip per message.
  The size of the message affects the average round-trip per message.
- Number of timeouts or lost packets.
  If the number of timeouts is high, it indicates that network is very lossy. This needs to be fixed.
- Number of stream errors
  Stream errors occur when the RLINK attempts to send messages faster than the network can handle.
- Number of memory errors
  Memory errors occur when the secondary has insufficient buffer space to handle incoming messages. To reduce the number of errors, try increasing the value of the tunable, `vol_max_nmpool_sz` on the secondary.
- Current Timeout Value
  This represents the packet timeout in milliseconds.

Displaying extended replication statistics

You can use the `vxrlink stats` command with the `-e` option to generate extended statistics, in addition to the statistics generated by the `vxrlink stats` command. The output generated by this command can be useful in assessing the reason for failure at the time it occurred.

The `vxrlink stats -e` command can be executed repeatedly at given intervals using the `-i` interval option. In this case, the displayed values indicate the change since the last interval. The `vxrlink stats -e` command can be executed from the Primary as well as the Secondary. The RLINK statistics are reset when the RLINK disconnects.

For detailed information about the available options, refer to the `vxrlink` manual page.

The output of the `vxrlink stats -e` command is displayed under the headings Messages and Errors. Each of these headings has the appropriate fields to display the required information. The first is the Messages heading which displays the following information:

- Number of blocks sent
  Displays the number of bytes that have been transmitted. This is different from the `Blocks` attribute displayed by the `vxrlink stats` command (without the -e option), which only displays the number of blocks that have been acknowledged.
- Compressed msgs

Displays the number of messages sent in compressed form.

- Compressed data
  Displays the amount of compressed data in bytes.
- Uncompressed data
  Displays the amount of uncompressed data in bytes.
- Compression ratio
  Displays the compression ratio achieved.
- Bandwidth savings
  Displays the network bandwidth savings achieved.

Note: The Compressed msgs, Compressed data, Uncompressed data, Compression ratio, and Bandwidth savings fields are only displayed if compression is enabled.

The Messages heading is followed by the Errors heading. It has nine fields that display the different kinds of error encountered, three of which are similar to that in the `vxrlink stats` command. The output includes the following details:

- No memory available
  This error occurs when there is no space in the systems kernel memory to process the message.
- No message slots available
  This error occurs if there is no memory to store the packets of the message that have arrived out of sequence. If a packet arrives out of sequence then it requires to be stored in the message buffer until all the related out-of-sequence packets arrive and can be assembled.
- No memory available in nmcom pool on Secondary
  The buffer space determined by the VVR tunable `vol_max_nmpool_sz` is already full and cannot store any new messages that arrive at the Secondary.
- Timeout errors
  Indicates the number of timeout errors, that is, the number of times the Primary timed out while waiting for an acknowledgment from the Secondary.
- Missing packet errors
  Indicates the number of times the last packet of a message was received, before one or more packets of the same message were received.
- Missing message errors
  Indicates the number of times messages have arrived out of sequence.
- Stream errors

Stream errors occur when the RLINK attempts to send messages faster than the network can handle.

- Checksum errors
  Displays the number of data checksum errors. Every time a packet is received at the Secondary VVR performs a checksum to ensure that the packet data is the same as that sent by the Primary.
- Unable to deliver due to transaction errors
  Displays the number of times the packets could not be delivered to the Secondary, due to transaction errors. If the Secondary is busy with some kernel operations when the packet arrives at the Secondary, then these packets may not be delivered until the transaction is complete.

Identifying the most up-to-date Secondary

VVR provides the `vxrlink updates` command to identify the most up-to-date Secondary in a VVR configuration. The `vxrlink updates` command can be issued on a Secondary only.

For multiple Secondaries, the `vxrlink updates` command enables you to determine the Secondary that contains the most up-to-date data and hence the most suitable replacement for the Primary in the case of a takeover.

For a single Secondary, the `vxrlink updates` command can be used to determine the extent to which the Secondary is behind the Primary. You can decide whether or not to take over the Primary role by looking at the update ID of the Secondary, number of updates the Primary is ahead of the Secondary, and how long you expect the Primary to be unavailable.

Issue the following command on the Secondary.

```
# vxrlink -g diskgroup -T updates rlink_name
```

To display only the update ID in the output, use the `vxrlink updates` command without the `-T` option. The output displays the update ID as a sequence number. A sequence number is a 64-bit value that increases incrementally and therefore is unique for each new update. The output of the `vxrlink updates` command displays the 64-bit number as two 32-bit sequence numbers separated by a dot. For example:

```
high_seq_num . low_seq_num
```

To display the exact time on the Primary at which the Secondary is up-to-date, use the `vxrlink updates` command with the `-T` option. The `-T` option displays the exact time in hours by which the Secondary is behind. Note that the update information may be inaccurate if:

- the Secondary has been rebooted and even before it comes up, the Primary becomes unavailable.
- the Secondary reboots and the RLINK gets disconnected.

The output of the `vxrlink -T updates` command is displayed in a three column structure with two rows; ID and Time. The ID row displays the update IDs. The timestamp in the Time row indicates the time at which the update was written on the Primary. The time is displayed in *`Mon date time`* format, where *`Mon`* is a locale abbreviated month name followed by the *`date`* and the *`time`* in the locale's appropriate time format.

The first column displays the last update ID and the time at which it was written on the Primary.

The second column displays the last update ID that has been received on the Secondary and the time when it was written on the Primary. If the Secondary is up-to-date then the ID and the time in this column is the same as that in the first column. However, if the Secondary is behind, then the ID and the time is different from that in the first column.

The third column indicates the exact number of updates by which the Secondary is behind and also the time in the locale's appropriate time format by which it is behind. This value is obtained as a difference between the second and first column.

Note: If the system time is reset to a value different from that of the current system time, the output of the `vxrlink -T updates` command appropriately shows a negative or an inaccurate value, until the updates that were completed before resetting the system time are replicated.

Example—to determine the most up-to-date Secondary

This example shows how to determine the most up-to-date Secondary in an RDS containing Primary `seattle` and the Secondaries `london` and `newyork`. This example displays the last update ID received by the Secondary and the last known update ID on the Primary.

To determine the most up-to-date Secondary

1 On the Secondary `london`, enter the following command:

```
# vxrlink -g diskgroup updates to_seattle
```

The output resembles:

```
Secondary has received an update ID of 37364.104, last known
update ID on Primary is 99 updates ahead.
```

2 On the Secondary `newyork`, enter the following command:

```
# vxrlink -g diskgroup updates to_seattle
```

The output resembles:

```
Secondary has received an update ID of 37364.118, last known
update on Primary is 95 updates ahead.
```

Compare the output on `london` and `newyork`. The host `newyork` has received the later update with the update ID 37364.118, whereas the other Secondary `london` is behind `newyork` by 14 updates. The host `newyork` is more up-to-date than the host `london`.

Example—to determine the status of the Secondary

This example shows how to determine the status of the Secondary in an RDS containing Primary `seattle` and the Secondary `london`, using the `-T` option with the `vxrlink updates` command.

To determine the status of the Secondary in an RDS containing Primary `seattle` **and the Secondary** `london`

◆ On the Secondary `london`, enter the following command:

```
# vxrlink -g diskgroup -T updates to_seattle
```

If the Secondary is up-to-date the output displays the following:

```
       Last update       Secondary             Secondary
       on Primary        up-to-date as of      behind by
ID     34666.0           34666.0                  0
Time   Oct 16 11:17:44   Oct 16 11:17:44       00:00:00
```

If the Secondary is not up-to-date the output displays the following:

```
       Last update       Secondary             Secondary
       on Primary        up-to-date as of      behind by
ID     34666.640         34666.592                48
Time   Oct 16 11:17:44   Oct 16 11:17:42       00:00:02
```

VVR event notification

VVR provides the `vrnotify` utility to notify administrators of VVR specific events, such as SRL full, resynchronization complete, etc. You can receive notification for a VVR event on the Primary or Secondary node, or both the nodes in an RDS.

The `vrnotify` command enables you to write a script that receives VVR event notification and notifies administrators of these events through email, pager, etc. See the examples in this section to see how event notifications can also be used to keep history of various events.

If you do not specify the *`local_rvgname`* in the `vrnotify` command, event notification is started for all the RDSs on the local host.

If any of the RDSs have RVGs in a shared disk group, `vrnotify` provides notification about events on the logowner for those RVGs.

Use the `-g` option to receive event notifications for RVGs in a specific disk group.

The `vrnotify` command displays the VVR events until you explicitly terminate or kill the command.

To receive event notifications on the Primary or Secondary, enter the following command:

```
# vrnotify -g diskgroup local_rvgname....
```

The argument *local_rvgname...* is a space-separated list of the names of the RVGs on the local host in the specified disk group.

The `vrnotify` command displays each event on a new line in the following format:

host_name:*event_type*:*RDS_name*:*event message*

For an RVG in a shared disk group, *host_name* is the logowner, and the displayed event information reflects activity on the logowner host.

The `vrnotify` command displays the following types of events:

Table 7-2 Event notifications

Event Type	Event Message
resync_started	Resync started on Primary RVG
resync_stopped	Resync stopped on Primary RVG
resync_paused	Resync paused on Primary RVG
lat_throttle_on	Latency throttling started
lat_throttle_off	Latency throttling stopped
lat_throttle_override	Latency throttling overridden
lat_throttle_fail	Latency throttling caused I/O failures
srlprot_throttle_on	SRL overflow protection throttling started
srlprot_throttle_off	SRL overflow protection throttling stopped
srlprot_override	SRL overflow protection overridden
srlprot_fail	SRL overflow protection caused I/O failures
srl_overflow	Replication stopped due to SRL overflow
srlprot_dcm_on	Started using DCM for SRL protection
srlprot_dcm_off	Stopped using DCM
rlk_connect	RLINK connected to remote
rlk_disconnect	RLINK disconnected from remote
srl_log_warn	SRL percentage full has changed by 10%
repmode_sync	Replicating in synchronous mode

Table 7-2 Event notifications *(continued)*

Event Type	Event Message
repmode_async	Replicating in asynchronous mode
repibc_freeze	Replication on Secondary frozen due to IBC
repibc_unfreeze	Replication on Secondary unfrozen after IBC
rvg_pritosec	RVG role changed from Primary to Secondary
rvg_sectopri	RVG role changed from Secondary to Primary
rvg_pritoactsec	RVG role changed from Primary to acting Secondary
rvg_actsectopri	RVG role changed from acting Secondary to Primary
rlk_paused	Secondary RLINK paused because of a configuration error
ibcmsg_discarded	IBC was discarded due to timeout on the Secondary.

Example:

The following example script shows how to use the `vrnotify` utility to receive event notifications for the `hr_rvg` RDS in the hrdg disk group and send email to the alias `vvradmin` if the event `srl_warning` occurs.

```
#!/bin/sh
IFS=:
vrnotify -g hrdg hr_rvg | while read host event rvg msg
do
    case $event in
    srl_log_warn)
        (echo "This message is sent by VVR notify mechanism"
         echo "$msg for RVG $rvg on host $host"
         ) | mailx -s "VVR SRL Log Warning" vvradmin;;
    esac
done
```

Chapter 8

Administering Volume Replicator

This chapter includes the following topics:

- Administering data volumes
- Administering the SRL
- Administering replication
- Administering the Replicated Data Set
- Administering Storage Checkpoints
- Creating RVG snapshots
- Verifying the DR readiness of a VVR setup
- Backing up the Secondary

Administering data volumes

You can administer Volume Replicator (VVR) using either the command line interface (CLI) or Veritas InfoScale Operations Manager. This chapter describes how administer VVR using the CLI.

For information on using Veritas InfoScale Operations Manager, see the *Veritas InfoScale Operations Manager User guide*.

An RDS is made up of data volumes on the Primary and Secondary. VVR enables you to perform tasks on one or more data volumes that are associated to the RDS. You can also associate or dissociate volumes or volume sets from the RDS.

Associating a volume to a Replicated Data Set

This section describes how to use the `vradmin addvol` command to add a volume to a RDS. The `vradmin addvol` command can also be used to add a volume set to a RDS, or to add a component volume to a volume set that is associated to an RDS. A component volume of a volume set cannot be added to the RDS directly.

See "Associating a volume set to an RDS" on page 201.

You can use the `vradmin addvol` command to add a volume to a RDS even when replication is in progress. This command associates a volume to all the RVGs of the RDS. Note that volumes of the same name and same length must exist on all Secondaries and the Primary of the RDS. You must create volumes of the required layout on the Secondary and Primary hosts before issuing the `vradmin addvol` command. If necessary, the `vradmin addvol` command can be used to add a volume to an RDS that only has a Primary RVG. In this case, there are no Secondary volumes.

By default, the `vradmin addvol` command adds DCM logs to the data volumes being added to the RDS, if they have not already been added. If any of the data volumes contains a DRL log, the `vradmin addvol` command removes the DRL log before adding the DCM to the data volume.

The `-nodcm` option with the `vradmin addvol` command adds data volumes to the RDS without adding DCMs to the data volumes. If any of the data volumes has a DRL, the DRL is removed before the data volume is associated with the RVG. If `-nodcm` is issued when any of the RLINKs has `srlprot` set to `dcm` or `autodcm`, and any of the data volumes being added to the RDS does not already have a DCM log, the command will fail.

The `vradmin addvol` command can be run from any host in the RDS. If the `vradmin addvol` command fails on any of the hosts in the RDS during its execution, the volume is not added on any host.

Before adding a volume, the `vradmin addvol` command displays a warning and prompts the user to confirm whether or not the Primary and Secondary volumes contain the same data. Verify that the Primary and Secondary volumes contain the same data before adding a volume.

See "Performing offline data verification " on page 275.

If the verification shows that the Primary and Secondary volumes do not contain the same data, synchronize the Primary and Secondary volumes.

See "Synchronizing volumes on the local host and remote hosts" on page 199.

To skip this confirmation, use the `-s` option with the `vradmin addvol` command. The `-s` option to the `vradmin addvol` command proves useful in scripts.

Prerequisites for adding a volume to an RDS:

- Create volumes of same name and length as the Primary volume on all the hosts in the RDS.
- Verify that the volumes to be added are inactive.
- Synchronize the volumes using the `vradmin syncvol` command before adding the volumes to the RDS.
  See "Synchronizing volumes on the local host and remote hosts" on page 199.

Note: To add a volume to an RDS that has only a Primary RVG, the prerequisites above do not apply.

To add a volume to an RDS

```
# vradmin -g diskgroup addvol local_rvgname volume_name
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`volume_name`* is the name of the volume to be added to the RDS. Only one volume can be added at a time.

Use the `-nodcm` option when you do not want to add DCMs to the data volumes. By default, DCMs are automatically added.

Example - Associating an existing volume to an RDS

This example shows how to add an existing volume `hr_dv01` to all RVGs of the RDS. The disk group `hrdg` contains the local RVG `hr_rvg` of the RDS. To add the volume `hr_dv01` to all RVGs in the RDS and automatically add DCMs to the data volumes, type the following command on any host:

```
# vradmin -g hrdg addvol hr_rvg hr_dv01
```

Verifying the data on the Primary and Secondary volumes

The `vradmin syncvol` command when used with the `-verify` option enables you to verify whether the remote volumes and the corresponding local volumes are identical before adding them to an RDS. Use this command when the volumes are not associated with an RVG and the application is inactive (the volumes are not in use). VVR also allows you to verify the data volumes after they have been added to an RDS.

See "Verifying the data on the Secondary" on page 273.

The `vradmin -verify syncvol` command only reports the amount of data that is different in percentage between remote and local volumes; it does not synchronize remote volumes with local volumes. If you find that the Primary data and Secondary data do not match then you can use some manual means such as backup and restore or some other method to make the data on the new Secondary volume the same as the Primary and then add it to the RDS.

Note: Remote volumes can be verified with local volumes only if the `/etc/vx/vras/.rdg` file on the remote host contains a local disk group ID entry. Ensure that each disk group ID entry in the `.rdg` file is on a separate line.

Note that the order of the volume names in the local and remote volume lists is important. The `vradmin -verify syncvol` command verifies the first volume in the remote volume list with the first volume in the local volume list, and so on. Hence, the number of volumes in the local and remote volume lists must be same. Also, the remote disk group name must be specified if volume names are different on local and remote hosts.

It is recommended that the names of the volumes on the local and remote hosts be the same. However, you can verify volumes with different names on the local and remote hosts using the `vradmin -verify syncvol` command.

To verify the difference between the local and remote data volumes

```
# vradmin -g diskgroup -verify syncvol local_vols_list \
  remote_hostname...
```

The argument *`local_vols_list`* is a comma-separated list of volumes on the local host. The names of the volumes on the local and remote hosts are assumed to be the same.

The argument *`remote_hostname`* is a space-separated list of names of the remote hosts on which the volumes to be verified reside. It must be possible for IP to resolve the remote host names.

See "About SmartMove for VVR" on page 157.

Example - Verifying the differences between remote volumes on a host

This example shows how to verify the differences between the remote volumes on host `london` with the local volumes `hr_dv01, hr_dv02, hr_dv03` in the disk group `hrdg` on the local host `seattle`. The names of the disk group and the volumes on the remote host are the same as names of the disk group and volumes on the local host.

```
# vradmin -g hrdg -verify syncvol hr_dv01,hr_dv02,hr_dv03 london
```

Synchronizing volumes on the local host and remote hosts

The `vradmin syncvol` command enables you to synchronize remote volumes with local volumes when the volumes are not associated with an RVG and the volumes are not in use. The data in the volumes on the local host, where you enter the command, is transferred over the network to the volumes on the remote host. The volumes to be synchronized can be component volumes of a volume set. The `vradmin syncvol` command can also be used to synchronize a volume set itself.

Use the `vradmin syncvol` command only to synchronize volumes that are not part of an RVG. For example, before adding a volume to an RDS, synchronize the volume using the `vradmin syncvol` command, and then add it to the RDS.

Using the `vradmin syncvol` command, you can synchronize remote volumes with local volumes using one of the following options:

- Difference-based synchronization
- Full synchronization

By default, the `vradmin syncvol` command synchronizes the volumes using difference-based synchronization. We recommend that the names of the volumes on the local and remote hosts be the same. However, you can synchronize volumes with different names on the local and remote hosts using the `vradmin syncvol` command.

You can supply a list of volumes to be synchronized. If you choose this method, the order of the volume names in the local and remote volume lists is important. The `vradmin syncvol` command synchronizes the first volume in the remote volume list with the first volume in the local volume list, and so on. Hence, the number of volumes in the local and remote volume lists must be the same. Also, the remote disk group name must be specified if volume names are different on the local and remote hosts.

Note: Remote volumes can be synchronized with local volumes only if the `/etc/vx/vras/.rdg` file on the remote host contains a local disk group ID entry. Ensure that each disk group ID entry in the `.rdg` file is on a separate line.

To enable the `vradmin syncvol` command for a specific disk group on a remote host, enter the local disk group ID in the `/etc/vx/vras/.rdg` file on the remote host. To enable the `vradmin syncvol` command for all disk groups on a remote host, enter a plus (+) sign in the `/etc/vx/vras/.rdg` file on the remote host. For more information, see the `vradmin(1M)` manual page.

Before synchronizing volumes, the `vradmin syncvol` command displays a warning and prompts the user to confirm whether or not the data on the volumes on the remote host can be overwritten with the data on the volumes on the local host. To skip this confirmation, use the `-s` option with the `vradmin syncvol` command. The `-s` option to the `vradmin syncvol` command proves useful in scripts.

See "About SmartMove for VVR" on page 157.

Synchronizing volumes using full synchronization

In full synchronization, all data is transferred between hosts. Use full synchronization to create initial copies of volumes. To do a full synchronization, specify the option `-full`.

To synchronize volumes on local and remote hosts using full synchronization

- Synchronize the volumes with the following command:

  ```
  # vradmin -g diskgroup -full syncvol local_vols_list \
      remote_hostname....
  ```

  The argument *`local_vols_list`* is a comma-separated list of volumes on the local host. The names of the volumes on the local and remote hosts are assumed to be the same.

  The argument *`remote_hostname`* is a space-separated list of names of the remote hosts on which the volumes to be resynchronized reside. It must be possible for IP to resolve the remote host names.

Note: When using the `syncvol` command on an IPv6 network, if there are two colons in the address, the disk group name and the volume name must be specified. If the disk group name and the volume name are not specified, the command fails.

Example - Full synchronization of volumes on local and remote hosts

This example shows how to do a full synchronization of the remote volumes on host `london` with the local volumes `hr_dv01, hr_dv02, hr_dv03` in the disk group `hrdg` on the local host `seattle`. The names of the disk group and the volumes on the remote host are the same as the names of the disk group and volumes on the local host.

```
# vradmin -g hrdg -full syncvol hr_dv01,hr_dv02,hr_dv03 london
```

Synchronizing volumes using difference-based synchronization

In difference-based synchronization, VVR compares the blocks of data between the hosts and then transfers over the network only those blocks of data that are

different. Difference-based synchronization is useful when there is little difference between the data on the local and remote volumes.

To synchronize volumes on local and remote hosts using difference-based synchronization:

```
# vradmin -g diskgroup syncvol local_vols_list remote_hostname....
```

The argument *local_vols_list* is a comma-separated list of volumes on the local host. In this case, the names of the volumes on the local and remote hosts are the same.

The argument *remote_hostname* is a space-separated list of names of the remote hosts on which the volumes to be resynchronized reside. It must be possible for IP to resolve the remote host names.

Example 1:

This example shows how to do a difference-based synchronization of the remote volumes on host london with the volumes `hr_dv01, hr_dv02, hr_dv03` in the disk group `hrdg` on local host `seattle`. The names of the disk group and the volumes on the remote host are the same as names of the disk group and volumes on the local host.

```
# vradmin -g hrdg syncvol hr_dv01,hr_dv02,hr_dv03 london
```

Example 2:

In this example, the names of the volumes on the remote host are different from the names of the volumes on the local host. It shows how to do a difference-based synchronization of the remote volumes `hr_dvmaster` and `hr_dvoralog` on host `london` with the local volumes `hr_dv01` and `hr_dv02` in the disk group hrdg.

```
# vradmin -g hrdg syncvol hr_dv01,hr_dv02 \
      london:hrdg:hr_dvmaster,hr_dvoralog
```

Associating a volume set to an RDS

This section describes how to associate a volume set to an RDS. A volume set is a container object for a group of volumes that can be a part of a multi-device file system (SmarTier). Associating the volume set to an RDS enables you to replicate a SmarTier.

For more information about volume sets, see the *Storage Foundation Administrator's Guide*.

The component volumes of a volume set have assigned indices. Applications use these indices to identify a component volume. To be able to successfully start

applications on the Secondary if a disaster occurs, the component volumes of the volume set on the Secondary must have the same indices as those of the corresponding Primary volumes.

This section assumes that the volume set that is to be replicated already exists on the Primary. If the volume set does not exist on the Primary, create the volume set.

The volume set need not exist on the Secondary; however, if the volume set already exists on the Secondary, the volume set on the Secondary must have the same characteristics as the volume set on the Primary. That is, the volume sets must have the same name, the same count of component, and the component volumes must have the same names, sizes, and indices. If the volume set does not exist on the Secondary, and the component volumes do exist with the same names, sizes, and indices as on the Primary, the `vradmin addvol` command creates the volume set on the Secondary.

You cannot associate a volume set, or a component volume of a volume set, as an SRL.

After a volume set is associated to an RDS, the `vradmin addvol` command can be used to add an independent volume to the volume set. A component volume added to the volume set this way becomes a part of the RVG and is replicated.

By default, the `vradmin addvol` command adds DCM logs to the component volumes when a volume set is added to the RDS, if they have not already been added. If any of the data volumes contains a DRL log, the `vradmin addvol` command removes the DRL log before adding the DCM to the data volume. The `-nodcm` option with the `vradmin addvol` command adds component volumes to the RDS without adding DCMs to the volumes. If `-nodcm` is issued when any of the RLINKs has `srlprot` set to `dcm` or `autodcm`, and any of the volumes being added to the RDS does not already have a DCM log, the command will fail. This behavior is the same as for independent data volumes.

To associate a volume set to an RDS

1. Verify whether the component volumes of the volume set on the Primary and its Secondaries have identical indices. To view the indices, use the following command:

   ```
   # vxvset -g diskgroup list vset_name
   ```

2. If the indices of the component volumes on the Primary volume set and Secondary volume set are identical, go to step 4.

3. If the indices of the component volumes on the Primary volume set and Secondary volume set are different, perform the following steps on the Secondary:

- Dissociate each volume from the volume set using the following command:

  ```
  # vxvset -g diskgroup rmvol vset_name compvol_name
  ```

  When you remove the last volume, the volume set is also removed.

- Create the volume set using the following command:

  ```
  # vxvset -g diskgroup -o index make vset_name \
      compvol_name index
  ```

- Associate each of the remaining volumes to the volume set specifying the index of the corresponding volumes on the Primary using the following command:

  ```
  # vxvset -g diskgroup -o index addvol vset_name \
      compvol_name index
  ```

4 Associate the volume set to the RDS using the following command:

```
# vradmin -g diskgroup addvol rvg_name vset_name
```

Note: use the volume set name in the command, not the names of each component volume. Specifying the component volume name causes the command to fail.

Example:

This example shows how to associate the component volumes `hr_cv1` and `hr_cv2` of the volume set `hr_vset` to the RDS `hr_rvg`. The example assumes that the component volumes have identical indices.

To associate the component volumes of the volume set to the RDS

1 Verify whether the component volumes of the volume set `hr_vset` on the Primary and its Secondaries have identical indices using the following command on the Primary and its Secondaries:

```
# vxvset -g hrdg list hr_vset
```

Output looks like this:

```
VOLUME        INDEX       LENGTH     KSTATE    CONTEXT
hr_cv1        0           8388608    ENABLED      -
hr_cv2        1           8388608    ENABLED      -
```

2 Associate the component volumes `hr_cv1` and `hr_cv2` to the RDS `hr_rvg` using the following command:

```
# vradmin -g hrdg addvol hr_rvg hr_vset
```

To associate an independent volume to a volume set associated to an RDS

◆ Associate the volume set to the RDS using the following command:

```
# vradmin -g diskgroup -tovset vset_name addvol rvg_name \
   volume_name[:index]
```

If an index is specified, that index is used to add the volume to the volume set on all the hosts in the RDS. The command fails if the specified index is already in use.

If an index is not specified, the `vradmin addvol` command ensures that the same index is used to add the volume to the volume set on all the hosts in the RDS.

Example

This example shows how to associate the independent volume `hr_cv3` to the volume set `hr_vset`, which is associated to the RDS `hr_rvg`.

◆ Associate the component volumes `hr_cv3` to the volume set `hr_vset` using the following command:

```
# vradmin -g hrdg -tovset hr_vset addvol hr_rvg hr_cv3
```

Associating a Data Change Map to a data volume

The `vradmin createpri`, `vradmin addsec`, and `vradmin addvol` commands associate the Data Change Map (DCM) to the data volume and mirrors the DCM by default. This section describes how to associate DCMs to a data volume in a new or existing VVR configuration.

To associate a Data Change Map to a Data Volume

The `vxassist` command enables you to associate a DCM to a new data volume or an existing data volume.

1. Create the data volume and associate the DCM as follows:

   ```
   # vxassist -g diskgroup make dv_name....... logtype=dcm
   ```

   OR

2. Associate the DCM with an existing data volume as follows:

   ```
   # vxassist -g diskgroup addlog dv_name logtype=dcm
   ```

VVR mirrors the DCM by default. If `loglen` is not specified, `vxassist` calculates a suitable size for the DCM.

See "Determining the region size " on page 205.

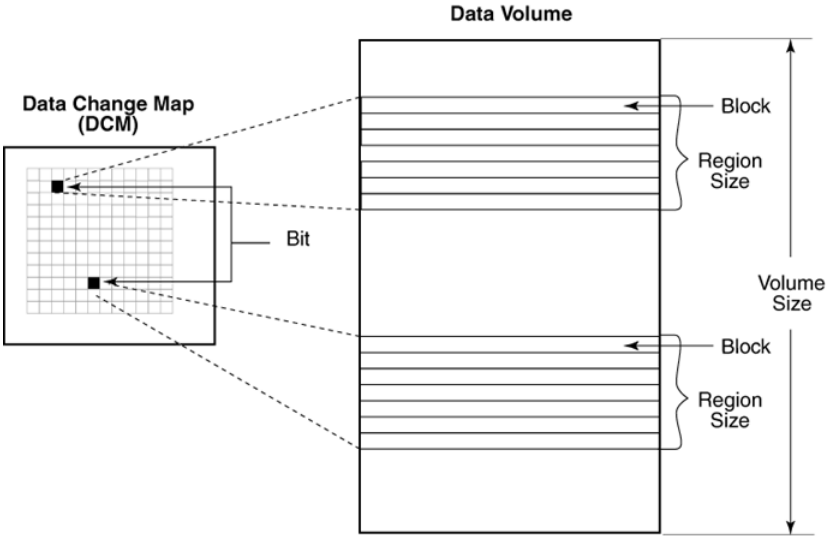
Note: If you try to grow a volume that has a DCM, an error message warns you if the DCM is not large enough for the increased size. In this case, dissociate the DCM, grow the volume, and then associate a new DCM to the volume.

Determining the region size

VVR calculates the DCM size based on the size of the volume. The default size of the DCM ranges from 4K to 256K depending on the size of the volume. However, you can specify the size of the DCM to a maximum of 2 MB. Internally, the DCM is divided into two maps: the active map and the replay map. Each bit in the DCM represents a contiguous number of blocks in the volume it is associated with, which is referred to as the region.

Figure 8-1 shows the DCM and the region size.

Figure 8-1 Data Change Map showing region size



The region size is calculated based on the volume size divided by half the DCM size in bits. The minimum region size is 64 blocks or 32K.

Table 8-1 gives examples of the region sizes for volumes of different sizes in a non-CDS (Cross-Platform Data Sharing) disk group with the default DCM size and a user-specified DCM size of 2 MB.

Table 8-1 Region sizes for volumes in a non-CDS disk group

Volume Size	Default DCM Size	Region Size for Default DCM Size	Region Size for a DCM Size of 2 MB Specified by the User
1 MB	1K	32K	32K
100 MB	1K	32K	32K
200 MB	2K	32K	32K
400 MB	4K	32K	32K
1 GB	9K	32K	32K
2 GB	17K	32K	32K
4 GB	33K	32K	32K

Table 8-1 Region sizes for volumes in a non-CDS disk group *(continued)*

Volume Size	Default DCM Size	Region Size for Default DCM Size	Region Size for a DCM Size of 2 MB Specified by the User
8 GB	65K	32K	32K
20 GB	161K	32K	32K
40 GB	161K	64K	32K
100 GB	201K	128K	32K
200 GB	229K	224K	32K
400 GB	247K	416K	64K
1 TB	249K	1056K	160K

Table 8-2 gives examples of the region sizes for volumes of different sizes in a CDS (Cross-Platform Data Sharing) disk group with the default DCM size and a user-specified DCM size of 2 MB.

Table 8-2 Region sizes for volumes in a CDS disk group

Volume Size	Default DCM Size	Region Size for Default DCM Size	Region Size for a DCM Size of 2 MB Specified by the User.
1 MB	16K	32K	32K
100 MB	16K	32K	32K
200 MB	16K	32K	32K
400 MB	16K	32K	32K
1 GB	16K	32K	32K
2 GB	32K	32K	32K
4 GB	48K	32K	32K
8 GB	80K	32K	32K
20 GB	176K	32K	32K
40 GB	176K	64K	32K

Table 8-2 Region sizes for volumes in a CDS disk group *(continued)*

Volume Size	Default DCM Size	Region Size for Default DCM Size	Region Size for a DCM Size of 2 MB Specified by the User.
100 GB	208K	128K	32K
200 GB	240K	224K	32K
400 GB	256K	416K	64K
1 TB	256K	1056K	160K

Resizing a data volume in a Replicated Data Set

The `vradmin resizevol` command enables you to resize a data volume in a Replicated Data Set (RDS) even when replication is in progress. You can resize an independent data volume or a component volume of a volume set. You cannot use the `vradmin resizevol` command to resize an entire volume set, only individual component volumes. The `vradmin resizevol` command resizes the data volumes in all the RVGs in the RDS. The `vradmin resizevol` command can be entered from any host in an RDS.

Caution: To avoid any problems with the file system on the Secondary, run the `vradmin resizevol` command only when the Secondary is up-to-date. VVR replicates changes to the meta data of a file system on the Primary data volumes to the Secondary. If a takeover happens while these changes are yet to be applied to the Secondary data volumes, the size of the file system may not match the size of the underlying data volume and it may not be possible to mount the file system on the new Primary. If this occurs, run the file system-specific commands to recover the file system.

Important notes on resizing a data volume in a Replicated Data Set

Observe the following notes on resizing a data volume in a Replicated Data Set:

- If the Primary data volume contains a file system, the `vradmin resizevol` command also resizes the file system using the `vxresize` command. For more information, see the `vxresize(1M)` manual page.
- The `vradmin resizevol` command pauses replication, resizes the data volume, and then resumes replication.

- If you want to increase the size of a data volume, make sure there is enough space on the Primary and the Secondary.

  Note: When you increase the size of a data volume, the newly added portions on the Primary and Secondary data volumes are not synchronized. In this case, the output of the `vradmin verifydata` command will show that the checksums for the Primary and Secondary data volumes do not match.

- If the `vradmin resizevol` command fails on any of the hosts in the RDS during its execution, the original volume sizes are not restored. This results in volume size mismatch on the Primary and its Secondaries. To correct this mismatch, correct the error condition and then reissue the `vradmin resizevol` command and resume the Secondary RLINKs.

Prerequisites for resizing a data volume in an RDS

The following items are the prerequisites for resizing a data volume in an RDS:

- The data volume must exist in the disk group and be associated with the RVGs for all hosts in the RDS.
- If you want to increase the size of a data volume, make sure there is enough space in the disk group on the Primary and the Secondary by issuing the following command:

```
# vxdg -g diskgroup free
```

- To resize a volume in an RDS:

  ```
  # vradmin -g diskgroup [-f] resizevol local_rvgname \
  volume_name volume_length [pridiskname=primary_disk_names] \
  [secdiskname=secondary_disk_names]
  ```

  The argument *local_rvgname* is the name of the RVG on the local host and represents its RDS. The `-f` option is required if the data volume involved in the `resizevol` operation is being decreased in size.

  The argument *volume_name* is the name of the data volume to be resized. You can specify a component volume of a volume set. Do not specify a volume set name.

  The argument *volume_length* is the desired size of the data volume to be resized. You can specify the volume length using the standard length convention. You can specify a prefix of either the plus (+) or minus (-) sign to increase or decrease the data volume size by the specified amount.

  The optional arguments `pridiskname` and `secdiskname` enable you to specify a comma-separated list of disk names for the resize operation. The space required for the resize operation is allocated from the specified disks on the primary and secondary, respectively.

Examples:

The following examples show how to resize to different lengths an existing volume `hr_dv01` in all RVGs of the RDS represented by its local RVG `hr_rvg`. The disk group `hrdg` contains the local RVG `hr_rvg`.

To resize the volume `hr_dv01` to 100 gigabytes, type the following command on any host in the RDS:

```
# vradmin -g hrdg resizevol hr_rvg hr_dv01 100G
```

To increase the size of the data volume `hr_dv01` by 100 megabytes when the Primary and Secondary data volumes are the same size, type the following command on any host in the RDS:

```
# vradmin -g hrdg resizevol hr_rvg hr_dv01 +100M
```

To decrease the size of the data volume `hr_dv01` by 500K when the Primary and Secondary data volumes are the same size, type the following command on any host in the RDS:

```
# vradmin -g hrdg -f resizevol hr_rvg hr_dv01 -500K
```

Renaming a data volume in a Replicated Data Set

The `vxedit set` command enables you to rename a data volume in a Replicated Data Set (RDS). Renaming the data volume using the `vxedit set` command pause replications, and you must resume replication using the `vradmin resumerep` command.

Note: If a volume is renamed on the Primary, but not on the Secondary, a configuration error results and the RLINK will be disconnected.

See "Data volume name mismatch error during modification of an RVG" on page 365.

To rename a volume in an RDS

1. Rename the volume:

   ```
   # vxedit set primary_datavol=new_name secondary_datavol
   ```

   where *new_name* is the new name of the data volume. Running this command pauses the replication.

2. Resume the Secondary RLINK by issuing the following command on any host in the RDS:

   ```
   # vradmin resumerep rvg_name
   ```

Dissociating a data volume from its Replicated Data Set

You can remove a data volume, a volume set, or a component volume of a volume set from a Replicated Data Set (RDS) using the `vradmin delvol` command. The `vradmin delvol` command dissociates a data volume from all the RVGs in an RDS; the volumes are not deleted.

The `vradmin delvol` command can be entered from any host in an RDS. If the `vradmin delvol` command fails on any of the hosts in the RDS during its execution, the original configuration remains unchanged.

To remove a data volume from an RDS when the Primary RVG has been stopped

- Type the following command on any host in the RDS:

```
# vradmin -g diskgroup delvol local_rvgname \
      volume_name|vset_name
```

The argument *local_rvgname* is the name of the RVG on the local host and represents its RDS.

The argument *volume_name* is the name of the volume to be removed from the RDS. If the volume specified is a component of a volume set, this command removes the component volume from the RDS but not from the volume set.

The argument *vset_name* can be used to specify a volume set name instead of a volume name; in this case, the entire volume set is dissociated from the RDS.

To remove a component volume from a volume set associated to the RDS when the Primary RVG has been stopped

- Type the following command on any host in the RDS:

```
# vradmin -g diskgroup delvol -fromvset local_rvgname \
   volume_name
```

The argument *local_rvgname* is the name of the RVG on the local host and represents its RDS.

The argument *volume_name* is the name of the component volume to be removed from the volume set. The specified volume is also removed from the RDS.

To remove a data volume from an RDS when the Primary RVG has not been stopped

- Use this procedure only with caution.

Note: Although you can use `-f` option with `vradmin delvol` command to remove a data volume from an RDS when the Primary RVG has not been stopped, this is not the recommended approach. It is recommended that you stop the Primary RVG before proceeding with this command.

Type the following command on any host in the RDS:

```
# vradmin -g diskgroup -f delvol local_rvgname volume_name
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`volume_name`* is the name of the volume to be removed from the RDS.

Example:

This example shows how to remove a data volume `hr_dv01` from all RVGs of its RDS. The data volume `hr_dv01` resides on the local host `london` on which the command is entered. The data volume `hr_dv01` is associated with the local RVG `hr_rvg`, which belongs to the disk group `hrdg`.

```
# vradmin -g hrdg delvol hr_rvg hr_dv01
```

To remove a component volume from a volume set associated to the RDS when the Primary RVG has not been stopped

- Use this procedure only with caution.

Note: Although you can use `-f` option with `vradmin delvol` command to remove a component volume from a volume set associated to an RDS when the Primary RVG has not been stopped, this is not the recommended approach. It is recommended that you stop the Primary RVG before proceeding with this command.

Type the following command on any host in the RDS:

```
# vradmin -g diskgroup -f delvol -fromvset local_rvgname \
      volume_name
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`volume_name`* is the name of the component volume to be removed from the volume set. The specified volume is also removed from the RDS.

Mapping the name of a Secondary data volume to a differently named Primary data volume

We recommend that you use the same name for a data volume in a Primary RVG and the corresponding data volume in a Secondary RVG. However, each Secondary data volume can have a different name from that of the corresponding Primary data volume. The Primary does not know whether the name is mapped to a different name at any given Secondary. The name mapping information is maintained entirely at the Secondary. To facilitate name-mapping, each data volume associated to an RVG has a `primary_datavol` field. This field can be set to the name of the corresponding data volume on the Primary.

By default, global mapping is in effect, that is, the `primary_datavol` field is not used on any Secondary data volumes. This requires that all the Secondary data volumes have the same names as used on the Primary.

One of the prerequisites for adding a Secondary using the `vradmin addsec` command is that data volumes of the same names and lengths as the Primary must exist on the Secondary. When adding a Secondary using the `vradmin addsec` command, the Secondary data volumes cannot have different names from that of the corresponding Primary data volumes.

If you use different names for the Secondary data volumes and their corresponding Primary data volumes, the `vradmin migrate` command does not set the `primary_datavol` field on the new Primary after transferring the Primary role. To facilitates seamless transfer of the Primary role, make sure you set the `primary_datavol` field of the Primary data volumes, in addition to the Secondary. Note that you can use the `vradmin` command to perform all other VVR operations in a configuration containing differently named volumes on the Primary and Secondary.

There are two ways to set the `primary_datavol` field on a Secondary data volume. In the examples that follow, the commands are executed only on the Secondary. The Secondary data volume is called `secondaryname-dv_name`, and the corresponding Primary data volume name is `dv_name`.

To map the name of a Secondary data volume after it is associated to the RVG

- To set the name of the corresponding Primary data volume on a Secondary data volume after the volume has been associated to the Secondary RVG, use the `vxedit` command:

```
# vxedit -g diskgroup set primary_datavol=dv_name \
  secondaryname-dv_name
```

To map the name of a Secondary data volume when it is being associated to the RVG

1. To set the name of the Primary data volume on a corresponding Secondary data volume when it is being associated with the Secondary RVG, specify the `-m` option on the `vxvol` command line:

```
# vxvol -g diskgroup -m assoc rvg_name \
  secondaryname-dv_name dv_name
```

2. On the Secondary, display the *primary_datavol* field for a volume using `vxprint -l`:

```
# vxprint -g diskgroup -l secondaryname-dv_name
```

Output resembles:

```
Volume:                 secondaryname-vol03
assoc:                  rvg=rvg_name
                        plexes=secondaryname-vol03-01
                        primary_datavol=dv_name
```

Note: If any volume (on the Primary or a Secondary) is associated with an RVG (as an SRL or a data volume), the `vxprint -l` listing for that volume will indicate the RVG name on the output line beginning with `assoc:`, as shown above.

Mapping disk groups

If the RVGs on the Primary and Secondary are in differently named disk groups, the disk group mapping can be specified either when the RLINK is created, or later.

For example, if the disk group on the Primary is *dg1* and on the Secondary is *dg2*, then use the following commands to map the disk group during RLINK creation:

Primary:

```
# vxmake -g dg1 rlink rlink_name remote_dg=dg2
```

Secondary:

```
# vxmake -g dg2 rlink rlink_name remote_dg=dg1
```

If the disk groups were not properly mapped at the time of RLINK creation, the RLINK cannot be attached. This problem can be corrected as follows:

Primary:

```
# vxedit -g dg1 set remote_dg=dg2 rlink_name
```

Secondary:

```
# vxedit -g dg2 set remote_dg=dg1 rlink_name
```

Administering the SRL

The size of the SRL is critical to the performance of replication. When the SRL overflows for a particular Secondary, the Secondary becomes out of date until a complete resynchronization with the Primary is performed. Because resynchronization is a time-consuming process and during this time the data on the Secondary cannot be used, it is important to prevent the SRL from overflowing. Hence, when initially configuring VVR, determine an appropriate size for the SRL. The maximum size of the SRL can be derived from various criteria, however, the size of the SRL volume cannot be less than 110 MB. If the size that you have specified for the SRL is less than 110MB, VVR displays a message that prompts you to specify a value that is equal to or greater than 110 MB.

See “Sizing the SRL” on page 77.

It is possible that an SRL of an appropriate size overflows because of changes in the environment. This section describes how to protect from SRL overflows and administer VVR if the SRL overflows.

Protecting from SRL overflow

To avoid complete synchronization of Secondary in the case of an SRL overflow, VVR provides `autodcm` or `dcm` mode of SRL protection.

See “About the srlprot attribute” on page 91.

Before enabling SRL protection, each data volume in the RDS must have an associated DCM.

See “Associating a Data Change Map to a data volume” on page 205.

To enable SRL protection, change the replication setting for SRL protection.

See “Changing the replication settings for a Secondary” on page 144.

Incrementally synchronizing the Secondary after SRL overflow

The default protection mode for the SRL is `autodcm` and every volume in the RVG must have a DCM. When the SRL fills up, whether the RLINK is connected or not, DCM logging is activated and a bit corresponding to the region of the update is turned on for every incoming update. When you are ready to replay the DCM, start the DCM resynchronization process. To start the resynchronization, use the command `vradmin resync`. Note that you can also use the `cache` or `cachesize` parameters with the `vradmin resync` command. Specifying these attributes will cause the command to first create a space-optimized snapshot of the Secondary data volumes before starting the resynchronization.

Data is transmitted to the Secondaries only after all the RLINKs taking part in the resynchronization have connected. All the Secondaries taking part in the resynchronization must remain connected for the resynchronization to continue. The resynchronization will pause if any of the Secondary RLINK is paused.

During DCM resynchronization, VVR does not maintain the order of updates to the Secondary. As a result, the Secondary remains inconsistent until the resynchronization operation is complete. Note that if the Primary becomes unavailable during the time the resynchronization is taking place, the applications cannot be restarted on the Secondary.

If the Secondary volumes are mirrored, you can break off mirrors to retain consistent (though out-of-date) copies of data until the resynchronization is complete. However, to overcome this problem, create snapshots of the Secondary volumes before the resynchronization starts by using the following procedure.

To create snapshots and resynchronize the Secondary volumes

1 Create the cache object for the data volumes. This step is optional if you plan to use the `cachesize` attribute with `vradmin resync` command.

See "Preparing the RVG volumes for snapshot operation" on page 253.

2 To start the resynchronization use the command:

```
# vradmin -g diskgroup [-wait] resync local_rvgname \
   [cache=cacheobj | cachesize=size]
```

The *cache* attribute specifies a name for the precreated cache object, on which the snapshots for the volumes in the specified RVG will be created. The *cachesize* attribute specifies a default size for the cache object with respect to the source volume. You can specify only one of these attributes at one time with the `vradmin resync` to create one cache object for each snapshot

The parameters *cache* and *cachesize* are optional. If you do not specify either of these parameters then the `vradmin resync` command will resynchronize the Secondary volumes using the DCM replay, without creating the snapshots.

The *-wait* option can be used with the `vradmin resync` command to wait for the synchronization process to complete.

SRL overflow protection with DCM—flags and definitions

If the SRL Overflow Protection With DCM feature has been activated, VVR sets the following flag on the corresponding RLINK and its RVG:

Flag Value	Definition
`dcm_logging`	Log Overflow Protection With DCM has been started and the DCM is in use.

If the `dcm_logging` flag is set on an RLINK or RVG and neither the `resync_started` nor the `resync_paused` flag is set, the resynchronization (resync) has not been started. After the `vradmin resync` command has been issued, one or both of the following flags are set:

Flag Value	Definition
`resync_started`	Resynchronization is in progress, that is, data is being transferred from the Primary to the Secondary.
`resync_paused`	Resynchronization is paused.

Prerequisite for incrementally synchronizing the Secondary

The following item is a prerequisite for incrementally synchronizing the Secondary.

- The RVG must have the `dcm_logging` flag set.

To incrementally synchronize the Secondary

```
# vradmin -g diskgroup resync local_rvgname
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

Example:

```
# vradmin -g hrdg resync hr_rvg
```

To determine the progress of the incremental synchronization

◆ Determine the progress of the incremental synchronization after SRL overflow by issuing the following command on the Primary host:

```
# vxrlink -g diskgroup status rlink_name
```

The argument *`rlink_name`* is the name of the Primary RLINK to the Secondary.

The output shows how much data is left to send.

To monitor the progress of the incremental synchronization

- Monitor the progress of the incremental synchronization by issuing the *vxrlink -i interval status rlink_name* command. For example, to see the status every 5 seconds, issue the following command:

```
# vxrlink -g hrdg -i5 status rlink_name
```

The output resembles:

```
VxVM VVR vxrlink INFO V-5-1-4464 Rlink rlink_name is in AUTOSYNC.
100864K remaining.
VxVM VVR vxrlink INFO V-5-1-4464 Rlink rlink_name is in AUTOSYNC.
94464K remaining.
VxVM VVR vxrlink INFO V-5-1-4464 Rlink rlink_name is in AUTOSYNC.
76800K remaining.
```

Breaking off mirrors before incremental synchronization

During DCM resynchronization, the data volumes on the Secondary are inconsistent and cannot be used to take over the Primary role. To maintain a consistent copy of the data volumes on the Secondary, break off a mirror from each data volume

before starting DCM resynchronization. In the case of a disaster, these mirrors can be used to take over the Primary role. If you have FastResync license, make sure FR is set for all the volumes.

When snapshot plexes are available:

To find out if snapshot plexes are available on a data volume, use the `vxprint` command. The output shows the state of the plex as SNAPDONE. If a snapshot plex is available for each data volume, use the `vxrvg snapshot` command to take a snapshot of the data volumes in an RVG. If required, the snapshot volumes can be used to take over the Primary role. After the DCM resynchronization is complete, reattach the snapshot plexes back to the original volume using the `vxrvg snapback` command.

When snapshot plexes are not available:

If snapshot plexes are not available, detach mirrors on each of the data volumes on the Secondary using the `vxplex` command. After the DCM resynchronization is complete, reattach the plexes using the `vxplex att` command. To use the data on a detached plex in situations such as takeover, you must create a volume for the detached plex when snapshot plexes are not available.

Example - Resynchronizing the Secondary using break off mirrors

This example explains how to break off a mirror from a data volume and reattach a plex after the DCM resynchronization is complete. This example uses the volume `hr_dv01` that has two plexes `hr_dv01_01`and `hr_dv01_02`.

To resynchronize the Secondary using break off mirrors

1. On the Secondary, detach a plex from the data volume by typing:

   ```
   # vxplex -g hrdg det hr_dv01_02
   ```

2. When the RLINK reconnects, incrementally synchronize the Secondary by typing:

   ```
   # vradmin -g hrdg resync hr_rvg
   ```

   For multiple Secondary hosts, VVR simultaneously synchronizes all Secondary hosts that are operating in `dcm logging` mode.

3. After the DCM resynchronization is complete, reattach the plex to the data volume on the Secondary by typing:

   ```
   # vxplex -g hrdg att hr_dv01 hr_dv01_02
   ```

Example - Recreating volumes if a disaster occurs during resynchronization

If during the resynchronization process, a disaster occurs and the Secondary takes over, you can recreate the volumes as they were before the resynchronization started. The example uses the RVG `hr_rvg` and the volume `hr_dv01`.

See "Example - Resynchronizing the Secondary using break off mirrors" on page 221.

All steps are performed on the former Secondary, which is now the Primary.

See "About taking over from an original Primary" on page 308.

To recreate the volumes if a disaster occurs during resynchronization

1. Detach the Secondary RLINK.

```
# vxrlink -g hrdg det rsec
```

2. Dissociate the original data volume from the Secondary RVG.

```
# vxvol -g hrdg dis hr_dv01
```

3. Remove the original data volume.

```
# vxedit -g hrdg -rf rm hr_dv01
```

4. Create the volume for the detached plex by typing:

```
# vxmake -g hrdg -U usetype vol hr_dv01 plex=hr_dv01_02
```

   If a volume contains a file system, specify the *usetype* as `fsgen`; otherwise, specify `gen`.

5. Start the data volume by typing:

```
# vxvol -g hrdg -f start hr_dv01
```

6. Associate the data volume to its RVG.

```
# vxvol -g hrdg assoc hr_rvg hr_dv01
```

7. The volume is no longer mirrored. To add mirrors, issue the following command:

```
# vxassist -g hrdg mirror hr_dv01
```

Notes on using incremental synchronization on SRL overflow

Observe the following notes on using incremental synchronization on SRL overflow:

- Each data volume in the Primary RVG must have a DCM associated with it. You cannot use the SRL Overflow Protection With DCM feature unless every data volume in the RVG has a DCM. If any of the data volumes in the RVG do not a have a DCM, you cannot set `srlprot=dcm` or `srlprot=autodcm`. An attempt to associate a volume without a DCM to an RVG that has an RLINK with `srlprot=dcm` or `srlprot=autodcm` will also fail.
- If an RLINK is undergoing Automatic Synchronization and an attached RLINK with SRL Overflow Protection With DCM is about to overflow, the Automatic Synchronization is abandoned and SRL Overflow Protection With DCM for the overflowing RLINK becomes active.
- If an existing RLINK is using the DCM mechanism and another existing RLINK is about to overflow, the second RLINK is detached unless the DCM resynchronization for the first RLINK has not yet sent any writes. In this case, the outstanding writes of the first RLINK are also sent on the second RLINK.
- To remove a Secondary from a DCM resynchronization process, detach the corresponding Primary RLINK.
- If you try to dissociate a DCM from a data volume while the DCM is in use, the operation fails.
- If the DCM is detached because of I/O errors while the DCM is in use, the resynchronization is abandoned and all the RLINKs that are being synchronized are detached.

Changing the size of the SRL on the Primary and the Secondary

The size of the SRL must be large enough to meet specific sizing constraints. These constraints can change with the changes in the business needs, application write rate, available network bandwidth, and so on. As a result, it is necessary to determine the appropriate size of the SRL again.

See "Sizing the SRL" on page 77.

VVR enables you to increase the size of the Primary SRL and the Secondary SRL in a Replicated Data Set (RDS) using the `vradmin resizesrl` command, even while the application is active or while replication is in progress. The `vradmin resizesrl` command increases the size of the SRL in the RDS on the Primary, on any valid Secondaries, and on the bunker node, if present. A valid Secondary is one that is correctly configured; that is, it does not have configuration errors. Use the `vradmin -l printrvg` command to view the configuration status of the RDS.

The `vradmin resizesrl` command does not resize the SRL on any Secondary that has configuration errors.

Before increasing the size of the SRL, do the following:

- On each host in the RDS, check whether there is enough free space in the disk group in which the SRL resides by issuing the following command:

  ```
  # vxassist -g diskgroup free
  ```

  If any host does not have enough space to resize the SRL, the `resizesrl` command will fail.

  The `vxassist -g diskgroup free` command displays the amount of free space per disk. To display the amount of total free space in the disk group, issue the following command:

  ```
  # vxassist -g diskgroup maxsize
  ```

To increase the size of the SRL on the Primary and the Secondary

Issue the following command on any host in the RDS:

```
# vradmin -g diskgroup resizesrl [-f] local_rvgname length \
[pridiskname=primary_disk_names] [secdiskname=secondary_disk_names]
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`length`* is the desired size for the Primary SRL. The length can be specified using the standard VxVM conventions. It can be prefixed by a plus sign (+) to indicate an increase in the size of the Primary SRL by the specified amount.

Use the `-f` option for the `vradmin resizesrl` command to resize the Primary SRL even if the Secondary or Bunker hosts do not have enough space to increase the SRL. This option may be necessary to protect the Primary SRL from overflow. With the `-f` option, the command succeeds and resizes the Primary SRL provided that the Primary has enough space. The command also attempts to resize the Secondary SRL and the Bunker SRL, if present. However, if any Secondary or Bunker host does not have enough free space to increase the SRL size, the resize operation fails on that host.

Warning: Using the `-f` option may result in different SRL sizes on different hosts.

The optional arguments `pridiskname` and `secdiskname` enable you to specify a comma-separated list of disk names for the resize operation. The space required

for the resize operation is allocated from the specified disks on the primary and secondary, respectively.

Decreasing the size of the SRL on the Primary

Before resizing the SRL, do the following:

- Stop the application.
- Verify that the RLINKs are up-to-date by issuing the following command:

```
# vxrlink -g diskgroup status rlink_name
```

To decrease the size of the SRL on the Primary

1. If the application and the RVG are not configured as VCS resources, proceed to the next step.

   OR

   If the application and the RVG are configured as VCS resources, OFFLINE the application resource, as well as, the RVG resource and then proceed to step 4.

   To OFFLINE resources, use the `hagrp` command.

   For more information on bringing resources offline, see the *Cluster Server User's Guide*.

2. Make sure that the application is not running.
3. Stop the RVG:

   ```
   # vxrvg -g diskgroup stop rvg_name
   ```

4. Make sure all RLINKs are up-to-date:

   ```
   # vxrlink -g diskgroup status rlink_name
   ```

   Note: If you see any outstanding writes, do not proceed to step 5.

5. Detach all RLINKs:

   ```
   # vxrlink -g diskgroup det rlink_name
   ```

6 Dissociate the SRL from the RVG.

Note: Any Storage Checkpoints that you have created will be lost after dissociating the SRL.

```
# vxvol -g diskgroup dis srl_name
```

7 Decrease the size of the Primary SRL using the `vxassist` command. For example, to decrease the size of the SRL:

```
# vxassist -g diskgroup shrinkto srl_name new_length
```

Note: It is recommended that the SRL reside on disks that are not being used for the data volumes. Also, it is recommended that the Primary and Secondary SRLs must be of the same size. See the `vxassist(1M)` manual page for more information.

---

8 Reassociate the SRL with the RVG:

```
# vxvol -g diskgroup aslog rvg_name srl_name
```

9 Attach all RLINKs:

```
# vxrlink -f att rlink_name
```

Note: The RLINK was up-to-date when it was detached in step 5 and the Secondary is consistent with the Primary; therefore, it is appropriate to use the force option here.

---

10 If the application and the RVG are not configured as VCS resources, proceed to the next step.

OR

If the application and the RVG are configured as VCS resources, ONLINE the RVG resource, as well as, the application resource. To ONLINE resources, use the `hagrp` command. For more information on bringing resources online, see the *Cluster Server User's Guide*. The resize operation is complete.

Do not perform step 11 and step 12.

11 Start the RVG:

```
# vxrvg -g diskgroup start rvg_name
```

12 Restart the application.

Administering replication

You can control replication in an RDS by changing the replication settings. Administering replication also includes pausing and resuming replication.

Changing the replication settings

You can change the VVR replication attributes according to your requirements with the `vradmin set` command. The `vradmin set` command enables you to set the following VVR replication attributes:

- Replication Mode
- Latency protection
- SRL protection
- Network transport protocol
- Packet size
- Bandwidth limit

See "Changing the replication settings for a Secondary" on page 144.

Pausing and resuming replication to a Secondary

Pausing an RLINK prevents new and already-queued updates from reaching the Secondary from the Primary, and the Primary and Secondary do not communicate.

The `vradmin pauserep` command does not provide a way to pause a Secondary RLINK. To do this, use the `vxrlink` command on the Secondary host. The `vradmin resumerep` command resumes both types of pauses on the selected RLINKs.

Note: If the latency protection is set to `override`, be sure you understand the consequences of pausing the Secondary.

See "About Latency protection when Primary and Secondary are disconnected" on page 95.

To pause and resume replication to a Secondary

1. Pause replication by issuing the following command on any host in the RDS:

   ```
   # vradmin -g diskgroup pauserep local_rvgname [sec_hostname]
   ```

   where *local_rvgname* is the name of the RVG on the host where the command is issued, and *sec_hostname* is the name of the Secondary host to which replication is being paused. For an RDS with a single Secondary, you do not have to specify the Secondary hostname.

2. On the Primary, issue the `vxprint` command to check that the state of the RLINK is `PAUSE`.

   ```
   # vxprint rlink_name
   ```

3. Resume replication to the Secondary.

   ```
   # vradmin -g diskgroup resumerep local_rvgname [sec_hostname]
   ```

   where *local_rvgname* is the name of the RVG on the host where the command is issued, and *sec_hostname* is the name of the host to which replication is being resumed.

Stopping replication to a Secondary

The `vradmin stoprep` command can be used to stop replication to a Secondary in an RDS. The `vradmin stoprep` command can be entered from any host in the RDS.

The `vradmin stoprep` command fails if the Primary and Secondary RLINKs are not up-to-date. Use the `-f` option to stop replication to a Secondary even when the RLINKs are not up-to-date.

Before stopping replication, the `vradmin stoprep` command displays a warning and prompts the user to confirm whether or not to stop replication. To skip this confirmation, use the `-s` option with the `vradmin stoprep` command. The `-s` option to the `vradmin stoprep` command proves useful in scripts.

To stop replication to a specific Secondary in an RDS

◆ To stop replication to a specific Secondary, use the following command:

```
# vradmin -g diskgroup stoprep local_rvgname sec_hostname
```

The argument *local_rvgname* is the name of the RVG on the local host and represents its RDS.

The argument *sec_hostname* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command. For an RDS with a single Secondary, you do not have to specify the Secondary host name

Example:

To stop replication from the Primary RVG `hr_rvg` on `seattle` to the Secondary RVG on host `london`, type:

```
# vradmin -g hrdg stoprep hr_rvg london
```

Changing the IP addresses used for replication

You may need to change the host name or IP address of the Primary and Secondary used for replication if you move a Primary or Secondary to a new location or if you need to make the replication use a different network. You can change the host name or IP address even after replication has been established. The `vradmin changeip` command enables you to change the replication network between the Primary and a Secondary in an RDS.

Prerequisites for changing the IP addresses used for replication

Observe the following prerequisites for changing the IP addresses used for replication:

- The new host names must be configured for proper resolution at both the Primary and Secondary sites using the appropriate mechanisms such as DNS, NIS, or hosts. This means that each system must be configured to bring up their addresses on reboot, or if this is a cluster, the cluster monitor must bring up the proper address.
- The Secondary must be reachable from the Primary either through the previous network, the new network, or both the networks.
- If the previous network is no longer available, the `vradmin changeip` command must be run from the Primary host.

Note: The VVR heartbeat port can be changed using the `vrport` command. To ensure that the RLINKs pick up the new port, always run the `vradmin changeip` command (without passing the *`newpri`* and *`newsec`* arguments) after changing the port. Restart the `vxnetd` daemon on the required system for the changes to take effect.

To change the IP addresses used for replication

- Change the IP address for the Primary or Secondary host, or both, using the following command:

  ```
  # vradmin [-g diskgroup] changeip local_rvgname [sec_hostname] \
    [newpri=<new_pri_ip | hostname>] [newsec=<new_sec_ip | hostname>]
  ```

  The argument *`diskgroup`* is the name of the local disk group that contains the RVG.

  The *`local_rvgname`* argument is the name of the RVG on the host where the command is issued.

  The *`sec_hostname`* is the name of the Secondary to which the replication network is being changed. This argument must be specified if the RDS has more than one Secondary.

  The `newpri` attribute specifies a new hostname or IP address for the Primary host that is to be used to establish a network connection for the replication to the Secondary. This is the new value for the `local_host` attribute of the Primary RLINK and the `remote_host` attribute of the corresponding Secondary RLINK.

  The `newsec` attribute specifies a new hostname or IP address for the Secondary host that is to be used to establish a network connection for the replication. This is the new value for the `remote_host` attribute of the Primary RLINK and the `local_host` attribute of the Secondary RLINK.

Example for changing IP addresses to a different IPv4 network

This example shows how to change the network used for replication to a different IPv4 network. Table 8-3 shows the current configuration.

Table 8-3 Configuration before network change

Attribute	Value on Primary	Value on Secondary
local_host displayed in the output of the *vxprint -l rlink_name* command	seattle	london
remote_host	london	seattle
RVG	hr_rvg	hr_rvg
Disk Group	hrdg	hrdg
RLINK	rlk_london_hr_rvg	rlk_seattle_hr_rvg

Table 8-4 shows the configuration after making the changes to the replication network.

Table 8-4 Configuration after network change

Attribute	Value on Primary	Value on Secondary
local_host displayed in the output of the *vxprint -l rlink_name* command	seattle_hrnet	london_hrnet
remote_host	london_hrnet	seattle_hrnet
RVG	hr_rvg	hr_rvg
Disk Group	hrdg	hrdg
RLINK	rlk_london_hr_rvg	rlk_seattle_hr_rvg

To change the IP addresses used for replication

1. From the Primary host `seattle`, issue the following command:

   ```
   # vradmin -g hrdg changeip hr_rvg newpri=seattle_hrnet \
   newsec=london_hrnet
   ```

   The `vradmin changeip` command changes the IP address of both the Primary RLINK and the corresponding Secondary RLINK to the new addresses `newpri` and `newsec` (in this example, `seattle_hrnet` and `london_hrnet`, respectively).

2. To verify the change on the Primary RLINK, issue the following command on the Primary host:

   ```
   # vxprint -l rlk_london_hr_rvg
   ```

   Output includes the following fields:

   ```
   Disk group: hrdg
   .
   .
   Rlink: rlk_london_hr_rvg
   .
   .
   remote_host=london_hrnet IP_addr=x.x.x.x
   .
   .
   local_host=seattle_hrnet IP_addr=x.x.x.x
   .
   .
   ```

   where `x.x.x.x` represents the corresponding IP address.

3 To verify the change on the Secondary RLINK, issue the following command on the Secondary host:

```
# vxprint -l rlk_seattle_hr_rvg
```

Output includes the following fields:

```
Disk group: hrdg
.
.
Rlink: rlk_seattle_hr_rvg
.
.
remote_host=seattle_hrnet IP_addr=x.x.x.x
.
.
local_host=london_hrnet IP_addr=x.x.x.x
.
.
```

where x.x.x.x represents the corresponding IP address.

Example for changing IP addresses to a different IPv6 network

This example shows how to change the network used for replication to a different IPv6 network. Table 8-5 shows the current configuration.

Table 8-5 Configuration before network change

Attribute	Value on Primary	Value on Secondary
local_host displayed in the output of the *vxprint -l rlink_name* command	seattle	london
remote_host	london	seattle
RVG	hr_rvg	hr_rvg
Disk Group	hrdg	hrdg
RLINK	rlk_london_hr_rvg	rlk_seattle_hr_rvg

Table 8-6 shows the configuration after making the changes to the replication network.

Table 8-6 Configuration after network change

Attribute	Value on Primary	Value on Secondary
local_host displayed in the output of the *vxprint -l rlink_name* command	seattle-v6_hrnet	london-v6_hrnet
remote_host	london-v6_hrnet	seattle-v6_hrnet
RVG	hr_rvg	hr_rvg
Disk Group	hrdg	hrdg
RLINK	rlk_london_hr_rvg	rlk_seattle_hr_rvg

To change the IP addresses used for replication

1 From the Primary host `seattle`, issue the following command:

```
# vradmin -g hrdg changeip hr_rvg newpri=seattle-v6_hrnet \
  newsec=london-v6_hrnet
```

The `vradmin changeip` command changes the IP address of both the Primary RLINK and the corresponding Secondary RLINK to the new addresses `newpri` and `newsec` (in this example, `seattle-v6_hrnet` and `london-v6_hrnet`, respectively).

Note: In a cluster volume replication (CVR) environment the new IP address must be plumbed on the logowner.

---

2 To verify the change on the Primary RLINK, issue the following command on the Primary host:

```
# vxprint -l rlk_london_hr_rvg
```

Output includes the following fields:

```
Disk group: hrdg
.
.
Rlink: rlk_london_hr_rvg
.
.
remote_host=london-v6_hrnet \
IP_addr=aaaa:bbbb:cccc:dddd:eeee:xxxx:yyyy:zzzz
.
.
local_host=seattle-v6_hrnet \
IP_addr=aaaa:bbbb:cccc:dddd:eeee:xxxx:yyyy:zzzz
.
.
```

where *aaaa:bbbb:cccc:dddd:eeee:xxxx:yyyy:zzzz* represents the corresponding IPv6 address.

3 To verify the change on the Secondary RLINK, issue the following command on the Secondary host:

```
# vxprint -l rlk_seattle_hr_rvg
```

Output includes the following fields:

```
Disk group: hrdg
.
.
Rlink: rlk_seattle_hr_rvg
.
.
remote_host=seattle-v6_hrnet \
IP_addr=aaaa:bbbb:cccc:dddd:eeee:xxxx:yyyy:zzzz
.
.
local_host=london-v6_hrnet \
IP_addr=aaaa:bbbb:cccc:dddd:eeee:xxxx:yyyy:zzzz
.
.
```

where *aaaa:bbbb:cccc:dddd:eeee:xxxx:yyyy:zzzz* represents the corresponding IPv6 address.

Changing the network ports used for replication

VVR uses the UDP and TCP transport protocols to communicate between the Primary and Secondary. You may need to change the network port numbers from the default ports.

Port numbers used by VVR

This section lists the default ports used by VVR.

Table 8-7 shows the ports which VVR uses by default when replicating data using UDP.

Table 8-7 Default ports used by VVR for replicating data using UDP

Port Numbers	Description
UDP 4145	IANA approved port for heartbeat communication between the Primary and Secondary.

Table 8-7 Default ports used by VVR for replicating data using UDP *(continued)*

Port Numbers	Description
TCP 8199	IANA approved port for communication between the `vradmind` daemons on the Primary and the Secondary.
TCP 8989	Communication between the `in.vxrsyncd` daemons, which are used for differences-based synchronization.
UDP Anonymous ports (OS dependent)	Ports used by each RLINK for data replication between the Primary and the Secondary.

Table 8-8 shows the ports which VVR uses by default when replicating data using TCP.

Table 8-8 Default ports used by VVR for replicating data using TCP

Port Numbers	Description
UDP 4145	IANA approved port for heartbeat communication between the Primary and Secondary.
TCP 4145	IANA approved port for TCP Listener port.
TCP 8199	IANA approved port for communication between the `vradmind` daemons on the Primary and the Secondary.
TCP 8989	Communication between the `in.vxrsyncd` daemons, which are used for differences-based synchronization.
TCP Anonymous ports	Ports used by each RLINK for replication on the Primary.

Displaying and changing the ports used by VVR

Use the `vrport(1M)` command to display, change or set the port numbers used by VVR. You may have to change the port numbers in the following cases:

- To resolve a port number conflict with other applications.
- To configure VVR to work in your firewall environment.
- To configure VVR to work in your firewall environment when using UDP; to specify a restricted number of ports to replicate data between the Primary and the Secondary.

Displaying the port used for heartbeats

Use the `vrport heartbeat` command to display the port number used by VVR, for heartbeats. To change the heartbeat port number on a host, specify the port number with the `vrport heartbeat` command. Use the `vradmin changeip` command to update the RLINKs with the new port information, and then restart the `vxnetd` daemon on the required system for the changes to take effect.

To display the port number used for heartbeats

```
# vrport heartbeat
```

To change the port number used for heartbeats

```
# vrport heartbeat port
```

Example - Changing the replication heartbeat port

This example shows how to change the replication heartbeat port on the host seattle. Follow the same steps to change the heartbeat port on secondary (london).

Note: VVR supports a configuration with different heartbeat port numbers on the primary and secondary.

To change the replication heartbeat port on seattle from 4145 to 5000

1. Use the `vrport` command to change the heartbeat port to 5000 on the required host.

   ```
   # vrport heartbeat 5000
   ```

2. Issue the `vradmin changeip` command without the `newpri` and `newsec` attributes.

   ```
   # vradmin -g hrdg changeip hr_rvg london
   ```

3. Verify the changes to the local RLINK by issuing the following command on the required host:

   ```
   # vxprint -g hrdg -l rlk_london_hr_rvg
   ```

4 Stop the `vxnetd` daemon.

```
# /usr/sbin/vxnetd stop
```

5 Restart the `vxnetd` daemon.

```
# /usr/sbin/vxnetd
```

Displaying the port used by vradmind

To display the port number used by `vradmind`, use the `vrport vradmind` command. To change the `vradmind` port, specify the port number with the `vrport vradmind` command.

To display the port number used by vradmind

```
# vrport vradmind
```

To change the port number used by vradmind

```
# vrport vradmind port
```

Note: You must restart the server `vradmind` for this change to take effect. Make sure you change the port number on all the hosts in the RDS.

Displaying the port used by in.vxrsyncd

To display the port numbers used by `in.vxrsyncd`, use the `vrport vxrsyncd` command. To change the port numbers used by `in.vxrsyncd`, specify the port number with the `vrport vxrsyncd` command.

To display the port number used by in.vxrsyncd

```
# vrport vxrsyncd
```

To change the port number used by in.vxrsyncd

```
# vrport vxrsyncd port
```

Note: You must restart the server `in.vxrsyncd` for this change to take effect. Make sure you change the port number on all the hosts in the RDS.

Displaying the ports used to replicate data using UDP

To display the ports used to replicate data when using UDP, use the `vrport data` command. To change the ports used to replicate data when using UDP, specify the list of port numbers to use with the `vrport data` command.

Each RLINK requires one UDP port for replication. Specify an unused, reserved port number that is less than 32768 so that there is no port conflict with other applications. The number of ports specified must be equal to or greater than the number of RLINKs on the system.

To display ports used to replicate data when using UDP

```
# vrport data
```

To change ports used to replicate data when using UDP

For a system configured with one RLINK, use the following command:

```
# vrport data port
```

For a system configured with multiple RLINKs, you can specify either a range of port numbers or a list of port numbers or both.

To specify a range of port numbers, use the following command:

```
# vrport data port1, port2, portlow-porthigh, .....
```

For example:

```
# vrport data 3400, 3405, 3500-3503, 5756-5760
```

Note: To use the new port information, execute `/usr/sbin/vxnetd`, and then pause and resume all RLINKs.

Displaying the ports used to replicate data using TCP

A Primary RVG with one Secondary must have a minimum of one outgoing port. For high latency networks, /home/pubs/sfhadocs/tot/linux/autobuild/online_pdfs Veritas recommends that you have up to 16 outgoing ports per Primary RVG to allow for multiple connections. As VVR uses anonymous ports for TCP, the ports that are used for replication using TCP cannot be displayed.

Administering the Replicated Data Set

A Replicated Data Set consists of a Primary RVG and one or more Secondary RVGs. Administering the Replicated Data Set includes removing the Secondary RVG and removing the Primary RVG.

Removing a Secondary RVG from a Replicated Data Set

The `vradmin delsec` command removes a Secondary RVG from its RDS. The `vradmin delsec` command can be entered from any host in an RDS.

The `vradmin delsec` command removes the Secondary RVG from its RDS on the specified Secondary host. Before executing this command, you must stop replication to the specified Secondary, using the `vradmin stoprep` command.

Caution: The operation performed by the `vradmin delsec` command is irreversible.

The `vradmin delsec` command performs the following by default:

- Dissociates the data volumes and SRL from the Secondary RVG.
- Removes the Secondary RVG from its RDS, deletes the Secondary RVG, and deletes the associated Primary and Secondary RLINKs.

The `vradmin delsec` command does not delete data volumes and the SRL.

To remove a Secondary from an RDS

```
# vradmin -g diskgroup delsec local_rvgname sec_hostname
```

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`sec_hostname`* is the name of the Secondary host as displayed in the output of the vradmin printrvg command.

Example:

This example removes the Secondary RVG `hr_rvg` from its RDS. Table 8-9 shows the sample configuration.

Table 8-9 Sample configuration

Attribute	Value on Primary	Value on Secondary
Host Name displayed in the output of the vradmin printrvg command	`seattle`	`london`
RVG	`hr_rvg`	`hr_rvg`
Disk Group	`hrdg`	`hrdg`

Because the command is entered on the Primary `seattle`, the local RVG is the Primary RVG `hr_rvg` belonging to the disk group `hrdg`.

To remove the Secondary RVG `hr_rvg` from its RDS, type the following command on `seattle`:

```
# vradmin -g hrdg delsec hr_rvg london
```

Removing a Primary RVG from a Replicated Data Set

The `vradmin delpri` command removes a Primary RVG from an RDS and thus deletes the corresponding RDS.

Prerequisite for deleting a Primary RVG from a Replicated Data Set

Observe the following prerequisites for deleting a Primary RVG:

- All Secondaries in the RDS must be removed.
  See "Removing a Secondary RVG from a Replicated Data Set" on page 241.

The `vradmin delpri` command performs the following by default:

- Dissociates the data volumes and SRL from the Primary RVG.
- Removes the Primary RVG.

The `vradmin delpri` command does not delete data volumes or the SRL from the Veritas Volume Manager configuration.

Note: This command can only be issued from the Primary host.

To remove a Primary RVG

```
# vradmin -g diskgroup delpri rvg_name
```

The argument *`rvg_name`* is the name of the Primary RVG to be removed.

When used with the `-f` option, the `vradmin delpri` command removes the Primary RVG even when the application is running on the Primary.

The `delpri` command cleans RVG on both the sites, if VVR configuration is not in an error state. If it is in an error state, then the `delpri` command cleans up only the primary site, and you have to manually clean up the secondary site. To clean up RVG on the secondary site, execute the following steps for the secondary site:

Examples to clean up RVG on the secondary site

To clean up RVG on the secondary site:

1. Stop the RVG:

   ```
   vxrvg -g diskgroup stop rvg_name
   ```

2. Detach all the secondary Rlinks:

   ```
   vxrlink -g diskgroup det rlink_name
   ```

3. Disassociate VSet from RVG (if any):

   ```
   vxvset -g diskgroup dis vset_name
   ```

4. Disassociate SRL from RVG:

   ```
   vxvol -g diskgroup -r rvg_name dis srl_name
   ```

5. Disassociate data volumes from RVG:

   ```
   vxvol -g diskgroup -r rvg_name dis vol_name
   ```

6. Delete all the secondary Rlinks:

   ```
   vxedit -g diskgroup rm rlink_name
   ```

7. Delete RVG on the secondary site:

   ```
   vxedit -g diskgroup rm rvg_name
   ```

Example - Removing a Primary RVG from an RDS when the application is inactive

To remove the Primary RVG `hr_rvg` when the application using the Primary data volumes is inactive, issue the following command on the Primary host:

```
# vradmin -g hrdg delpri hr_rvg
```

Example - Removing a Primary RVG from an RDS when the application is active

To remove the Primary RVG `hr_rvg` when the application using the Primary data volumes is active, issue the following command on the Primary host:

```
# vradmin -g hrdg -f delpri hr_rvg
```

Administering Storage Checkpoints

A Storage Checkpoint is a user-defined marker in the SRL that can be used during the following tasks:

- synchronizing the Secondary when the application is active.
  See "Synchronizing the Secondary and starting replication" on page 152.

- restoring the Secondary data volumes from backup.
  See "Backing up the Secondary" on page 279.

See "Understanding Storage Checkpoints" on page 49.

Creating Storage Checkpoints

VVR enables you to create Primary and Secondary Storage Checkpoints. The Primary Storage Checkpoints are associated with an RVG. However, Secondary Storage Checkpoints are associated with an RLINK. VVR allows you to create a maximum of 46 Storage Checkpoints.

To create a Primary Storage Checkpoint

```
# vxrvg -c checkpt_name checkstart rvg_name
```

The argument *checkpt_name* is the name that you choose to specify for the Storage Checkpoint.

To create a Secondary Storage Checkpoint

```
# vxrlink -c checkpt_name pause rlink_name
```

The argument *checkpt_name* is the name that you choose to specify for the Storage Checkpoint.

Ending Storage Checkpoints

The end of the Storage Checkpoint, or checkend, marks the position in the SRL for the end of a process such as synchronization or backup. When you are ready to end the Storage Checkpoint, for example, when the backup is complete, end the Storage Checkpoint in the SRL.

To end a Primary Storage Checkpoint

```
# vxrvg -g diskgroup checkend rvg_name
```

To end a Secondary Storage Checkpoint

For a Secondary Storage Checkpoint, resume replication to the Primary.

```
# vxrlink -c checkpt_name resume rlink_name
```

The Storage Checkpoint will end when the resume operation starts.

Secondary Storage Checkpoints are used when you back up the Secondary.

See "Backing up the Secondary" on page 279.

Viewing Storage Checkpoints

Primary Storage Checkpoints are associated with an RVG. You can display the list of Primary Storage Checkpoints by using the `vxrvg cplist` command.

Secondary Storage Checkpoints on the other hand are associated with an RLINK. You can display the list of Secondary Storage Checkpoints by using the vxrlink `cplist command` on the Primary.

See "Displaying a list of Storage Checkpoints" on page 176.

Deleting Storage Checkpoints

After you have finished using the Storage Checkpoints, you can delete the Storage Checkpoints. VVR allows you to retain a maximum of 46 Storage Checkpoints. To create any new Storage Checkpoint, delete an earlier Storage Checkpoint that you no longer require.

To delete a Primary Storage Checkpoint

```
# vxrvg -g diskgroup -c checkpt_name checkdelete rvg_name
```

The argument *rvg_name* is the name of the Primary RVG for which the Storage Checkpoint is to be deleted.

The argument *checkpt_name* is the name of the specific Storage Checkpoint to be deleted.

To delete a Secondary Storage Checkpoint

```
# vxrlink -g diskgroup -c checkpt_name checkdelete rlink_name
```

Note that this command must be run only on the Primary.

The argument *rlink_name* is the name of the RLINK for which the Storage Checkpoint is to be deleted.

The argument *checkpt_name* is the name of the specific Storage Checkpoint to be deleted.

Creating RVG snapshots

VVR enables you to create snapshots that are images of the online data volumes at a given point in time. The data in the original volume may change; however, the snapshot can still be used as a stable and independent copy for various purposes. VVR provides two methods of creating snapshots: instant snapshots and traditional snapshots.

Note: If the Secondary RVG is inconsistent, then VVR does not allow you to create snapshots of the volumes under this RVG.

The instant snapshot feature is a separately licensed feature of VVR. The advantages of this method over the traditional snapshot method are that the snapshots are available immediately, and they may be space-optimized, thus requiring less space than the traditional snapshots.

See "Using the instant snapshot feature" on page 246.

With the traditional snapshot method, depending on the size of the volume, the time required for initial synchronization of the plexes can be very large.

See "Using the traditional snapshot feature" on page 265.

After a volume has been prepared for the instant snapshot feature, you cannot use it to take a snapshot using the traditional snapshot method. To use the traditional snapshot method, you must first unprepare the volume. Thus, you cannot use the same volume for both the traditional method and the instant snapshot method at the same time.

If an RVG contains a volume set, the `vxrvg snapshot` command can be used to take snapshots of its data volumes.

Using the instant snapshot feature

VVR enables you to create instant snapshots using the `vxrvg snapshot` command. This command takes snapshots of the data volumes in the RVG. However, the snapshot volumes are not part of the RVG. Each data volume in an RVG can have more than one snapshot volume. When creating full instant or space-optimized snapshots, the snapshot volumes do not need to be synchronized beforehand, therefore the snapshots are available instantly. The snapshot volumes are then synchronized later in the background.

Snapshot naming conventions

The snapshot volumes must be created using the correct naming convention, that is, *<prefix>-dv_name*. You can create snapshot volumes with appropriate prefixes using the `-P` option. However, you must ensure that this prefix matches the prefix that you specified for the data volumes.

For example, if you specify the prefix as *month*, the name of each snapshot data volume will start with the prefix *month*; that is, it will be named as *month-dv_name*. Thus, a data volume *hr_dv01* can have snapshot volumes such as *june-hr_dv01*, *july-hr_dv01*.

Note: We recommend that you create snapshots with prefixes using the `-P` option so that the snapshots can be easily identified for restoring the data. However, if you do not specify any prefix, the default prefix *`SNAP`* will be used.

The instant snapshot feature provides the following methods to create instant snapshots:

- See "About instant full snapshots" on page 247.
- See "About instant space-optimized snapshots" on page 252.
- See "About instant plex-breakoff snapshots" on page 257.

About instant full snapshots

The `vxrvg -F snapshot` command enables you to create an instant full snapshot of all the volumes in the RVG, at a single point in time. The snapshot is available for use immediately, because the snapshot volumes do not have to be completely synchronized, in the beginning. The snapshots volumes are synchronized later in the background.

The `vxrvg snapshot` command creates the data volume snapshots for all the volumes in the RVG, similar to the ones created by the `vxsnap make` command.

Note: You must create and prepare the snapshot volumes before they can be used for creating snapshots.

Prerequisites for creating instant full snapshots

Observe the following prerequisites:

- Make sure you create the snapshot volumes and then prepare them before creating the snapshots.
- Make sure the snapshot volumes are the same size as the original volumes.
- Make sure the snapshot volumes follow a proper naming convention such that the snapshot volume names can be easily linked to the original volumes.
  See "Snapshot naming conventions" on page 246.

The steps required to create instant full snapshots are as follows:

- Creating Snapshot Volumes for Data Volumes in an RVG
- Preparing the Volume
- Freezing or Pausing Replication

- Taking a Snapshot
- Unfreezing or Resuming Replication

Creating snapshot volumes for data volumes in an RVG

You must create snapshot volumes for the data volumes in an RVG before you take an instant full snapshot, because the `vxrvg snapshot` command does not create the snapshot volumes. Use the `vxassist make` commands or other Volume Manager commands to create the required volumes.

For more information on creating the volumes, refer to the *Storage Foundation Administrator's Guide*.

Preparing the volumes prior to using the instant snapshot feature

Before using the instant snapshot feature, prepare the volumes by performing the following tasks in the order shown:

- Upgrading the disk group for an instant snapshot
- Preparing volumes for an instant snapshot

Upgrading the disk group for an instant snapshot

To use the instant snapshot feature, the disk groups must be version 110 or above. If you are using disk groups created with an earlier disk group version, the first step in preparing the volumes is to upgrade the disk groups.

To explicitly upgrade the disk groups, run the `vxdg upgrade` command.

For details on upgrading disk groups, refer to the *Storage Foundation Administrator's Guide*.

Preparing volumes for an instant snapshot

Use the following command to prepare the volumes for instant snapshots:

```
# vxsnap -g diskgroup prepare volume [region=size] \
   [ndcomirs=number] [storage_attribute...]
```

Note: Run this command once on every data volume in the RVG.

After running the command the volumes are ready for the instant snapshot operations. Use the following command to verify whether the volumes are prepared. This command checks the settings of the *`instant`* flag for the required volumes:

```
# vxprint -g diskgroup -F%instant <volume>
```

For more information on the `vxsnap prepare` command, refer to the *Storage Foundation Administrator's Guide*.

Freezing or pausing replication prior to taking a snapshot

Before taking the snapshot on the Secondary, make sure the data volumes are consistent at the application level by either freezing or pausing replication. To make the data volumes consistent at the application level, use the IBC messaging utility `vxibc`.

See "About using VVR for off-host processing" on page 283.

For a failed Primary, you can pause the Primary RLINK, and then take a snapshot of the RVG. If you do not use `vxibc`, you pause the RLINK before taking the snapshot.

VVR provides you with sample scripts that can be used to freeze the replication before creating instant snapshots. When you install VVR, these scripts are installed in the following directory:

```
/etc/vx/vvr/ibc_scripts/sample_so_snapshot
```

Refer to the README file in this directory for instructions on how to use the sample scripts to create the instant snapshots.

Creating instant full snapshots

Use the following command to create an instant full snapshot for each data volume in an RVG:

```
# vxrvg -g diskgroup [-P prefix] -F snapshot rvg_name \
    [instantso=volume_list {cache=cachename|cachesize=size}] \
    [plexbreakoff=volume_list [plexprefix=plex_prefix]]\
    [exclude=volume_list] [syncing=yes|no] [comment="<comment>"]
```

Use the `vxrvg snapshot` command with its different attributes to specify the type of snapshot that you want to create. The `-F` option specifies instant full snapshots. By default, all the volumes in the RVG are considered for the instant full snapshots. To exclude any of the volumes from being considered for the instant full snapshots, use one of the following attributes. Depending on the attribute specified with the `vxrvg snapshot` command, the appropriate snapshots of the volumes are created.

The attribute `instantfull` need not be specified when the `-F` option is specified. This option can be used only if the `-F` option has not been specified and you still require a full instant snapshot of some of the volumes.

The attribute `exclude` specifies a comma-separated list of volumes that do not need to be considered for any kind of snapshots.

The attribute `instantso` specifies a comma-separated list of volumes that can be included for instant space-optimized snapshots when taking the instant full snapshots.

The attribute `syncing` specifies whether you want to start synchronizing the volumes in the background. By default, the value for this attribute is *`yes`*. The synchronization process can be started or stopped as required.

See "Synchronizing volumes on the local host and remote hosts" on page 199.

By default, when using the `vxrvg snapshot` command with the `-F` option you need not specify the volume list since all the volumes are included. However, you can also specify the volume list for the `instantso`, `plexbreakoff` and the `exclude` attribute when using the `-F` option. This will result in some volumes having an instant full snapshot, some having an instant space-optimized snapshot and some of them being excluded.

Any volumes in the RVG that are not specified in the *`volume_lists`* of the attributes `exclude`, `plexbreakoff`, or `instantso` will be snapped in the same way as the specified snapshot type, which is the instant full snapshot. Note that it is not possible to create two different types of snapshots for the same volume. The snapshot operation may result in some volumes with no snapshots and some with one of the three types of snapshots.

You can also reattach the snapshots to the data volumes in the RVG using the `vxrvg snapback` command.

See "Reattaching the snapshot plexes to the data volumes (snapback)" on page 260.

Example - Specifying a prefix for snapshots of each data volume in an RVG

To specify a prefix for the snapshots of each data volume in an RVG, use the following command. Make sure the snapshot volumes have been created in advance.

```
# vxrvg -g diskgroup -P june -F snapshot rvg_name
```

A snapshot data volume with the name *`june-dv_name`* is created for each data volume in the RVG. You can have more than one snapshot of the data volumes in an RVG.

Example - Creating an instant full snpashot for an RVG

This example shows how to create an instant full snapshot for an RVG.

To create an instant full snapshot for an RVG

1. Find the size of the original volumes for which you want to create snapshots on the hosts `seattle` or `london`.

   ```
   # vxprint -g hrdg -F"%name %len" hr_dv01 hr_dv02
   ```

   OR

   ```
   # vxprint -g hrdg -F"%name %len" 'vxrvg -g hrdg getdatavols hr_rvg'
   ```

2. Prepare the volumes for the instant snapshot operations, by using the following command on each data volume in the RVG for which you intend to create snapshots:

   ```
   # vxsnap -g hrdg prepare hr_dv01
   # vxsnap -g hrdg prepare hr_dv02
   ```

   Note: Make sure that all the applications running on these volumes are closed.

3. Create the snapshot volumes of the same size (or greater) as the original volumes and with an appropriate prefix.

   ```
   # vxassist -g hrdg make JUNE-hr_dv01 20971520
   # vxassist -g hrdg make JUNE-hr_dv02 20971520
   ```

4. Prepare the snapshot volumes:

   ```
   # vxsnap -g hrdg prepare JUNE-hr_dv01
   # vxsnap -g hrdg prepare JUNE-hr_dv02
   ```

5. Do one of the following:

   Pause replication to the Secondary.

   See “Pausing and resuming replication to a Secondary” on page 227.

   Freeze replication to the Secondary.

   See “About the IBC messaging utility vxibc” on page 477.

6 Create a snapshot of the required volumes in the RVG:

```
# vxrvg -g hrdg -F -P JUNE snapshot hr_rvg
```

This command creates a snapshot of all the volumes in the RVG with a prefix JUNE.

7 Resume or unfreeze replication on the Secondary, depending on the action you took in step 5.

If you paused replication, resume replication.

See "Pausing and resuming replication to a Secondary" on page 227.

If you used IBC messaging to freeze replication, unfreeze replication.

See "About the IBC messaging utility vxibc" on page 477.

Unfreezing or resuming replication after taking a snapshot

After taking a snapshot, unfreeze replication if you are using IBC messaging; otherwise, if you have paused replication, resume it. The snapshots are now ready for use.

About instant space-optimized snapshots

The `vxrvg -S snapshot` command creates an instant space-optimized snapshot of all the volumes in the RVG at a single point in time. The `vxrvg snapshot` command creates the same type of snapshots as the `vxsnap make` command and uses a cache object that functions as a space-optimized persistent store. The space required by the space-optimized snapshots is less than that of the original volume because space-optimized snapshots store only the changed data. The data change between the source volume and the snapshot usually is minimal during the lifetime of the snapshot.

If the size of the cache object is not enough for the incoming writes, the cache object can grow in size automatically, provided that the *autogrow* attribute has been set to `on`.

The values of the *highwatermark*, *autogrowby* and *maxautogrow* attributes can be set when a cache object is created using `vxmake`. If necessary, you can use the `vxcache set` command to change the values of these attributes for an existing cache. The default value for these attributes is as follows:

autogrow	Default is `off`.
autogrowby	Default value is 20% of the size of the cache volume in blocks.

highwatermark	Default is 90% of the size of the cache volume in blocks.
maxautogrow	Default is twice the size of the cache volume in blocks.

When the cache volume that is used by the snapshot reaches the preset *highwatermark* value, the Veritas Volume Manager cache daemon, vxcached, is invoked. The values of the *highwatermark*, *autogrowby* and *maxautogrow* attributes for the cache object determine the behavior of vxcached daemon.

- If the cache usage reaches the *highwatermark* value and the new required cache size cannot exceed the value of *maxautogrow* then vxcached grows the size of the cache volume by the size *autogrowby*.
- When cache usage reaches the *highwatermark* value, and the value of the new cache that needs to be created exceeds the value of *maxautogrow*, then vxcached deletes the oldest snapshot in the cache. If there are several snapshots that have been created during the same time period, then the largest of these is deleted.
- If the autogrow feature has been disabled for the cache object and the cache usage reaches the *highwatermark* value, then vxcached deletes the oldest snapshot in the cache. If there are several snapshots that have been created during the same time period, the largest of these is deleted. If there is only a single snapshot, the snapshot is detached and marked as invalid.

For more information on the vxcached daemon or the attributes of the autogrow parameter, refer to the *Storage Foundation Administrator's Guide*.

The vxrvg snapshot command also enables you to specify the size for the cache using the *cachesize* parameter. In this case, a separate cache object is created for every space-optimized snapshot.

The steps required to create space-optimized snapshots are as follows:

- Preparing the RVG volumes for snapshot operation
- Creating the cache object
- Freezing or Pausing Replication
- Taking a space-optimized snapshot
- Unfreezing or Resuming Replication

Preparing the RVG volumes for snapshot operation

Prior to creating an instant space-optimized snapshot, you must prepare the volumes under the RVG for the snapshot operation.

See "Preparing the volumes prior to using the instant snapshot feature" on page 248.

Creating the cache object for instant space-optimized snapshots

If you intend to create instant space-optimized snapshots then you must create the cache object within the same disk group as the data volumes. Use the `vxassist make` command to create the cache volume. After creating the cache volume, create the cache object using the `vxmake cache` command. This command allows you to set the `autogrow` option for the cache object which allows the cache object to grow automatically, if the size of the cache object is not enough for the incoming writes.

For example, to create the cache volume of size 1GB with a name *`cache-vol`* and with a mirrored layout, type the following command on `seattle`:

```
# vxassist -g hrdg make cache-vol 1g layout=mirror init=active
```

Now, you can create a cache object named *`cache-obj`* for the cache volume by typing the following commands on `seattle`:

```
# vxmake -g hrdg cache cache-obj cachevolname=cache-vol \
  autogrow=on regionsize=128

# vxcache -g hrdg start cache-obj
```

However, you can also create the cache object by specifying a value for the *`cachesize`* parameter in the `vxrvg snapshot` command. This command creates one cache object for every space-optimized snapshot. To create one cache object for all the space-optimized snapshots, you must create the cache object using the `vxassist make` command.

Freezing or pausing replication prior to creating an instant space-optimized snapshot

You must freeze or pause replication prior to creating an instant space-optimized snapshot.

See "Freezing or pausing replication prior to taking a snapshot" on page 249.

Creating instant space-optimized snapshots

To create a space-optimized snapshot for each data volume in an RVG, use the following command:

```
# vxrvg -g diskgroup [-P prefix] -S snapshot rvg_name \
    [instantfull=volume_list [syncing=yes|no]] \
```

```
[exclude=volume_list] [plexbreakoff=volume_list] \
[plexprefix=plex_prefix]] {cache=cachename|cachesize=size} \
[comment="<comment>"]
```

Use the `vxrvg snapshot` command with its attributes to specify the type of snapshot that you want to create. By default, all the volumes in the RVG are considered for the space-optimized snapshots. To exclude any of the volumes from being considered for the space-optimized snapshots, use one of the following attributes. Depending on the attribute specified with the `vxrvg snapshot` command, appropriate snapshots of the volumes are created.

The attribute `instantso` need not be specified when the `-S` option is specified.

The attribute `instantfull` specifies a comma-separated list of volumes that need to be included when creating an instant full snapshot of the volumes in an RVG.

The attribute `exclude` specifies a comma-separated list of volumes that do not need to be considered for any kind of snapshots.

The attribute `cache` specifies a name for the cache object. However, even if you do not specify a name, you can still specify a size for the cache. The `cachesize` attribute specifies a default size for the cache object with respect to the source volume. These operations together create one cache object per snapshot volume.

You can specify the volume list for the attributes `instantfull`, `exclude` or `plexbreakoff` when creating the instant space-optimized snapshots. This results in some volumes having an instant full snapshot, some with an instant space-optimized snapshot, some of them with instant plex-breakoff and some being excluded. Any volumes in the RVG that are not specified in the *`volume_lists`* of the attributes `exclude`, `plexbreakoff`, or `instantfull` will be snapped in the same way as the specified snapshot type, which is the instant space-optimized snapshot.

Example: Creating instant space-optimized snapshots

This example describes the steps to create an instant space-optimized snapshot for the specified RVG:

To create a space-optimized snapshot

1 Prepare the required volumes if the volumes have not been prepared already.

```
# vxsnap -g hrdg prepare hr_dv01
# vxsnap -g hrdg prepare hr_dv02
```

Perform this operation for all the data volumes in the RVG for which you intend to create snapshots.

2 You can create the cache volume and the cache object if you want to create all the space-optimized snapshots on a single cache object.

See "Creating the cache object for instant space-optimized snapshots" on page 254.

However, if you want to create separate cache objects for each snapshot proceed to the next step. You can create the cache object for each snapshot by specifying the *cachesize* or *cache* parameter.

3 Follow one of the steps provided depending on the method you have chosen for the cache object creation.

- To create the space-optimized snapshot for the volumes with a precreated cache object, issue the command:

```
# vxrvg -g hrdg -S -P SO snapshot hr_rvg cache=snap-cacheobj
```

- To create the space-optimized snapshot for the volumes with a separate cache object for each volume, issue the command:

```
# vxrvg -g hrdg -S -P SO1 snapshot hr_rvg cachesize=10%
```

The cache objects are created for each snapshot with cache volumes that are 10% of the source volume. You can also specify an absolute value for the *cachesize* parameter.

Note: If the size of the cache volume is less than 5MB, this command will fail.

Unfreezing or resuming replication after taking an instant space-optimized snapshot

After taking a snapshot, unfreeze replication if you are using IBC messaging; otherwise, if you have paused replication, resume it. The snapshots are now ready for use.

About instant plex-breakoff snapshots

The `vxrvg snapshot` command creates instant plex-breakoff snapshots of all the volumes in the RVG at a single point in time.

The steps required to create plex-breakoff snapshots are as follows:

- Preparing the RVG volumes for snapshot operation
- Creating snapshot plexes for Data Volumes in an RVG
- Freezing or Pausing Replication
- Creating an instant plex-breakoff snapshot
- Unfreezing or Resuming Replication

Preparing the RVG volumes for snapshot operation

It is necessary to prepare the volumes under the RVG for the snapshot operation.

See "Preparing the volumes prior to using the instant snapshot feature" on page 248.

Creating snapshot plexes for data volumes in an RVG

You must create plexes for the required volumes before you take an instant plex-breakoff snapshot.

Use the `vxsnap addmir` command to add one or more plexes to a volume:

```
# vxsnap -g diskgroup [-b] addmir volume [nmirror=<N>] \
            [attributes...]
```

Note: Run this command on every data volume in the RVG that needs the plex-breakoff snapshot to be created.

For more information on creating the plexes, refer to the *Storage Foundation Administrator's Guide*.

Freezing or pausing replication prior to creating a plex-breakoff snapshot

You must freeze or pause replication prior to creating a plex-breakoff snapshot.

See "Freezing or pausing replication prior to taking a snapshot" on page 249.

Creating instant plex-breakoff snapshots

The instant plex-breakoff snapshot feature enables you to create plex-breakoff snapshots just like the traditional snapshot feature.

Prerequisites for creating instant plex-breakoff snapshots

Observe the following prerequisites:

- Make sure the volumes for which you want to create plex-breakoff snapshots already have the appropriate plexes created and are in an SNAPDONE state.
- Make sure you create the plexes using appropriate prefixes in case you want to use specific plexes for the snapshot operation.
  For example, `<plexprefix>-<volume_name>`

Note: If you do not specify the `plexprefix` attribute when creating the plex-breakoff snapshots, a plex that is in the SNAPDONE state gets selected, automatically.

To create a plex-breakoff snapshot of each data volume in an RVG, use the following command:

```
# vxrvg -g diskgroup [-P prefix] snapshot rvg_name \
    [instantfull=volume_list [syncing=yes|no]] \
    [instantso=volume_list {cache=cachename|cachesize=size}] \
    [exclude=volume_list] [plexprefix=plex_prefix] \
    [comment="<comment>"]
```

Use the `vxrvg snapshot` command with its attributes to specify the type of snapshot that you want to create. This is the default if neither the `-S` nor the `-F` option is specified. By default, all the volumes will be included for instant plex-breakoff snapshots, provided that they have the plex volumes appropriately created. To exclude any of the volumes, use one of the following attributes. Depending on the attribute specified with the `vxrvg snapshot` command, appropriate snapshots of the volumes are created.

The attribute `exclude` specifies a comma-separated list of volumes that do not need to be considered for any kind of snapshot.

The `plexprefix` attribute specifies a prefix for the plexes that will be used for creating the plex-breakoff snapshots. This is allowed only if the `-F` or `-S` option is not specified or if you have specified a list of volumes for creating plex-breakoff volumes with the `vxrvg snapshot` command.

Example - Creating an instant plex-breakoff snapshot

This example describes the steps to create an instant plex-breakoff snapshot for an RVG:

To create an instant plex-breakoff snapshot

1. Prepare the required volumes if they have not been prepared already.

   ```
   # vxsnap -g hrdg prepare hr_dv01
   ```

2. If the volumes for which the plex-breakoff snapshots need to be created do not have the required plexes, create them using the following command:

   ```
   # vxsnap -g hrdg addmir hr_dv01
   ```

   Repeat this step for all the required data volumes in the RVG. Initial synchronizing of the plexes may take some time depending on the size of the volume.

   If you need to use specific plexes during the snapshot operation, make sure you name them appropriately when creating them. However, you can also do it later using the following command:

   ```
   # vxedit -g hrdg rename hr_dv01-02 snapplex-dv01
   ```

3. Use the following command to create snapshots using the specific plex prefixes:

   ```
   # vxrvg -g hrdg -P JULY snapshot hr_rvg plexprefix=snapplex
   ```

   Use the following command to create the snapshots without specifying the plex prefix:

   ```
   # vxrvg -g hrdg -P JULY1 snapshot hr_rvg
   ```

Unfreezing or resuming replication after taking a plex-breakoff snapshot

After taking a snapshot, unfreeze replication if you are using IBC messaging; otherwise, if you have paused replication, resume it. The snapshots are now ready for use.

Administering snapshots

VVR enables you to perform tasks such as refreshing snapshots, reattaching the snapshots to the plexes, and displaying the snapshots.

Refreshing snapshots

The `vxrvg snaprefresh` command allows you to refresh the snapshots of the volumes in an RVG. It creates a new point-in-time image of the volume, and the snapshots are immediately available for usre with their new contents. For example, a snapshot taken on Monday can be refreshed on Tuesday. Before refreshing the snapshot make sure the data volumes are consistent at the application level by either freezing or pausing replication. After refreshing the snapshot, unfreeze replication if you are using IBC messaging; otherwise, if you have paused replication, resume it.

Note: If the Secondary RVG is inconsistent, then VVR does not allow you to refresh the snapshots using the volumes under this RVG.

Use the following command to refresh existing snapshots:

```
# vxrvg -g diskgroup [-P <prefix>] snaprefresh rvg_name
```

The `-P` option can be used to select the snapshot volume names that begin with the specified prefix for the refresh operation. The prefix "SNAP" is used if no prefix is specified.

Note: After refreshing the snapshots you must issue the command `vxsnap syncstart` to start synchronizing the instant full snapshots. This is not required for instant space-optimized snapshots.

Reattaching the snapshot plexes to the data volumes (snapback)

The `snapback` operation reattaches the snapshots of the instant full snapshot volume or the plexes of the plex-breakoff snapshot volume back to the original volume. After working with the snapshot volumes, you can reattach the plexes to the data volumes in the RVG using the `snapback` operation. The `snapback` operation is instant as the plexes are resynchronized in the background.

Note: The `snapback` operation can be performed only on instant full snapshots and plex-breakoff snapshots but not on space-optimized snapshots.

The `vxrvg snapback` command snaps back the snapshot, that is, reattaches the snapshot plexes to their respective data volumes in the RVG.

You can use the default action of the `vxrvg snapback` command if the data volumes in the RVG have only one snapshot. If the data volumes have more than one snapshot plex, use the `-a` option with the `vxrvg snapback` command to reattach all snapshots; the `-a` option snaps back all the plexes to their original data volumes.

Note that if the plexes had been added using the `vxsnap addmir` command, then `vxrvg snapback` command will reattach the plexes in the SNAPDONE state. Otherwise, it will reattach the plexes in the active state.

For example, use the `-a` option to reattach the snapshot volumes *june-dv_name* and *july-dv_name* to each data volume *dv_name* in the RVG.

The `-P` option when used with the `vxrvg snapback` command enables you to reattach a specific set of snapshots that have been identified by the prefix. To snapback all the data volumes with the prefix *month* in their names, specify the prefix month using the `-P` option.

To snapback a single snapshot plex to the data volume in an RVG, use the following command:

```
# vxrvg -g diskgroup -P prefix snapback rvg_name
```

To snapback the plexes from all snapshots, of the volume in an RVG, use the following command:

```
# vxrvg -g diskgroup -a snapback rvg_name
```

All the snapshot plexes are snapped back to their original data volumes in the RVG *rvg_name*.

To snapback the snapshot plexes with a specific prefix for all data volumes in an RVG, use the following command:

```
# vxrvg -g diskgroup -P june snapback rvg_name
```

Snapshot plexes with the name *june-dv_name* are snapped back to their original data volumes in the RVG.

Restoring data from the snapshots

Use the `vxrvg snaprestore` command to restore the data from the required snapshots.

Note: When restoring the volumes you must specify the exact snapshot corresponding to the respective volumes that need to be restored. Therefore, it is recommended that you create snapshots with prefixes using the `-P` option so that they can be easily restored. However, if you do not specify any prefix, the default prefix *`SNAP`* will be used. The `vxrvg snaprestore` command can be used only for restoring the data from the instant snapshots.

```
# vxrvg -g diskgroup [-P prefix] snaprestore rvg_name
```

The restore operation is very useful when the data in a volume is corrupted, for example, because of a software or user error, and must be restored to a known state preserved in a snapshot of that volume taken some time earlier. Because a replicated volume under the RVG propagates all changes to the Secondary, the Secondary volume must be restored back to a known state. VVR can now do this automatically using the instant snapshots. It uses the DCM logging to resynchronize the Secondary so that only the regions changed as part of the restore operation are applied to the Secondary volumes.

If there are multiple RLINKs in an RVG, then VVR synchronizes all the Secondary hosts using the bits on the DCM log. If one of the RLINKs is already in a DCM mode, then VVR also updates the bits corresponding to the regions that need to be restored as a part of the restore operation to the DCM. Now, VVR resynchronizes all the Secondary hosts using the consolidated bits on the DCM log.

Note: In case of the multiple RLINK setup, if either the autosync or resync operation was already in progress across some RLINK in the RVG, then the resynchronization for the other RLINKs that have already switched to the DCM mode as a part of the restore operation starts automatically.

The `vxrvg snaprestore` command can be used to restore data both from the Primary and the Secondary. On the Primary the `vxrvg snaprestore` command populates the DCM for replay that can be used on the Secondary only if it has no attached RLINKS or if the RLINK is in the fail state.

See "Rules for restoring volumes" on page 262.

Note: Restoring data on the Primary RVG volumes using the `vxrvg snaprestore` command deletes all the existing Storage Checkpoints.

Rules for restoring volumes

The volumes in an RVG can be restored only according to the rules mentioned below. This is irrespective of whether the volumes are restored using the `vxrvg`

`snaprestore` command or specific volumes under the RVG are restored using the `vxsnap restore` command.

On Primary

- If the RLINK is detached, the volume is restored like any other VxVM volume.
- If the RLINK is active, the RLINK is put into DCM logging mode and the regions that need to be modified by the restore operation are marked on the DCM, and the volume is restored. The RVG must be resynchronized using `vxrvg resync` command, to ensure that the restored data is available on the Secondary RVG. This is independent of the SRL protection setting and works even if the `srlprot` attribute is not set to *`dcm`* or *`autodcm`*.
- If the RLINK is active and the volumes are not DCM logging enabled, then the restore operation fails, unless the `-f` (force) option is specified. If the force option is specified, the RLINK is detached before the volume is restored.

On Secondary

The restore operation is allowed only if:

- the RLINKs are detached.
- the attached RLINK in the RVG is in the FAIL state.

To restore the snapshots on the Primary

`On Primary:`

1. To stop the specific RVG use the following command:

   ```
   # vxrvg -g hrdg stop hr_rvg
   ```

2. To restore the volumes from snapshot with a specific prefix, use the following command:

   ```
   # vxrvg -g hrdg -P JULY snaprestore hr_rvg
   ```

   The RLINK changes to the DCM mode if it is not already in this mode.

3. To replicate the new changes to the Secondary, use the following command:

   ```
   # vxrvg -g hrdg resync hr_rvg
   ```

4. Start the RVG, as it may remain in the DISABLED state after restoring the volumes from the snapshot:

   ```
   # vxrvg -g hrdg start hr_rvg
   ```

Displaying the snapshot information

The `vxrvg snapprint` command displays information on the relationship that exists between the original volumes and the corresponding snapshots. To display information on the snapshots use the following command:

```
# vxrvg -g diskgroup snapprint rvg_name
```

The output of the command resembles:

```
vxrvg snapprint hr_rvg
Creation Time : Fri Feb 14 02:25:58 2011
Source Volume      Snapshot Volume   Snapshot Type   Sync Status
-------------      ---------------   -------------   -----------
hr-dv01            JULY1-dv01        Inst-Full          Complete
hr-dv02            JULY1-dv02        Inst-Full          Complete

Creation Time : Fri Feb 14 02:25:45 2011
Source Volume     Snapshot Volume    Snapshot Type   Sync Status
-------------     ---------------    -------------   -----------
hr-dv01           JULY-dv01              Inst-Full      Complete
hr-dv02           JULY-dv02              Inst-Full      Complete

Creation Time : Fri Feb 14 01:46:38 2011
Source Volume     Snapshot Volume    Snapshot Type   Sync Status
-------------     ---------------    -------------   -----------
hr-dv01           SO1-dv01               Inst-SO      Incomplete
hr-dv02           SO1-dv02               Inst-SO      Incomplete

Creation Time : Fri Feb 14 01:44:55 2011
Source Volume     Snapshot Volume    Snapshot Type   Sync Status
-------------     ---------------    -------------   -----------
hr-dv01           SO-dv01                Inst-SO      Incomplete
hr-dv02           SO-dv02                Inst-SO      Incomplete

Creation Time : Thu Feb 13 09:14:11 2011
Source Volume     Snapshot Volume   Snapshot Type   Sync Status
-------------     ---------------   -------------   -----------
hr-dv01           JUNE-vol1             Inst-Full      Complete
hr-dv02           JUNE-vol2             Inst-Full      Complete
```

Note: The `vxrvg snapprint` command can also be used to display the status of the snapshots that have been created using the traditional snapshot feature. However, this output will not display the correct time.

Destroying the snapshots

The `vxrvg snapdestroy` command enables you to destroy or delete the snapshot volumes from the RVG. The `vxrvg snapdestroy` command first dissociates the snapshot volumes from the original volumes and then destroys the volumes.

To destroy the snapshot volumes, use the following command:

```
# vxrvg -g diskgroup [-P prefix] [-o keepcache] snapdestroy \
      rvg_name
```

The argument *snapdestroy* along with the different attributes specifies the snapshot that is to be destroyed.

By default, the *snapdestroy* attribute removes the cache object along with the instant snapshots of the specified prefix. However, if you specify the `-o keepcache` option, then the cache object is not deleted. The `-o keepcache` option can be specified only for the pre-created cache objects. The same cache object can then be used for creating new snapshots.

Using the traditional snapshot feature

This snapshot feature of VVR enables you to break off mirrors from the data volumes in an RVG thus providing snapshots of the data volumes in the RVG. Snapshots can be used to perform operations such as Decision Support Systems (DSS) and backup. Snapshots can also be used to retain a consistent copy of the Secondary data volumes during Data Change Map (DCM) resynchronization.

The `vxrvg snapshot` command takes a snapshot of all the volumes in the RVG at a single point in time; therefore, the operation is atomic in nature. The `vxrvg snapback` command reattaches the plexes of the snapshot volumes to the original data volumes in the RVG. The `vxrvg snapshot` command creates the same type of snapshot on the data volumes as a `vxassist snapshot` command would create on a volume associated or unassociated with an RVG. To snapshot and snapback a specific volume or specific plexes of one or more volumes, use the `vxassist` command.

If an RVG contains a volume set, the `vxrvg snapshot` command can be used to take snapshots of its data volumes.

Using the snapshot feature involves the following tasks:

- See “Creating snapshot plexes for data volumes in an RVG” on page 266.
- See “Freezing or pausing replication” on page 266.
- See “Taking a snapshot” on page 266.
- See “Unfreezing or resuming replication” on page 267.

- See "Reattaching the snapshot plexes to the data volumes (Snapback)" on page 267.

Creating snapshot plexes for data volumes in an RVG

To use the RVG snapshot feature, create snapshot plexes for each data volume in the RVG. Creating the snapshot plexes is a one-time operation.

To create a snapshot plex for a volume, use the following command:

```
# vxassist -g diskgroup snapstart dv_name
```

The `vxassist snapstart` command creates a new plex for the volume *dv_name* and attaches it to the volume. When the attach is complete, the state of the plex is snapdone and a snapshot can be taken.

Freezing or pausing replication

Before taking the snapshot on the Secondary, make the data volumes consistent at the application level by either freezing or pausing replication. To make the data volumes consistent at the application level, use the IBC Messaging utility `vxibc`.

See "About the IBC messaging utility vxibc" on page 477.

For a failed Primary, pause the Primary RLINK, and then take a snapshot of the RVG. If you do not use `vxibc`, pause the RLINK before taking the snapshot.

Taking a snapshot

The `vxrvg snapshot` command takes snapshots of the data volumes in the RVG. It creates a snapshot volume with the name SNAP-*dv_name* for each data volume in the RVG.

Each data volume in an RVG can have more than one snapshot volume. The `-P` option to the `vxrvg snapshot` command enables you to specify a prefix for the names of the snapshot plexes. If you specify the prefix *month*, the name of each snapshot data volume starts with *month*; the resulting snapshot volume is named *month-dv_name*. For example, the data volume `hr_dv01` can have snapshot volumes such as `june-hr_dv01`, `july-hr_dv01`.

To take a snapshot of each data volume in an RVG, use the following command:

```
# vxrvg -g diskgroup snapshot rvg_name
```

To specify a prefix for the snapshot of each data volume in an RVG, use the following command:

```
# vxrvg -g diskgroup -P june snapshot rvg_name
```

A snapshot data volume with the name *june-dv_name* is created for each data volume in the RVG. You can have more than one snapshot of the data volumes in an RVG.

Perform the required operation on the snapshots; then snapback, that is, reattach the snapshots to the data volumes in the RVG using the `vxrvg snapback` command.

Unfreezing or resuming replication

After taking a snapshot, unfreeze replication if you are using IBC messaging; otherwise if you have paused replication resume it. The snapshots are ready for use.

Performing the required operations on the snapshot

Use snapshots to perform off-host processing operations including Decision Support Systems (DSS), backup, and trial failover in VVR. Snapshots can also be used to keep a consistent copy of the data volumes in an RVG when DCM resynchronization is in progress. After performing the required operation on the snapshots, reattach them.

Reattaching the snapshot plexes to the data volumes (Snapback)

The snapback operation reattaches a snapshot volume with the original volume. After working with the snapshot volumes, reattach them to the data volumes in the RVG. The snapback operation may take a long time to complete because it performs a complete resynchronization of the snapshot plexes.

To perform a faster and more efficient snapback operation, use the traditional snapshot features.

See "Using the traditional snapshot feature" on page 265.

The `vxrvg snapback` command snaps back, that is, it reattaches the snapshot plexes to their respective data volumes in the RVG.

You can use the default action of the `vxrvg snapback` command if the data volumes in the RVG have one snapshot. If the data volumes have more than one snapshot plex, use the `-a` option with the `vxrvg snapback` command to reattach all snapshots; the `-a` option snaps back all the plexes to their original data volumes. For example, use the `-a` option to reattach the snapshot volumes *june-dv_name* and *july-dv_name* to each data volume *dv_name* in the RVG.

The `-P` option to the `vxrvg snapback` command enables you to reattach a specified snapshot. To reattach all the data volumes with the prefix *month* in their names, specify the prefix `month` using the `-P` option.

For data volumes with single snapshot plexes in an RVG, snapback using the following command:

```
# vxrvg -g diskgroup snapback rvg_name
```

To snapback all plexes for each data volume in an RVG, use the following command:

```
# vxrvg -g diskgroup -a snapback rvg_name
```

All the snapshot plexes are snapped back to their original data volumes in the RVG *rvg_name*.

To snapback snapshot plexes with a specific prefix for all data volumes in an RVG, use the following command:

```
# vxrvg -g diskgroup -P june snapback rvg_name
```

Snapshot plexes with the name *june-dv_name* are snapped back to their original data volumes in the RVG.

To snapback a plex to a specific data volume in an RVG, use the following command:

```
# vxassist -g diskgroup snapback SNAP-dv_name
```

For more information on using the `vxassist snapback` command, see the *Storage Foundation Administrator's Guide*.

Using snapback with resyncfromreplica option

The default action of the `vxassist snapback` command is to resynchronize the snapshot plex with the contents of the original volume. The `resyncfromreplica` option of the `vxassist snapback` command synchronizes the original volume with the contents of the snapshot plex. This operation is similar to a restore from a backup operation. In most cases, the default action of the `vxassist snapback` command must be used, but there are some situations where the `resyncfromreplica` option may be used. The `vxrvg snapback` command does not provide the `resyncfromreplica` option, therefore, the operation must be performed one volume at a time. The `resyncfromreplica` operation is not allowed on the Primary or the Secondary SRL.

Caution: Improper use of the `resyncfromreplica` option of the `vxassist snapback` command on a replicated volume can cause data corruption. You must read the following sections before proceeding.

Using resyncfromreplica option to recover from logical corruption of data

If there is a logical corruption of data and a good snapshot of the data volumes exists, it can be used to restore the data volumes to a version before the error occurred. If the snapshot exists on the Primary, before issuing the `vxassist -o resyncfromreplica snapback` command, shutdown the application and detach all the RLINKs. The `resyncfromreplica` operation will fail if the RLINK is not detached. On completing the snapback operation, perform a complete synchronization of the Secondary data volumes.

See "Methods to synchronize the Secondary" on page 152.

If the snapshots exist on the Secondary, before issuing the `vxassist -o resyncfromreplica snapback` command, migrate the Primary role to this Secondary host, but do not start the application.

See "Migrating the Primary" on page 300.

After migrating the Primary role, detach the RLINK to the original Primary, which is now a Secondary, and then perform the snapback operation. On completing the snapback operation, perform a complete synchronization of the Secondary data volumes.

If you choose to completely synchronize the Secondary using a Storage Checkpoint, make sure that any Primary Storage Checkpoints that were taken before the snapback operation are not used to resynchronize the Secondary. VVR may show these Storage Checkpoints as valid if they have not overflowed; however, the Storage Checkpoints are not valid. You can only use Primary Storage Checkpoints taken after the `resyncfromreplica` operation to resynchronize the Secondaries.

Using resyncfromreplica to recover failed Secondary data volumes

The `resyncfromreplica` option can also be used to restore Secondary data volumes that are corrupt due to disk errors. In this case, the data volumes can be restored from existing snapshots. The RLINK must be in the fail state to perform the `resyncfromreplica` operation. Use these snapshots instead of backups.

See "Restoring the Secondary from online backup" on page 280.

If you choose to restore the Secondary using Storage Checkpoints, you must ensure that:

- The snapshot volume being used for the `resyncfromreplica` operation corresponds to the Storage Checkpoint to be used in the `vxrlink restore` command for the RLINK.

- The Storage Checkpoint is still valid before proceeding with the `resyncfromreplica` snapback operation. Issue the following command to determine whether the Storage Checkpoint is still valid:

  ```
  # vxrlink -g diskgroup cplist rlink_name
  ```

- The snapshot volumes were never written to.

VVR cannot ensure or check if the above conditions are met and failure to meet all of the conditions can result in an inconsistency between the Primary and the Secondary.

Using Veritas Volume Manager FastResync

FastResync (FR) enables you to split off a plex from a mirrored volume, manipulate it, and then reattach it to the original volume without doing a complete resynchronization of the volume. FastResync is a separately licensed feature of VxVM.

FR maintains a bitmap of changes to the volume while the plex was split off, and also of changes to the split-off mirror. When the plex is attached, only the blocks represented in the map are resynchronized.

You can use FR to perform the snapback operation after the off-host processing operation is complete.

The operations performed by the `vxrvg snapshot` and `vxrvg snapback` commands are based on one of the following conditions:

- If you do not have an FR license, the `vxrvg snapshot` command creates a simple snapshot without any FR bitmaps. A `vxrvg snapback` operation then results in a full resynchronization of the plexes.
- If you have an FR license and FastResync is enabled on the volumes, but no DCO logs are attached to the data volumes, the `vxrvg snapshot` command creates a snapshot with a non-persistent FR bitmap. A `vxrvg snapback` operation performs a FastResync, but if the system reboots at any time after the snapshot operation, the information in the FR bitmap is lost and a full resynchronization takes place.
- If you have an FR license, FastResync is enabled, and DCO logs are attached to the data volumes, the `vxrvg snapshot` command creates a snapshot with persistent FR bitmap. A `vxrvg snapback` operation performs a FastResync even if the system reboots at any point in time after the snapshot operation.

For more information on Persistent and Non-Persistent FR, see the *Storage Foundation Administrator's Guide*.

Enabling FastResync

To enable FR on a data volume, type:

```
# vxvol -g diskgroup set fmr=on dv_name
```

Do this for all the data volumes in the RVG that you want to access.

Refer to the *Storage Foundation Administrator's Guide* for more information.

Verifying the DR readiness of a VVR setup

When setting up a Disaster Recovery (DR) solution, it is very important to verify the effectiveness of the DR solution. Although VVR guarantees that integrity of data is maintained between the Primary and Secondary data volumes, validating data is necessary to ensure that there has been no data loss due to administrative error, user error, or some other technical reasons. Validation also helps you to be certain that the data that has been replicated to the Secondary (Disaster Recovery site) can be used to bring up the applications in case of a disaster.

The way to validate the DR readiness of the DR site is to bring up the application on the DR site. This can be done in two ways. One is to migrate the Primary role to the Secondary and then run the applications on the new Secondary using the replicated data. Another way of performing a fire drill is using the snapshot feature. Using this feature VVR creates snapshots of the data volumes that can be used to bring up the application on the Secondary.

Data validation can be used to verify the integrity of the data that has been replicated to the Secondary from the Primary. This is done by comparing the data with the data on the Primary. When the Secondary data volumes are validated after the replication has been stopped, the volumes are dissociated from an RVG. This could be very useful in case you want to validate the data volume before adding it back to the RDS. However, data can also be validated online, that is, when the replication is in progress. This is achieved by using the instant space-optimized snapshot feature to create point in time snapshots of the primary and secondary data volumes. In this case, instead of the actual volumes the snapshot volumes are compared and validated.

See “Creating RVG snapshots” on page 245.

VVR enables you to validate the DR Readiness of the Secondary by using one of the following methods.

- See “Performing a failover” on page 272.
- See “Performing a fire drill” on page 272.
- See “Verifying the data on the Secondary” on page 273.

Performing a failover

A disaster like scenario can be tested by using the migrate operation to perform a complete failover testing. This can be done by migrating the role of the Secondary to a Primary and making sure that the application is running on the new Primary.

See "About transferring the Primary role" on page 299.

Performing a fire drill

Fire drill is a process of bringing up the application on the Secondary using the replicated data. This data is then used to perform some processing in order to validate the consistency and correctness of the data.

To test the failover you can use a point-in-time image of the data on the Secondary data volumes. VVR provides you with the option of creating instant full and space-optimized snapshots.

See "Creating RVG snapshots" on page 245.

You can use the appropriate type of snapshot method to create the snapshots. The instant space-optimized snapshot requires much less space compared to the instant full or plex-breakoff snapshots. These space-optimized snapshots are used to test the Secondary failover.

Observe the following points about fire drills:

- A fire drill cannot be performed using the Secondary volumes therefore the snapshots must be used.
- The Secondary must be in a consistent state when the snapshots are taken.
- When performing a fire drill no IBC messages are required to be sent so that a failover scenario similar to a real one is simulated.

Automating the fire drill procedure

The fire drill procedure is most effective only when it is performed on a regular basis. The above method requires you to test the Secondary failover, manually, at frequent intervals. However, in case VVR is used in a VCS setup where the appropriate agents are installed, then the fire drill procedure can be automated using the `RVGSnapshot` and `RVGPrimary` agents that VCS provides. For more information on how you can use these agents to automate the fire drill testing, refer to the VCS Documents.

Verifying the data on the Secondary

VVR enables you to verify that the data on the Secondary is identical to the data on the Primary data volumes, either when the application is active or inactive. VVR provides the following methods to verify the data at the Secondary site: online data verification and offline data verification.

Online data verification allows you to validate the data even when replication is in progress. In this method instead of the actual volumes, the point-in-time snapshots are compared. This method is referred to as online data verification.

Offline data verification can be performed only when replication is not active. If you have already created the Primary and Secondary volumes and replication is in progress, you need to pause replication and then perform data validation between the corresponding Primary and Secondary volumes to make sure that the data volumes on the Primary and Secondary are the same. To do this, use the `vradmin syncrvg` command with the `-verify` option. To use this command for verifying the data, the Secondary must be up-to-date. This command performs a comparison of the checksums on the corresponding Primary and Secondary volumes.

You can also validate the data on new data volumes before adding them to an RDS.

See "Verifying the data on the Primary and Secondary volumes " on page 197.

See "About SmartMove for VVR" on page 157.

Performing online data verification

The space-optimized snapshots that are created using the `vxrvg snapshot` command can be used to verify whether the data on the Primary and Secondary RVG volumes is the same.

The major advantage of this feature over the `vradmin -verify syncrvg` command is that you do not need to stop the replication. The verification can be done even while the replication is in progress because the point-in-time snapshots, and not the volumes, are compared. This feature is very useful if you want to check the integrity of the data volumes on the Secondary when replication is in progress.

The `vradmin verifydata` command creates the space-optimized snapshots on the Primary and the Secondary before it proceeds with performing online data verification. The `vradmin verifydata` command also ensures that the snapshots are taken only after the replication has been paused using the `vxibc freeze` command. As a result there may be a momentary pause in the replication. It is necessary to freeze the writes so that the snapshots can be taken at an identical point in replication time, on each of the required hosts.

The `vradmin verifydata` then verifies the data between the remote and local hosts by comparing the space-optimized snapshots.

The `vradmin verifydata` command performs the following tasks:

- Registering the application on the Primary and the Secondary.
- Freezing replication on the Primary and Secondary.
- Taking snapshots and verifying the data.
- Destroying the snapshots.

By default, the `vradmin verifydata` command destroys the snapshot volume and the cache object after the data verification has proceeded successfully. However, if you want to preserve the snapshot volumes then you must use the `vradmin verifydata` command with the `-k snap` option. If you want to preserve the cache object then use the `vradmin verifydata` command with the `-k cache` option. The same cache object can then be reused when creating future snapshots. You cannot use the -k option if you have used the `cachesize` option, as it is an invalid combination and the command fails with an error message. Note that when specifying the `-k` option you must specify either the *`cache`* or the *`snap`* argument with it.

Note: When the `-k snap` option is specified the cache object is also preserved along with the snapshot since the snapshot cannot exist without the cache object.

VVR also provides you with sample scripts that can be used to freeze the replication and then take instant space-optimized snapshots.

See "Sample vradmin ibc command scripts" on page 293.

To perform online data verification

1. Prepare the volumes that need to be included in the snapshot.

   See "Preparing the volumes prior to using the instant snapshot feature" on page 248.

2. Create the required cache object within the same disk group as the data volume.

   See "Preparing the RVG volumes for snapshot operation" on page 253.

3. To perform online data verification, use the command:

   ```
   vradmin [-g diskgroup] [-k {cache|snap}] verifydata rvg_name \
         sechost {cache=cacheobj | cachesize=size}
   ```

   The attribute `sechost` specifies the name of the Secondary host.

   The `cache` attribute specifies a name for the precreated cache object, on which the snapshots for the volumes in the specified RVG will be created. The `cachesize` attribute specifies a default size for the cache object with respect to the source volume.

   You must specify only one of these attributes at one time for the command to create a cache object for each snapshot.

Performing offline data verification

VVR enables you to verify whether the data on the Secondary is identical to the data on the Primary data volumes when the application is inactive. The `vradmin syncrvg` command with the `-verify` option verifies and reports any differences between the data volumes associated with the Secondary RVG and the corresponding Primary RVG. If a volume set is associated to the RDS, the `vradmin -verify syncrvg` command verifies only the component volumes that are associated to the RVG. The `vradmin -verify syncrvg` command only reports whether the Primary and Secondary volumes are identical or not. It does not make them identical. As the command runs, it reports the progress every 10 seconds. An MD5 checksum is used to calculate the difference between the Primary and the Secondary data volumes.

See "Using difference-based synchronization" on page 509.

Prerequisites for using the `vradmin -verify syncrvg` command

Observe the following prerequisites:

- All applications using the Primary data volumes must be stopped before running the `vradmin -verify syncrvg` command.

To verify the differences between the Primary and Secondary data volumes

- Use the following command to verify the differences between the Primary and Secondary data volumes

  ```
  # vradmin -g diskgroup -verify syncrvg local_rvgname \
          sec_hostname...
  ```

  When this command is invoked, you are prompted to confirm that the Primary data volumes are not in use. You can use the `-s` option to skip this confirmation step.

  The argument local_rvgname is the name of the RVG on the local host and represents the RDS.

  The argument sec_hostname is a space-separated list of the names of the Secondary hosts as displayed in the output of the `vradmin printrvg` command.

  This command checks the status of the Primary RLINK to each of the Secondary RVGs being verified. If any of the RLINKs are not up-to-date, the `vradmin -verify syncrvg` command returns with a message to indicate that the RLINKs are not up-to-date. In this scenario, verification is not be performed. Use the `vxrlink status` command to determine the extent to which the Secondary is behind.

Example:

To verify the data differences between the Primary RVG `hr_rvg` on `seattle` and the Secondary RVG on host `london`, issue the following command from any host in the RDS:

```
# vradmin -g hrdg -verify syncrvg hr_rvg london
```

```
The output resembles the following if the Primary and Secondary data volumes
are identical:
Message from Primary:
VxVM VVR vxrsync INFO V-5-52-2210 Starting volume verification to remote
VxVM VVR vxrsync INFO V-5-52-2211    Source host: 10.182.136.192
VxVM VVR vxrsync INFO V-5-52-2212    Destination host(s): 10.182.136.193
VxVM VVR vxrsync INFO V-5-52-2213    Total volumes:       1
VxVM VVR vxrsync INFO V-5-52-2214    Total size:          4.000 G


Eps_time Dest_host       Src_vol     Dest_vol     F'shed/Tot_sz  Diff  Done
00:00:00 10.182.136.193    hr_dv       hr_dv                 0M/4096M     0%    0%
00:00:10 10.182.136.193    hr_dv       hr_dv               221M/4096M     0%    5%
Message from Primary:
```

```
00:00:20 10.182.136.193    hr_dv       hr_dv               468M/4096M     0%   11%
Message from Primary:
00:00:30 10.182.136.193    hr_dv       hr_dv               705M/4096M     0%   17%
Message from Primary:
00:00:40 10.182.136.193    hr_dv       hr_dv               945M/4096M     0%   23%
Message from Primary:
00:00:50 10.182.136.193    hr_dv       hr_dv              1184M/4096M     0%   29%
Message from Primary:
00:01:00 10.182.136.193    hr_dv       hr_dv              1419M/4096M     0%   35%
Message from Primary:
00:01:10 10.182.136.193    hr_dv       hr_dv              1655M/4096M     0%   40%
Message from Primary:
00:01:20 10.182.136.193    hr_dv       hr_dv              1886M/4096M     0%   46%
Message from Primary:
00:01:30 10.182.136.193    hr_dv       hr_dv              2124M/4096M     0%   52%
Message from Primary:
00:01:40 10.182.136.193    hr_dv       hr_dv              2356M/4096M     0%   58%
00:01:50 10.182.136.193    hr_dv       hr_dv              2590M/4096M     0%   63%
Message from Primary:
00:02:00 10.182.136.193    hr_dv       hr_dv              2838M/4096M     0%   69%
Message from Primary:
00:02:10 10.182.136.193    hr_dv       hr_dv              3091M/4096M     0%   75%
Message from Primary:
00:02:20 10.182.136.193    hr_dv       hr_dv              3324M/4096M     0%   81%
Message from Primary:
00:02:30 10.182.136.193    hr_dv       hr_dv              3564M/4096M     0%   87%
Message from Primary:
00:02:40 10.182.136.193    hr_dv       hr_dv              3809M/4096M     0%   93%
Message from Primary:
00:02:50 10.182.136.193    hr_dv       hr_dv              4070M/4096M     0%   99%
00:02:51 10.182.136.193    hr_dv       hr_dv              4096M/4096M     0%  100%
VxVM VVR vxrsync INFO V-5-52-2217 The volumes are verified as identical.


VxVM VVR vxrsync INFO V-5-52-2219 VxRSync operation completed.
VxVM VVR vxrsync INFO V-5-52-2220 Total elapsed time: 0:02:51
```

If there are differences in the data volumes, the output looks similar to the one shown below:

```
Message from Primary:
VxVM VVR vxrsync INFO V-5-52-2210 Starting volume verification to remote
VxVM VVR vxrsync INFO V-5-52-2211    Source host:         10.182.136.192
VxVM VVR vxrsync INFO V-5-52-2212    Destination host(s): 10.182.136.193
```

```
VxVM VVR vxrsync INFO V-5-52-2213    Total volumes:       1
VxVM VVR vxrsync INFO V-5-52-2214    Total size:          4.000 G


Eps_time Dest_host       Src_vol     Dest_vol     F'shed/Tot_sz  Diff  Done
00:00:01 10.182.136.193    hr_dv       hr_dv                 0M/4096M     0%    0%
00:00:11 10.182.136.193    hr_dv       hr_dv               231M/4096M    48%    6%
Message from Primary:
00:00:21 10.182.136.193    hr_dv       hr_dv               476M/4096M    23%   12%
Message from Primary:
00:00:31 10.182.136.193    hr_dv       hr_dv               719M/4096M    15%   18%
Message from Primary:
00:00:41 10.182.136.193    hr_dv       hr_dv               954M/4096M    12%   23%
Message from Primary:
00:00:51 10.182.136.193    hr_dv       hr_dv              1202M/4096M     9%   29%
Message from Primary:
00:01:01 10.182.136.193    hr_dv       hr_dv              1438M/4096M     8%   35%
Message from Primary:
00:01:11 10.182.136.193    hr_dv       hr_dv              1680M/4096M     7%   41%
Message from Primary:
00:01:21 10.182.136.193    hr_dv       hr_dv              1924M/4096M     6%   47%
Message from Primary:
00:01:31 10.182.136.193    hr_dv       hr_dv              2165M/4096M     5%   53%
Message from Primary:
00:01:41 10.182.136.193    hr_dv       hr_dv              2418M/4096M     5%   59%
Message from Primary:
00:01:51 10.182.136.193    hr_dv       hr_dv              2668M/4096M     4%   65%
00:02:01 10.182.136.193    hr_dv       hr_dv              2906M/4096M     4%   71%
Message from Primary:
00:02:11 10.182.136.193    hr_dv       hr_dv              3140M/4096M     4%   77%
Message from Primary:
00:02:21 10.182.136.193    hr_dv       hr_dv              3386M/4096M     3%   83%
Message from Primary:
00:02:31 10.182.136.193    hr_dv       hr_dv              3630M/4096M     3%   89%
Message from Primary:
00:02:41 10.182.136.193    hr_dv       hr_dv              3881M/4096M     3%   95%
Message from Primary:
00:02:49 10.182.136.193    hr_dv       hr_dv              4096M/4096M     3%  100%
VxVM VVR vxrsync INFO V-5-52-2218 Verification of the remote volumes found
differences.
```

```
VxVM VVR vxrsync INFO V-5-52-2219 VxRSync operation completed.
VxVM VVR vxrsync INFO V-5-52-2220 Total elapsed time: 0:02:50
```

Backing up the Secondary

You must take backups on the Secondary on a regular basis to guard against the consequences of disk failures. The Secondary checkpointing facility allows volume-level backups of the RVG to be restored at the Secondary node.

To get a consistent backup, replication must not be active. You do this by initiating a Secondary Storage Checkpoint at the Secondary node. This causes a request to be sent to the Primary node to pause updates and record the Secondary Storage Checkpoint in the SRL. While replication is paused, take a block-level backup of the RVG at the Secondary node. When the backup is complete, resume replication. Initiating a `resume` at the Secondary node causes a request to be sent to the Primary node to resume updates. Note that the Secondary cannot do a Storage Checkpoint if it has lost contact with the Primary.

If you must recover from Secondary data volume failure, then after the block-level backup has been restored, subsequent updates can be replayed from the Primary starting at the Storage Checkpoint and the Secondary can be brought up-to-date. The Secondary can be brought up-to-date only if the updates are still in the SRL. You can display a list of Storage Checkpoints on the Secondary using the `vxrlink cplist` command.

See "Displaying a list of Storage Checkpoints" on page 176.

Restoring the Secondary RLINK from a Secondary Storage Checkpoint

If a Secondary data volume fails and there is a Storage Checkpoint backup created as outlined above, you can restore from this backup copy without having to do a full Primary resynchronization of all the volumes. This procedure is also referred to as doing an online restore for the Secondary, because it is not necessary to stop the Primary RVG to bring a new copy of the Secondary data volume up-to-date.

Note: The names of existing Secondary Storage Checkpoints can be obtained by performing the `vxrlink cplist` command on the Primary. The `vxrlink cplist` command can also be used to monitor whether the earlier Storage Checkpoints are about to overflow.

The Storage Checkpoint string can be up to 19 characters long for either a Primary or a Secondary Storage Checkpoint. Only the most recent Primary Storage Checkpoint string is displayed in the Primary RVG.

Unlike a simple Secondary pause, a Storage Checkpoint Secondary pause will fail if the Secondary is disconnected from the Primary at the time the command is executed, because communication with the Primary is required to create the Storage Checkpoint.

To create a Secondary Storage Checkpoint

On the Secondary:

1. Pause the RLINK using a Storage Checkpoint.

```
# vxrlink -g diskgroup -c sec_checkpointname pause \
 rlink_name
```

Note: Pausing the RLINK with a Storage Checkpoint on the Secondary creates the Secondary Storage Checkpoint.

---

2. Back up all the volumes in the Secondary RVG, using a block-level backup.
3. Resume the RLINK.

```
# vxrlink -g diskgroup resume rlink_name
```

To delete a Secondary Storage Checkpoint

1. Pause the RLINK using a Storage Checkpoint on Secondary.

```
# vxrlink -g diskgroup -c sec_checkpointname pause \
 rlink_name
```

2. Delete the Secondary Storage Checkpoint, using the following command:

```
# vxrlink -g diskgroup -c sec_checkpointname checkdelete \
 rlink_name
```

Note: Perform step 2 only on the Primary.

Restoring the Secondary from online backup

To restore the Secondary from online backup, perform the following tasks, in the order shown:

- See "Restoring from Secondary Storage Checkpoints" on page 281.
- See "Restoring a Secondary RLINK " on page 281.

Restoring from Secondary Storage Checkpoints

If a Secondary volume becomes corrupted due to I/O errors, the volume can be restored from backup. When a `vxrlink restore` is initiated, a request is sent to the Primary to begin updates from a previously recorded Storage Checkpoint. A `restore` is not guaranteed to succeed though because Storage Checkpoints can become stale which means that the Primary has stopped maintaining the updates necessary for the restore. If this occurs, the Secondary RVG must be re-initialized using a Primary Storage Checkpoint or an autosync attach instead of being restored from backup.

Restoring a Secondary RLINK

If a Secondary data volume fails the RLINK is put into the FAIL state. A restore from an online backup copy becomes necessary. This can only be done if a suitable Primary or Secondary Storage Checkpoint exists. If a Primary Storage Checkpoint still exists, it can be used if there is no Secondary Storage Checkpoint.

To restore a Secondary from an on-line backup, first restore the data from the on-line backup to all of the volumes. Because of internal constraints, you must restore all volumes even if only one has failed. (The normally read-only Secondary data volumes are writable while the Secondary is in fail state.) Then, execute the `vxrlink -c` *`checkpoint_name`* `restore` *`rlink`* command, which causes the Secondary to request all updates which were made subsequent to the Storage Checkpoint from the Primary.

As with Primary Storage Checkpoints, if the Storage Checkpoint is not used before the SRL wraps around and the SRL overflows, the Storage Checkpoint will become STALE. If the Storage Checkpoint becomes STALE, you cannot use the methods described in this section to restore the data. You must synchronize the RLINK.

See "Methods to synchronize the Secondary" on page 152.

To prevent the Storage Checkpoint from becoming STALE, make sure the SRL is large enough to hold all of the updates which occur between the `vxrlink -c` *`checkpoint`* `pause` command and the `vxrlink -c` *`checkpoint`* `restore` command.

Synchronizing the RLINK to prevent STALE Storage Checkpoints

On the Secondary:

1. Assuming the RLINK is in the fail state, restore the backup to the data volumes.
2. Restore the RLINK to initiate updates to the Secondary data volumes:

```
# vxrlink -g diskgroup -c checkpoint_name restore rlink_name
```

Note: In situations where you need to perform a restore when the RLINK is not in a FAIL state, use the following command to get the RLINK in a fail state:

```
# vxrlink -g diskgroup -w pause rlink_name
```

For example, you need to get an RLINK back in the FAIL state if a data volume fails, you restore from backup, and then after performing the restore command, you realize that the wrong backup was used. In this case, you need to get the RLINK back in the FAIL state before you perform the `restore` command.

While the restore is occurring, the RLINK is inconsistent. It becomes consistent when the `vxrlink restore` command completes successfully.

Chapter 9

Using VVR for off-host processing

This chapter includes the following topics:

- About using VVR for off-host processing
- What is off-host processing?
- In-Band Control Messaging overview
- In-BandControl Messaging explained
- Performing off-host processing tasks
- Examples of off-host processing

About using VVR for off-host processing

This chapter explains how to use Volume Replicator (VVR) for off-host processing on the Secondary host. You can use the In-Band Control (IBC) Messaging feature with the FastResync (FMR) feature of Veritas Volume Manager (VxVM) and its integration with VVR to take application-consistent snapshots at the replicated volume group (RVG) level. This lets you perform off-host processing on the Secondary host.

This chapter explains how to perform off-host processing operations using the `vradmin ibc` command. You can also use the `vxibc` commands to perform off-host processing operations.

See "About the IBC messaging utility vxibc" on page 477.

What is off-host processing?

Off-host processing consists of performing operations on application data on a host other than the one where the application is running. Typical operations include Decision Support Systems (DSS) and backup. In a VVR environment, off-host processing operations can be performed on the Secondary of the Replicated Data Set. This reduces the load on the application server, the Primary.

The model for data access on the Secondary is that you break off a mirror from each data volume in the RVG, perform the operation on the mirror, and then reattach the mirror while replication is in progress.

In-Band Control Messaging overview

When you take a snapshot on the Secondary, it contains a point-in-time copy of the data on the Primary. Because the Secondary may be behind the Primary, it is not known at exactly what time this point-in-time copy was made.

IBC messaging enables you to send a message in the replication stream to notify the Secondary that an event has occurred on the Primary. In the case of a file system, you can use the `sync` command on the Primary, and then send an IBC message. When this message arrives on the Secondary, the data on the Secondary is consistent at the file system level and replication stops. You then take a snapshot, which now contains a consistent image of the file system, and unfreeze replication.

The model with IBC Messaging is that a process on the Secondary waits for the IBC Message, and a process on the Primary sends the message when the desired event has occurred.

VVR provides the following options for IBC Messaging:

- Single command to perform off-host processing operations—`vradmin ibc`
  This chapter explains the function of IBC Messaging and how to use the command `vradmin ibc` for off-host processing.
- IBC Messaging command-line utility—The `vxibc` utility
  See "Using the IBC messaging command-line utility" on page 479.

How to use the data on the Secondary

To use the data on the Secondary host for perform off-host processing operations, use snapshots of the Secondary data volumes. Do not mount the Secondary RVG volumes directly, even in read-only mode.

Using snapshots of Secondary data volumes

A snapshot is an image of the online data volumes at a specific point-in-time. Use snapshots of the Secondary data volumes to perform off-host processing operations, instead of directly using the Secondary data volumes. The data on the original volumes may change but the data on the snapshot can still be used as a stable and independent copy for various purposes.

VVR provides two methods of creating snapshots: instant snapshots and traditional snapshots. The instant snapshot feature is a separately licensed feature of Veritas Volume Manager (VxVM).

See "Using the instant snapshot feature" on page 246.

For more information on FastResync, see the FastResync section in the *Storage Foundation Administrator's Guide*.

VVR also provides sample IBC scripts that can be used for creating snapshots.

See "Sample vradmin ibc command scripts" on page 293.

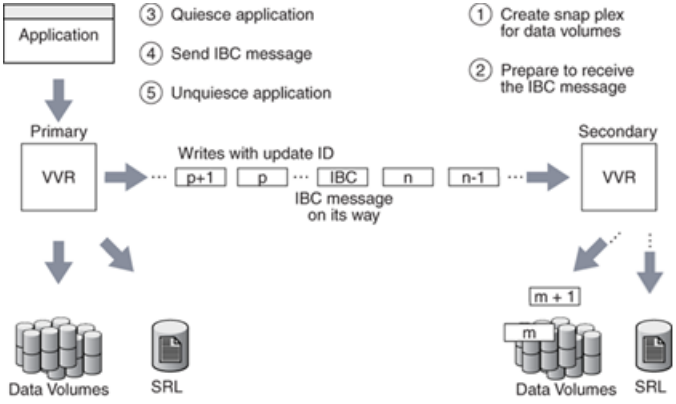
With the traditional snapshot feature, depending on the size of the volume, the time required for initial synchronization of the plexes can be very large.

See "Using the traditional snapshot feature" on page 265.

Before you can use the snapshot, some application-dependent recovery procedure has to be performed. For example, if the volume contains a file system, run the `fsck` program before mounting the file system.

In-BandControl Messaging explained

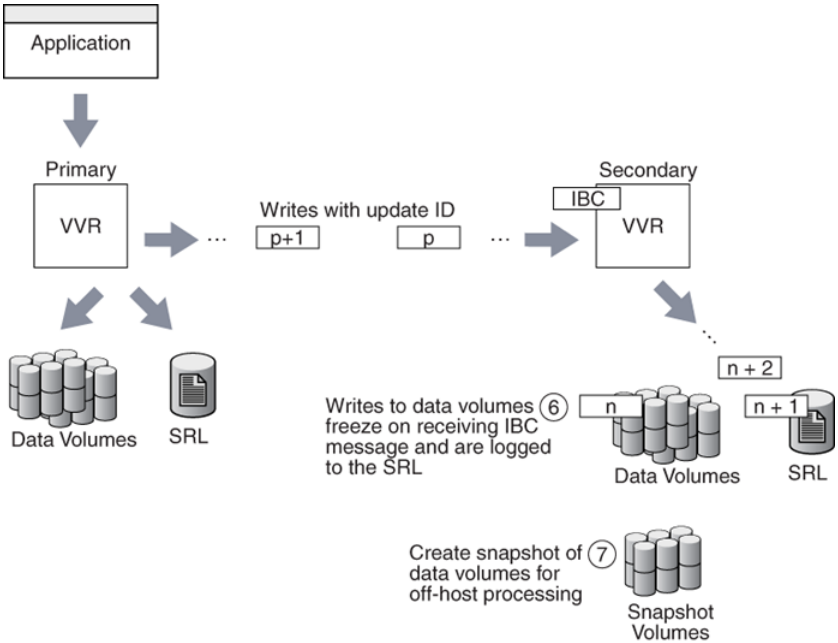
You can use IBC Messaging to notify the Secondary that the data volumes in the Primary RVG are consistent at the application-level.



Using IBC messaging, you can inject a user-defined control message into the update stream of an RVG at a point when the Primary is consistent at the application level. When the IBC message reaches the Secondary, data volumes on the Secondary are frozen. As a result, the Secondary does not reflect further updates to data volumes until the user acknowledges the IBC message.

Note: During this process, all messages are continuously logged into the Secondary SRL

The Secondary data volumes are now consistent at the application level.



While replication is frozen, take snapshots of the data volumes on the Secondary. The created snapshot volumes are consistent at the application level and require less time to recover when the application on the Secondary starts.

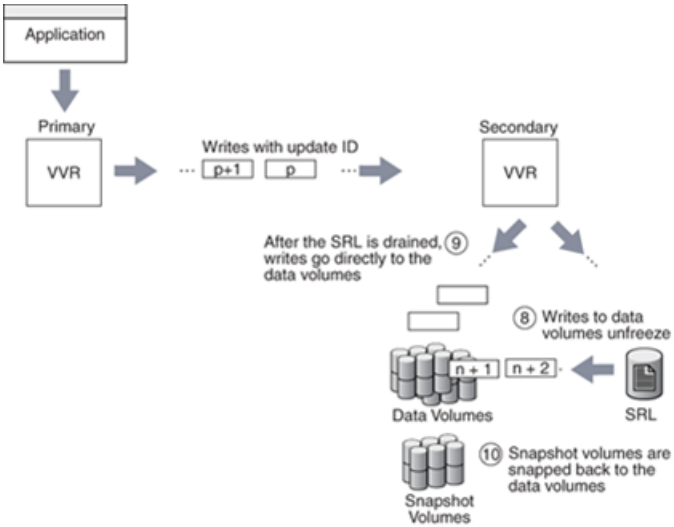
An application must be in a quiesced mode before the IBC message is sent to the Secondary, to achieve application-level consistency. For a database application running on the Primary host, an IBC message can be inserted at a point at which the application considers its raw volume contents to be consistent, such as during the database hot-backup mode.

In the case of a file system, when you enter the `sync` command on the Primary to flush the previously unwritten file system buffers to the data volume, the file

modifications up to that point are saved to the data volume. You can insert an IBC message on the Primary to notify the Secondary that the `sync` command is complete. In general, there is no way to keep the file system in the synced state while you generate the IBC; however, if this is done as soon as the sync completes, there should be little to recover on the Secondary.

Even if you are using synchronous replication, IBC Messaging is useful to notify the Secondary when the consistency point is reached and to ensure that the writes on the Secondary are written to the data volumes after the snapshot is taken.

When the IBC reaches the Secondary, the data volumes are frozen. Now, the Secondary data volumes are consistent at the application level, and you can take a snapshot. If you take a backup of the snapshot volumes, the file systems on the backup volumes are consistent.



An IBC message is delivered to the Secondary host in causal order with respect to other activity on the data volumes. Before delivery of the message to the Secondary host, any previous update activity is flushed. You have the option of allowing subsequent updates to continue on the Secdoary SRL and data volumes, or logging the subsequent updates into the SRL only until released by unfreezing the Secondary RVG. Since the purpose of the IBC message is to achieve a sync point on the Secondary, choose the option to stop subsequent updates until the snapshot is taken. This is the default option in the examples below.

IBC messages are guaranteed to be delivered at least once and might be delivered more than once if an error such as a network outage or machine crash occurs during

the delivery. The script you write to perform the IBC operation must be able to manage receiving the multiple delivery of the same IBC message.

If the Secondary crashes before it receives the IBC, the receiving program must be restarted when the Secondary comes up again. Note that in a shared disk group environment, if the node that is the current master leaves the cluster, the IBC program must re-register the application name on the node that becomes the master.

Performing off-host processing tasks

VVR enables you to integrate application preparation tasks, IBC messaging, and off-host processing tasks using the single command `vradmin ibc`. The `vradmin ibc` command simplifies the off-host processing task by providing scripts for application-specific tasks. The `vradmin ibc` command uses IBC Messaging and executes a set of user-defined scripts to perform the required off-host processing task. You are not required to remember the sequence in which to perform these tasks. The `vradmin ibc` command simplifies the off-host processing process by performing the following operations:

- Executes application-specific scripts and the scripts to be used for off-host processing in the required sequence.
- Executes these scripts on the appropriate host, that is, Primary or the Secondary.
- Inserts the IBC message at the appropriate time.

Tasks to perform for off-host processing

Following is the typical sequence of tasks that must be performed to use IBC Messaging for off-host processing:

1. Prepare the Secondary for off-host processing. For example, create snapshot plexes on the data volumes on the Secondary (`prefreeze` task).
2. Register an application name for sending and receiving IBC messages on the Primary and the Secondary. Prepare to receive the IBC message on the Secondary.
3. Quiesce the application on the Primary (`quiesce` task).
4. Send the IBC message from the Primary to the Secondary.
5. Unquiesce the application on the Primary (`unquiesce` task).
6. Perform the off-host processing task on the Secondary after the Secondary receives the IBC message and replication freezes (`onfreeze` task). Note that the updates to the Secondary data volume is frozen while the off-host processing task is in progress.

7. Unfreeze the Secondary after the off-host processing task completes.
8. Perform additional tasks on the Secondary after replication has resumed. For example, reattaching the snapshot volumes to the original data volumes on the Secondary (`postfreeze` task).
9. Unregister the application on both the Primary and the Secondary.

The single command `vradmin ibc` can be used to perform this sequence of tasks. To use the `vradmin ibc` command, you need to provide scripts named `prefreeze`, `quiesce`, `unquiesce`, `onfreeze`, `postfreeze` to perform the tasks in step 1, step 3, step 5, step 6, and step 8 respectively. The `vradmin ibc` command uses these user-defined scripts with IBC Messaging to perform these tasks in sequence.

See "Understanding the scripts used for the vradmin ibc command" on page 291.

You can also use the `vxibc` commands to perform off-host processing operations.

See "Using the IBC messaging command-line utility" on page 479.

Using the IBC messaging command vradmin ibc

The `vradmin ibc` command enables you to perform off-host processing operation in a single command.

Prerequisites for using vradmin ibc command

Observe the following prerequisites:

- The Primary RLINK that points to the Secondary host participating in the `vradmin ibc` command must be in the connect state.
- The `onfreeze` script must exist on each Secondary host participating in the vradmin ibc command.
- Make sure each user-defined script to be used in the `vradmin ibc` command exits with a status of 0 on successful completion and a status of nonzero on unsuccessful completion.
- The user-defined scripts must have `execute` permissions for `root` user.
- Make sure that scripts have interpreter information in the first line of the IBC shell script. If the first line is a comment or is blank, `vradmin ibc` does not execute the script.

Caution: The `vradmin ibc` executes scripts using root privileges. If the scripts can be modified by a non-privileged user, there is a potential security risk. To prevent this, ensure that you have the proper access privileges set on the scripts used with the `vradmin ibc` command.

To perform an off-host processing task on one or more Secondary RVGs in an RDS

1. Make sure the RLINKs are in the CONNECT state. If the RLINKs are not in the CONNECT state, use the `vradmin startrep` command to start replication.

2. Create a directory to store the user-defined scripts for this off-host processing task. Create the following directory on all hosts participating in the `vradmin ibc` command:

   ```
   /etc/vx/vvr/ibc_scripts/task_name
   ```

   where *`task_name`* is the name of the off-host processing task and is the same as the *`task_name`* argument used in the `vradmin ibc` command.

3. Create the appropriate scripts for the required off-host processing task and copy the scripts into the directory created in step 2.

   See "Understanding the scripts used for the vradmin ibc command" on page 291.

4. Run the following command from any host in the RDS:

   ```
   # vradmin -g diskgroup ibc rvg_name task_name [sechost]...[-all]
   ```

   The argument *`diskgroup`* represents the disk group that contains the RVG on the local host.

   The argument *`rvg_name`* is the name of the RVG on the local host and represents its RDS.

   The argument *`task_name`* is the name of the off-host processing task and is the same as the name of the directory created in step 2.

   The argument *`sechost`* is the name of the Secondary host as displayed in the output of the `vradmin printrvg` command. The argument *`sechost`* is optional if the RDS contains only one Secondary. To perform the task on multiple Secondary hosts, specify a space-separated list with the name of each Secondary to be included. Use the `-all` option to perform the task on all the Secondary hosts in the RDS.

Example – Creating a snapshot on the Secondary using the vradmin ibc command

This example shows how to create a snapshot of the data volumes on the Secondary `london` using the `vradmin ibc` command. The RVG `hr_rvg`, which belongs to the disk group `hrdg`, has been created on the Primary and Secondary. This example also assumes that Secondary data volumes have associated snapshot plexes. It uses the application name `dss_app`.

1. Create the following directory on Secondary host:

   ```
   /etc/vx/vvr/ibc_scripts/dss_app
   ```

2. Create the `onfreeze` script in the `/etc/vx/vvr/ibc_scripts/dss_app` directory on the Secondary host by including the following command to create the snapshot of the data volumes on the Secondary:

   ```
   #!/bin/sh
   /usr/sbin/vxrvg -g hrdg snapshot hr_rvg
   ```

   You can use the `vxrvg snapshot` command to create instantfull, instantso, or plexbreakoff snapshots.

   See "VVR command reference" on page 466.

3. On the Primary, put the application using the Primary data volumes in the quiesce mode.

4. Create the snapshot by running the following command on any host in the RDS:

   ```
   # vradmin -g hrdg ibc hr_rvg dss_app london
   ```

5. On the Primary, take the application out of quiesced mode to resume it.

Understanding the scripts used for the vradmin ibc command

The `vradmin ibc` command executes the user-defined scripts: `prefreeze`, `quiesce`, `unquiesce`, `onfreeze`, and `postfreeze`. Note that the `onfreeze` script is mandatory and must be present on the Secondary. The scripts `prefreeze`, `quiesce`, `unquiesce`, and `postfreeze` are optional. However, if you provide the `quiesce` script, you must provide the `unquiesce` script and vice versa. You must name the user-defined scripts `prefreeze`, `quiesce`, `unquiesce`, `onfreeze`, or `postfreeze`.

Note: A user-defined script can either be a shell script or a binary.

The following describes how each script is used with the `vradmin ibc` command:

- The `prefreeze` script

  Use this script on the Secondary to prepare for the tasks to be performed in the `onfreeze` script while the replication to Secondary is frozen. For example, if you want to take a snapshot of the Secondary data volumes while the replication on the Secondary is frozen, the `prefreeze` script can be used to add snapshot plexes to the Secondary data volumes to prepare for the `snapshot` command.

- The `quiesce` script

  The `vradmin ibc` command executes the `quiesce` script on the Primary before it sends the IBC message to the Secondary. Use this script to quiesce the application running on the Primary RVG and to make the Primary data volumes consistent at the application level. The `vradmin ibc` command injects an IBC message in a small amount of time, and hence the duration for which the application remains quiesced is small.

- The `unquiesce` script

  The `vradmin ibc` command executes this script on the Primary after it sends the IBC message to the Secondary. Use this script to resume the application running on the Primary if the application was quiesced.

- The `onfreeze` script

  The `vradmin ibc` command executes this script on the Secondary while replication on the Secondary is frozen after receiving the IBC message from the Primary. Use this script to perform the required off-host processing operation, for example, taking a snapshot of the Secondary data volumes.

- The `postfreeze` script

  The `vradmin ibc` command executes this script on the Secondary after it executes the `onfreeze` script and after the replication on the Secondary is unfrozen. For example, if a snapshot of the Secondary data volumes was taken in the `onfreeze` script, this script can be used to reattach the snapshot volumes to the Secondary data volumes.

Location of the scripts used for the vradmin ibc command

The scripts must reside in the `/etc/vx/vvr/ibc_scripts/`*`task_name`* directory on the Primary and the Secondary host. Note that *`task_name`* is the name of the off-host processing task and is the same as the *`task_name`* argument used in the `vradmin ibc` command. For example, if the off-host processing task is Decision Support Systems (DSS), you can choose a task name of `dss`; or if the off-host processing task is `Backup`, you can choose the task name of backup.

Table 9-1 shows the locations of the scripts for off-host processing.

Table 9-1 Locations of the scripts for off-host processing

Host	Directory	Name of Script
Primary	/etc/vx/vvr/ibc_scripts/*task_name*	quiesce
Primary	/etc/vx/vvr/ibc_scripts/*task_name*	unquiesce
Secondary	/etc/vx/vvr/ibc_scripts/*task_name*	onfreeze
Secondary	/etc/vx/vvr/ibc_scripts/*task_name*	prefreeze
Secondary	/etc/vx/vvr/ibc_scripts/*task_name*	postfreeze

In a shared disk group environment, the scripts must exist on each node in the Primary or Secondary cluster. That is, the `quiesce` and `unquiesce` scripts must exist on each node in the Primary cluster; the `onfreeze`, `prefreeze`, and `postfreeze` scripts must exist on each node in the Secondary cluster.

When the `vradmin ibc` command executes each script, it passes the following arguments to the script:

Argument 1	Name of the disk group of the Primary RVG.
Argument 2	Name of the Primary RVG.
Argument 3	The *task_name* specified in the `vradmin ibc` command.
Remaining Arguments	Names of the RLINKs participating in the `vradmin ibc` command.

Sample vradmin ibc command scripts

The `/etc/vx/vvr/ibc_scripts` directory contains the following sample script directories:

```
sample_db_snapshot
sample_vxfs_snapshot
sample_so_snapshot
```

These sample scripts show how to use the user-defined scripts with the `vradmin ibc` command. Refer to the README file provided in each sample directory for instructions on how to use the corresponding script to perform off-host processing tasks.

Note: Sample scripts are provided for reference. Customize the sample scripts to suit your requirements.

Examples of off-host processing

The examples in this chapter assume that the following VVR configuration has been set up on the Primary and Secondary hosts:

Name of the Primary host: `seattle`

`hrdg`	Disk group
`hr_rvg`	Primary RVG
`rlk_london_hr_rvg`	Primary RLINK for Secondary `london`
`hr_dv01`	Primary data volume #1
`hr_dv02`	Primary data volume #2
`hr_srl`	Primary SRL volume

Name of the Secondary host: `london`

`hrdg`	Disk group
`hr_rvg`	Secondary RVG
`rlk_seattle_hr_rvg`	Secondary RLINK for Primary `seattle`
`hr_dv01`	Secondary data volume #1
`hr_dv02`	Secondary data volume #2
`hr_srl`	Secondary SRL volume

These examples use the application name `dss_app` for off-host processing tasks.

Example - Decision support using the snapshot feature and the vradmin ibc command

This example shows implementing decision support using the snapshot feature and the `vradmin ibc` command.

To use the snapshot feature and the `vradmin ibc` command for decision support

1. Create the following directory on both Primary and Secondary hosts:

   `/etc/vx/vvr/ibc_scripts/dss_app`

2. Create the quiesce and unquiesce scripts and copy them to the `/etc/vx/vvr/ibc_scripts/dss_app` directory on the Primary host.

   In the `quiesce` script, provide commands to put the application that is using the Primary data volumes hr_dv01 and hr_dv02 in quiesce mode.

   In the `unquiesce` script, provide commands to resume the application or take it out of the quiesce mode.

3. Create the prefreeze and onfreeze scripts and copy them to the`/etc/vx/vvr/ibc_scripts/dss_app` directory on the Secondary host. In the prefreeze script, include the following commands to add snapshot plexes to the Secondary data volumes `hr_dv01` and `hr_dv02`:

   ```
   #!/bin/sh
   /usr/sbin/vxsnap -g hrdg prepare hr_dv01
   /usr/sbin/vxsnap -g hrdg prepare hr_dv02

   /usr/sbin/vxassist -g hrdg make SNAP-hr_dv01 datavolume-length
   /usr/sbin/vxassist -g hrdg make SNAP-hr_dv02 datavolume-length

   /usr/sbin/vxsnap -g hrdg prepare SNAP-hr_dv01
   /usr/sbin/vxsnap -g hrdg prepare SNAP-hr_dv01
   ```

   In the onfreeze script, include the following command to take the snapshot of the Secondary volumes:

   ```
   /usr/sbin/vxrvg -g vvrdg -F -P SNAP snapshot hr_rvg
   ```

4. Run the following `vradmin ibc` command from any host in the RDS:

   ```
   # vradmin -g hrdg ibc hr_rvg dss_app london
   ```

5. On the Secondary, use the snapshot data volumes `SNAP-hr_dv01` and `SNAP-hr_dv02` to run the DSS application, that is, for off-host processing.
6. When the DSS application completes, reattach the snapshot plexes to the data volumes by issuing the following command on the Secondary host london:

   ```
   # vxrvg -g hrdg snapback hr_rvg
   ```

   The snapback destroys the SNAP volumes and reattaches the snapshot plexes to their original volumes. If you have enabled FR on these volumes, the reattach is faster.

Example - Backing up using the snapshot feature and the vradmin ibc command

This example shows backing up using the snapshot feature and the `vradmin ibc` command.

To back up using the snapshot feature and the `vradmin ibc` command

1. Create the following directory on both Primary and Secondary hosts:

   ```
   /etc/vx/vvr/ibc_scripts/dss_app
   ```

2. Create the `quiesce` and `unquiesce` scripts and copy them to the `/etc/vx/vvr/ibc_scripts/dss_app` directory on the Primary host.

   In the `quiesce` script, provide commands to put the application that is using the Primary data volumes hr_dv01 and hr_dv02 in quiesce mode.

   In the `unquiesce` script, provide commands to resume the application or take it out of the quiesce mode.

3 Create the prefreeze and onfreeze scripts and copy them to the `/etc/vx/vvr/ibc_scripts/dss_app` directory on the Secondary host. In the prefreeze script, include the following commands to add snapshot plexes to the Secondary data volumes `hr_dv01` and `hr_dv02`:

```
#!/bin/sh
/usr/sbin/vxsnap -g hrdg prepare hr_dv01
/usr/sbin/vxsnap -g hrdg prepare hr_dv02

/usr/sbin/vxassist -g hrdg make SNAP-hr_dv01 datavolume-length
/usr/sbin/vxassist -g hrdg make SNAP-hr_dv02 datavolume-length

/usr/sbin/vxsnap -g hrdg prepare SNAP-hr_dv01
/usr/sbin/vxsnap -g hrdg prepare SNAP-hr_dv01
```

In the onfreeze script, include the following command to take the snapshot of the Secondary volumes:

```
/usr/sbin/vxrvg -g vvrdg -F -P SNAP snapshot hr_rvg
```

4 Run the following `vradmin ibc` command from any host in the RDS:

```
# vradmin -g hrdg ibc hr_rvg dss_app london
```

Example - Performing block-level backup of the Secondary data using the vradmin ibc command

This method performs the backup directly from the Secondary data volumes while the replication to Secondary is frozen. Make sure the Secondary SRL is large enough to hold the writes sent from the Primary while the backup is in progress. In this method, the Secondary data volumes are under replication control, and hence you cannot write to them. Use this method only if the backup process does not perform any writes to the Secondary data volumes.

To perform block-level backup of the Secondary data

1. Create the following directory on both Primary and Secondary hosts:

   ```
   /etc/vx/vvr/ibc_scripts/dss_app
   ```

2. Create the `quiesce` and `unquiesce` scripts and copy them to the `/etc/vx/vvr/ibc_scripts/dss_app` directory on the Primary host.

   In the `quiesce` script, provide commands to put the application that is using the Primary data volumes hr_dv01 and hr_dv02 in quiesce mode.

   In the `unquiesce` script, provide commands to resume the application or take it out of the quiesce mode.

3. Create the `onfreeze` script and copy it in the `/etc/vx/vvr/ibc_scripts/dss_app` directory on Secondary host:

   In the `onfreeze` script, include the following commands to perform block-level backup of the Secondary data volumes:

   ```
   #!/bin/sh
   dd if=/dev/vx/rdsk/hrdg/hr_dv01 of=/dev/rmt/0
   dd if=/dev/vx/rdsk/hrdg/hr_dv02 of=/dev/rmt/0
   ```

   Note: This example does not need the `prefreeze` and `postfreeze` scripts.

4. Run the following `vradmin ibc` command from any host in the RDS:

   ```
   # vradmin -g hrdg ibc hr_rvg dss_app london
   ```

Chapter 10

Transferring the Primary role

This chapter includes the following topics:

- About transferring the Primary role
- Migrating the Primary
- About taking over from an original Primary
- Failing back to the original Primary
- About choosing the Primary site after a site disaster or network disruption
- Application availability in the case of a network disruption
- Configuring VCS global clustering so you can choose the Primary site
- Choosing the Primary site after a site disaster or network disruption
- Troubleshooting the primary-elect feature
- Primary-elect configuration limitations

About transferring the Primary role

In a VVR environment, applications can only write to the Primary data volumes. When replication is active, the application cannot write to the Secondary data volumes. To start the applications on the Secondary, you must transfer the Primary role to the Secondary. After transferring the role, you can start the application on the new Primary.

VVR enables you to transfer the Primary role from a healthy or failed Primary using a single command. It also enables you to fail back to the original Primary using a simple set of commands.

VVR offers the following methods to transfer the Primary role:

- See "Migrating the Primary" on page 300.
- See "About taking over from an original Primary" on page 308.
- See "Failing back to the original Primary" on page 316.

Note: If the RDS is configured in a Cluster Server (VCS) environment, use the `hagrp` command to offline and online the corresponding resources. The RVGPrimary agent for VVR can be used to manage the role transfer.

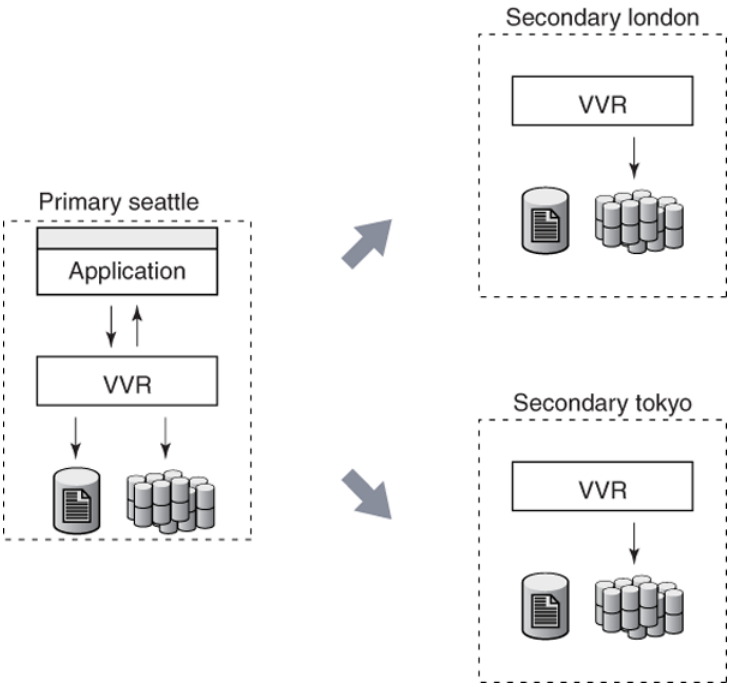
For more information on bringing resources online or offline, see the *Cluster Server Administrator Guide*.

See "Requirements for configuring VVR in a VCS environment" on page 108.

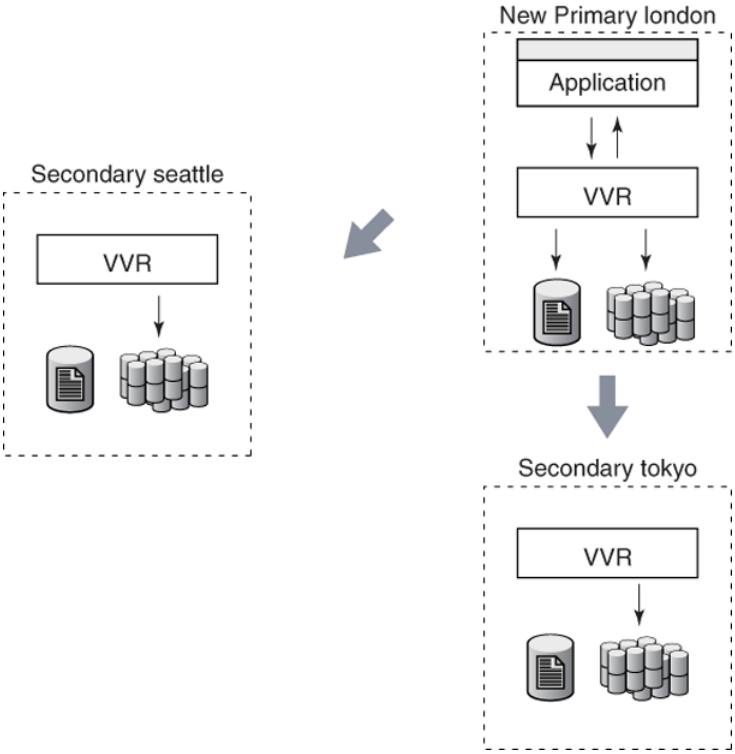
Migrating the Primary

Migration involves transferring a healthy Primary of a Replicated Data Set (RDS) to a Secondary when the application involved in replication is inactive. Migration of a Primary to a Secondary is useful when the Primary must be brought down for reasons such as maintenance or to make the application active on another node.

In the following illustration, the Primary `seattle` is replicating to the Secondary hosts `london` and `tokyo`.



In the following illustration, the Primary role has been migrated from `seattle` to `london` and the new Primary `london` is replicating to `seattle` and `tokyo`. If you had already created RLINKs between london and tokyo when setting up replication, you do not need to manually reconfigure the additional Secondary `tokyo` as a part of the RDS. It will automatically be added as a Secondary of the new Primary `london`.



VVR provides the `vradmin migrate` command to migrate a healthy Primary. The `vradmin migrate` command performs the following functions:

- Migrates the Primary role of an RVG to a Secondary, thus converting the Secondary RVG to the Primary RVG.
- Converts the original Primary of the RDS to Secondary in the RDS.
- Reconfigures the original Primary and the new Primary.

The `vradmin migrate` command restores the original configuration if it fails on any of the hosts during its execution.

Before migrating the Primary role, the `vradmin migrate` command displays a warning and prompts the user to confirm whether or not all the applications using the Primary volumes are stopped. To skip this confirmation, use the `-s` option with the `vradmin migrate` command, which proves useful in scripts.

Prerequisites for migrating the Primary

Observe the following prerequisites:

- The data volumes in the RDS must be inactive, that is, applications that use the Primary data volumes must be stopped.
- All Secondaries must be up-to-date.
- All attached RLINKs must be in the CONNECT state.

To migrate a healthy Primary from any host in an RDS:

```
# vradmin -g diskgroup migrate local_rvgname [newprimary_name]
```

The argument *diskgroup* is the disk group on the local host

The argument *local_rvgname* is the name of the RVG on the local host

The argument *newprimary_name* is the name of the new Primary host, that is, the existing Secondary host. For an RDS with only one Secondary, this argument is optional. Note that the *newprimary_name* argument must be the host name displayed by the `vradmin printrvg` command.

Important notes for migrating the Primary role

The following are important notes about migrating the Primary role:

- We recommend that you set the size of the SRL the same on the Primary and Secondary nodes because any of the Secondary nodes could be later converted to a Primary by using the `vradmin migrate` or the `vradmin takeover` command. If necessary, you can resize the SRL for an existing RVG.
  See "Changing the size of the SRL on the Primary and the Secondary" on page 223.
- We recommend that you configure the Primary and Secondary data volumes with the same names. However, if the names of the data volumes on the Primary and the Secondary do not match, map the names of the Secondary data volumes with the differently named Primary data volumes.
  See "Mapping the name of a Secondary data volume to a differently named Primary data volume" on page 214.
- For an RDS with multiple Secondaries:
  - We recommend that you wait until all the Secondaries are up-to-date before migrating the Primary role. The `vradmin migrate` command fails if all the Secondaries are not up-to-date. If the additional Secondaries were out of date before performing the migrate operation, then you would need to perform a complete synchronization.
  - After successful migration, if you had already created RLINKs between every pair of secondaries while setting up replication, you do not need to manually

reconfigure the additional secondaries as a part of the RDS. Otherwise, you must reconfigure them manually.

See "Example - Migrating the Primary role in a setup with multiple Secondaries" on page 305.

Example - Migrating from a healthy Primary

This example explains how to migrate the original Primary `seattle` to the Secondary host `london`.

Note: Create SRLs of the same size on the Primary and Secondary hosts.

Before migration, the configuration of the RDS looks like this:

	On Primary	On Secondary
Host Name displayed by the vradmin printrvg command	`seattle`	`london`
RVG	`hr_rvg`	`hr_rvg`
RLINK	`rlk_london_hr_rvg`	`rlk_seattle_hr_rvg`

To migrate the Primary RVG hr_rvg to host london:

1. Stop the applications that use the Primary data volumes. For example, if the application is a file system, unmount it.
2. Verify that the Primary RLINK is up-to-date by using the `vxrlink status` command. On the Primary `seattle`, issue the following command:

   ```
   # vxrlink -g hrdg status rlk_london_hr_rvg
   ```

   The `vradmin migrate` command fails if the Primary RLINK is not up-to-date or not in the CONNECT state. It also fails if the data volumes are active.

3. Migrate the Primary RVG `hr_rvg` by typing the following command from any host in the RDS:

```
# vradmin -g hrdg migrate hr_rvg london
```

`london` is the name of the Secondary host displayed by the `vradmin printrvg` command.

4. Restart the application.

Because the application was stopped properly before the migration, an application recovery is not required.

By default, the `vradmin migrate` command enables replication from the new Primary `london`. To start the application before enabling replication, first, issue the `vradmin pauserep` command, start the application, and then resume replication.

After the migration, the configuration of the RDS looks like this:

	On Primary	**On Secondary**
Host Name displayed by the vradmin printrvg command	`london`	`seattle`
RVG	`hr_rvg`	`hr_rvg`
RLINK	`rlk_seattle_hr_rvg`	`rlk_london_hr_rvg`

Example - Migrating the Primary role in a setup with multiple Secondaries

We recommend that you create RLINKs between hosts `london` and `tokyo` when setting up the RDS.

Note: Create SRLs of the same size on the Primary and Secondary hosts.

Before migration, the configuration of the RDS looks like this:

	On Primary	On Secondary	On Secondary
Host Name displayed by the vradmin `printrvg` command	`seattle`	`london`	`tokyo`
RVG	`hr_rvg`	`hr_rvg`	`hr_rvg`
RLINKs	`rlk_london_hr_rvg (active)` `rlk_tokyo_hr_rvg (active)`	`rlk_seattle_hr_rvg (active)` `rlk_tokyo_hr_rvg`	`rlk_seattle_hr_rvg (active)` `rlk_london_hr_rvg`

To migrate the Primary RVG hr_rvg to host london:

1. Stop the applications that use the Primary data volumes. For example, if the application is a file system, unmount it.
2. Verify that the Primary RLINKs are up-to-date by using the `vxrlink status` command. On the Primary `seattle`, issue the following commands:

   ```
   # vxrlink -g hrdg status rlk_london_hr_rvg
   # vxrlink -g hrdg status rlk_tokyo_hr_rvg
   ```

   The `vradmin migrate` command fails if the Primary RLINKs are not up-to-date or not in the CONNECT state. It also fails if the data volumes are active.
3. Migrate the Primary RVG `hr_rvg` by typing the following command from any host in the RDS:

   ```
   # vradmin -g hrdg migrate hr_rvg london
   ```

4 If you had created RLINKs between the Secondary `london` and the additional Secondary `tokyo`, host `tokyo` is automatically added to the new configuration.

Otherwise, you must manually add `tokyo` as a Secondary to the new Primary `london`. To do this, create RLINKs between `london` and `tokyo` and associate them to the respective RVGs using the following commands.

On host `london`:

```
# vxmake -g hrdg rlink rlk_tokyo_hr_rvg local_host=london \
  remote_host=tokyo remote_rlink=rlk_london_hr_rvg \
  remote_dg=hrdg
 # vxrlink -g hrdg assoc hr_rvg rlk_tokyo_hr_rvg
```

On host `tokyo`:

```
# vxmake -g hrdg rlink rlk_london_hr_rvg local_host=tokyo \
  remote_host=london remote_rlink=rlk_tokyo_hr_rvg \
  remote_dg=hrdg
 # vxrlink -g hrdg assoc hr_rvg rlk_london_hr_rvg
```

By default, the `vxmake rlink` command creates the RLINK with the protocol set to TCP/IP. If necessary, you can change the protocol to UDP/IP.

See “Setting the network transport protocol for a Secondary” on page 149.

5 Start replication to `tokyo` using the following command:

```
# vradmin -g hrdg -f startrep hr_rvg tokyo
```

Note: Ensure that the above command is run before starting the application on the new Primary `london`.

6 Restart the application.

Because the application was stopped properly before the migration, an application recovery is not required.

By default, the `vradmin migrate` command enables replication from the new Primary `london`. To start the application before enabling replication, first, issue the `vradmin pauserep` command, start the application, and then resume replication.

After migration, the configuration of the RDS looks like this:

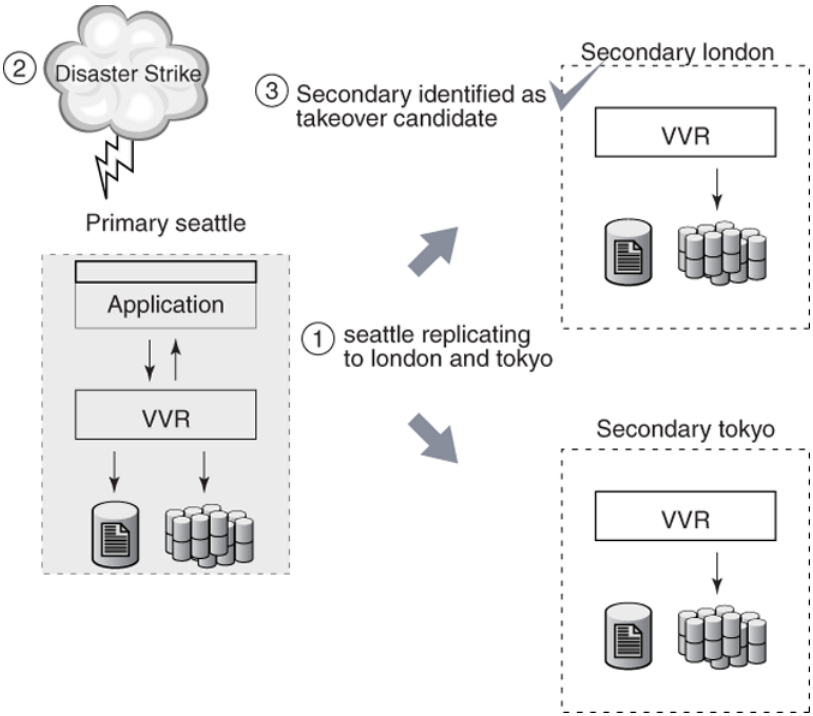
	On Primary	On Secondary	On Secondary
Host Name displayed by the command: vradmin `printrvg`	`london`	`seattle`	`tokyo`
RVG	`hr_rvg`	`hr_rvg`	`hr_rvg`
RLINKs	`rlk_seattle_hr_rvg (active)` `rlk_tokyo_hr_rvg (active)`	`rlk_london_hr_rvg (active)` `rlk_tokyo_hr_rvg`	`rlk_london_hr_rvg (active)` `rlk_seattle_hr_rvg`

About taking over from an original Primary

The takeover procedure involves transferring the Primary role from an original Primary to a Secondary. When the original Primary fails or is destroyed because of a disaster, the takeover procedure enables you to convert a consistent Secondary to a Primary. The takeover of a Primary role by a Secondary is useful when the Primary experiences unscheduled down times or is destroyed because of a disaster.

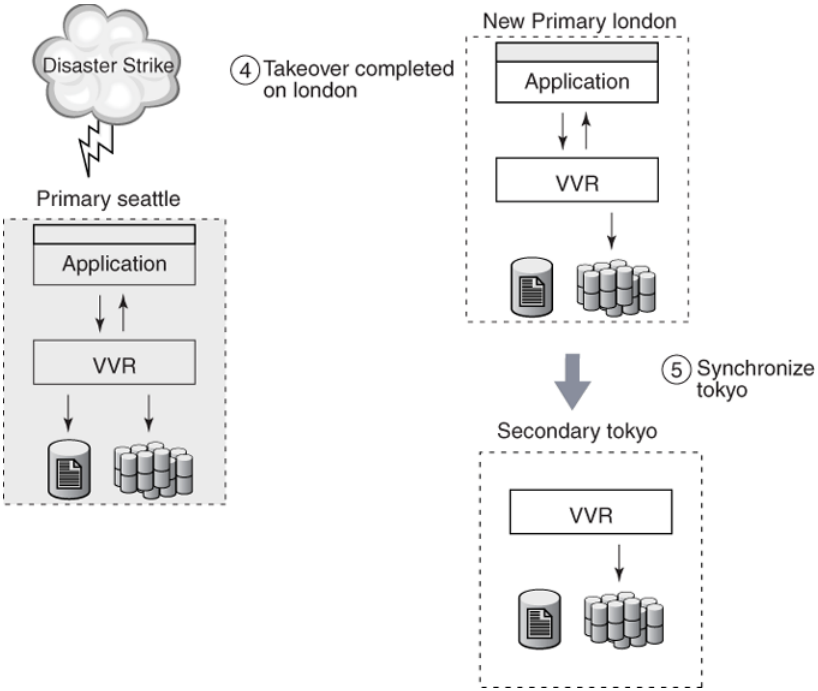
In the following illustration, the Primary `seattle` is replicating to the Secondary hosts `london` and `tokyo` when disaster strikes the Primary `seattle`. The Secondary

`london` has been identified as the Secondary for takeover and will become the new Primary after the takeover is complete.



After the takeover is complete, the Secondary `london` becomes the new Primary. If you had already created RLINKs between `london` and `tokyo` when setting up replication, you do not need to manually reconfigure the additional Secondary `tokyo` as a part of the RDS. It will automatically be added as a Secondary of the new Primary `london`.

You must then synchronize `tokyo` with the new Primary `london` and start replication.



VVR provides the `vradmin takeover` command to transfer the Primary role from an original Primary to a Secondary. This command can be run from the Secondary host only when the Primary is not reachable from the Secondary on which the `takeover` command is to be run. Upon successful completion of the takeover, the Secondary becomes the Primary. VVR displays an error message if you enter the `vradmin takeover` command on the Primary.

For configurations with multiple Secondaries, you can use the `vxrlink updates` command to find the most suitable replacement for the failed Primary.

See "Identifying the most up-to-date Secondary " on page 189.

Important notes about taking over from an original Primary

The following are important notes about taking over from an original Primary:

- The Secondary that is to become the new Primary must be consistent before taking over the Primary role. Any consistent Secondary can be selected to be the new Primary. A Secondary that is undergoing DCM resynchronization or autosync is inconsistent and cannot be used as a target of a takeover operation. Use the `vxprint -l` on the Secondary `RLINK` to check whether the `consistent` flag is set.

- Writes that were not replicated to the Secondary before the Primary became unusable are lost.

  To preserve the data that was not replicated to the Secondary before the Primary became unusable, we recommend that you take a snapshot of the Primary data volumes before starting takeover with fast failback synchronization. The applications can later be started from the snapshot and you can apply the transactions or files that did not get replicated, to the active data.

- The Primary role takeover must be based on Recovery Point Objective (RPO) or the Recovery Time Objective (RTO) depending on the specific business requirement. For example, consider the scenario where the Secondary was behind by 100MB at the time of Primary crash and if the Primary is not expected to be available for another four hours, you will need to decide between having the application up and available on the Secondary or waiting for the Primary to become available after four hours. If it is required that the application must be available immediately, then the data on the Primary that was not yet replicated will be lost. Thus, takeover can result in data loss.

- The Primary role takeover is intended to support disaster recovery applications. Only a limited number of error scenarios prior to loss of the Primary node can prevent a successful takeover. These error scenarios leave the Secondary RVG in an inconsistent state and prevent takeover. All such scenarios involve a hardware failure of a Secondary data volume or Secondary SRL, and as a result, the Secondary RVG will be in an inconsistent state. The chances of such an occurrence are reduced if these volumes are configured (locally) as mirrored volumes.

  Note: The takeover procedure does not guarantee that the new Primary and any additional Secondary RVGs have identical contents. The remaining Secondaries must be completely synchronized with the new Primary.

- We recommend that you set the size of the SRL the same on the Primary and Secondary nodes because any of the Secondary nodes could be later converted to a Primary by using a `migrate` or `takeover` command.

- Each data volume on the new Primary must have a Data Change Map (DCM) associated with it.

The `vradmin takeover` command performs the following functions on the RDS to which the original Primary belongs:

- Converts the Secondary RVG to Primary RVG.

- Enables the fast failback feature on the new Primary, which speeds up failback to the original Primary when it recovers. The fast failback feature is described in the next section.

About fast failback

The applications are started on the new Primary after the takeover is complete. The fast failback feature uses failback logging for incremental synchronization of the original Primary with the new Primary. Fast failback works by keeping track of the incoming writes to the new Primary and the writes to the original Primary that did not reach the Secondary before the original Primary failed. Based on the tracked writes, data can be transferred to the original Primary after it recovers. This eliminates the need to completely resynchronize the original Primary with the new Primary after the original Primary recovers; only the changed blocks are resynchronized. Fast failback uses DCMs on the new Primary to track the changed blocks.

To enable fast failback, each data volume on the Secondary must have an associated DCM. The `takeover` command enables fast failback on the new Primary by activating the DCM. The DCM is later used to synchronize the data volumes on the original Primary with the data volumes on the new Primary.

When the original Primary recovers, it needs to be synchronized with the new Primary by playing back the DCM on the new Primary. To receive the missing updates, the original Primary must first be converted to a Secondary. The new Primary can then begin DCM playback to the original Primary. This process can be initiated automatically upon recovery of the original Primary by using the `-autofb` option to the `takeover` command, or it can be initiated manually at some later time by using the `vradmin fbsync` command.

When you use the `-autofb` option with the `vradmin takeover` command, it automatically synchronizes the original Primary when it becomes available. The `-autofb` option converts the original Primary to a Secondary after it comes up and also uses the DCM to synchronize the data volumes on the original Primary using fast failback. The `-autofb` option can be used only if each Secondary data volume has an associated DCM.

If you prefer not to have the original Primary automatically converted to a secondary upon reboot, the process can be performed manually using the `vradmin fbsync` command.

To change the role from Secondary to Primary without enabling fast failback, use the `-N` option with the `vradmin takeover` command. Use the `-N` option if you are sure that the original Primary will not recover or if most of the data on the Primary is going to change while the Primary is down. When performing `vradmin takeover` with the `-N` option the command automatically detaches the RLINKs from the old

Primary to the new Primary. This requires either difference-based synchronization (`vradmin syncrvg`) or full synchronization of the data volumes on the original Primary.

Taking over from an original Primary

To take over from the Primary RVG `hr_rvg` on host `seattle` to the Secondary RVG `hr_rvg` on host `london`, make sure that the data volumes on the Secondary have associated DCMs. Use the following command to check that the `LOGONLY` attribute is set for the data volumes:

```
# vxprint -g hrdg -ht hr_rvg
```

We recommend that you use the fast failback synchronization method to synchronize the original Primary with the new Primary.

See "Failing back using fast failback synchronization" on page 317.

For an RDS with multiple Secondaries, after take over, VVR automatically reconfigures any additional Secondaries as Secondaries of the new Primary, if RLINKs had been created between the Secondaries. Otherwise, you must manually reconfigure them.

See "Example - Taking over from an original Primary in a setup with multiple Secondaries" on page 314.

To take over an original Primary with fast failback enabled (default), issue the following command on the Secondary that is to take over the Primary role:

```
# vradmin -g diskgroup takeover local_rvgname
```

The argument *`diskgroup`* is the disk group on the local host.

The argument *`local_rvgname`* is the name of the RVG on the local host.

Example -Taking over from an original Primary

In this example the Primary host `seattle` has failed. This example explains how to take over from the original Primary host `seattle` to the Secondary host `london`. The Secondary host `london` is converted to the new Primary. The disk group name is `hrdg`.

To take over from the Primary seattle to Secondary RVG hr_rvg on host london:

1. Make sure that the Secondary is consistent by using the following command to check that the `consistent` flag is set:

   ```
   # vxprint -l rlink_name
   ```

2. Make sure that the data volumes on the Secondary have associated DCMs.

   ```
   # vxprint -g hrdg -ht hr_rvg
   ```

3. Make the Secondary RVG `hr_rvg` the new Primary RVG by typing the following command on the Secondary `london`:

   ```
   # vradmin -g hrdg takeover hr_rvg
   ```

   The `vradmin takeover` command enables fast failback.

4. Verify whether fast failback is enabled by typing the following command on the Secondary `london`:

   ```
   # vxprint -l rlink_name
   ```

   If fast failback is enabled, the `dcm_logging` flag is set.

5. Start the application on the new Primary `london`. Starting the applications on the new Primary after a takeover may require an application recovery to be run.

Example - Taking over from an original Primary in a setup with multiple Secondaries

We recommend that you create RLINKs between the hosts `london` and `tokyo` when setting up the RDS.

In this example the Primary host `seattle` has failed. The example explains how to take over from the original Primary host `seattle` to the Secondary host `london`. This example also explains how to start replication from the new Primary `london` to the additional Secondary `tokyo`.

To take over from the Primary `seattle` to Secondary RVG on host `london`

1. Make sure that the Secondary is consistent by using the following command to check that the `consistent` flag is set:

```
# vxprint -l rlink_name
```

2. Make sure that the data volumes on the Secondary have associated DCMs.

```
# vxprint -g hrdg -ht hr_rvg
```

3. Make the Secondary RVG `hr_rvg` the new Primary RVG by typing the following command on the Secondary `london`:

```
# vradmin -g hrdg takeover hr_rvg
```

   The `vradmin takeover` command enables fast failback.

4. Verify whether fast failback is enabled by typing the following command on the Secondary `london`:

```
# vxprint -l rlink_name
```

   If fast failback is enabled, the `dcm_logging` flag is set.

5. If you had created RLINKs between the Secondary `london` and the additional Secondary `tokyo`, host `tokyo` is automatically added to the new configuration.

   Otherwise, you must manually add `tokyo` as a Secondary to the new Primary `london`. To do this, create RLINKs between `london` and `tokyo` and associate them to the respective RVGs using the following commands.

   On host `london`:

```
# vxmake -g hrdg rlink rlk_tokyo_hr_rvg local_host=london \
  remote_host=tokyo remote_rlink=rlk_london_hr_rvg \
  remote_dg=hrdg
# vxrlink -g hrdg assoc hr_rvg rlk_tokyo_hr_rvg
```

   On host `tokyo`:

```
# vxmake -g hrdg rlink rlk_london_hr_rvg local_host=tokyo \
  remote_host=london remote_rlink=rlk_tokyo_hr_rvg \
  remote_dg=hrdg
 # vxrlink -g hrdg assoc hr_rvg rlk_london_hr_rvg
```

6 By default, the `vxmake rlink` command creates the RLINK with the protocol set to TCP/IP. If necessary, change the protocol to UDP/IP.

See "Setting the network transport protocol for a Secondary" on page 149.

7 Even after takeover, the RLINK from `tokyo` to the original primary `seattle` still remains attached. Detach this RLINK using the following command on the new Primary `london` or on the Secondary `tokyo`:

```
# vradmin -g hrdg stoprep hr_rvg tokyo
```

8 On the new Primary `london`:

- Synchronize the data volumes in the Secondary RVG `hr_rvg` on `tokyo` with the data volumes in the original Primary RVG `hr_rvg` using the difference-based synchronization and Storage Checkpoint. To do this, use the following command on any host in the RDS:

  ```
  # vradmin -g hrdg -c checkpt syncrvg hr_rvg tokyo
  ```

  The `-c` option when used with the `vradmin syncrvg` command automatically starts a Storage Checkpoint with the specified name, `checkpt`, in this example. After the data volumes are synchronized, the Storage Checkpoint is ended.

- Start replication to `tokyo` using the Storage Checkpoint created above:

  ```
  # vradmin -g hrdg -c checkpt startrep hr_rvg tokyo
  ```

9 Start the application on the new Primary `london`. Starting the applications on the new Primary after a takeover may require an application recovery to be run.

Failing back to the original Primary

After an unexpected failure, a failed Primary host might start up to find that one of its Secondaries has been promoted to a Primary by a takeover. This happens when a Secondary of this Primary has taken over the Primary role because of the unexpected outage on this Primary. The process of transferring the role of the Primary back to this original Primary is called failback.

VVR provides the following methods to fail back to the original Primary:

- See "Failing back using fast failback synchronization" on page 317.
- See "Failing back using difference-based synchronization" on page 323.

Note: We recommend that you use the fast failback synchronization method.

Fast failback versus difference-based synchronization

In the case of fast failback, the data blocks that changed while the original Primary was unavailable are tracked using the DCM for each volume. Difference-based synchronization computes MD5 checksums for a fixed size data block on the Primary and Secondary data volumes, compares it, and then determines whether this data block needs to be transferred from the Primary data volume to the Secondary data volume. The fast failback feature is recommended over the difference-based synchronization for the following reasons:

- For difference-based synchronization, all the blocks on all the Primary and Secondary data volumes are read; in the case of fast failback, only the blocks that changed on the new Primary are read and hence the number of read operations required is smaller.
- For difference-based synchronization, the differences are determined by computing and comparing checksum of each of the data chunks on the Secondary and Primary; in the case of fast failback, there is no need to compute checksum because the differences are tracked as they happen, which makes fast failback faster.

The following sections describe each of the above methods for failing back to the original Primary.

Failing back using fast failback synchronization

We recommend that you use the fast failback synchronization method. This procedure assumes that the fast failback feature was enabled on the new

Primary when takeover was performed. Failing back to the original Primary using fast failback involves the following steps:

1. Converting the original Primary to an acting Secondary, as shown in the *`vxprint -l rvgname`* output, and replaying the DCM or SRL of the original Primary to set bits in the DCM of the new Primary. This is performed automatically when the Primary recovers, unless fast failback was disabled during the takeover.

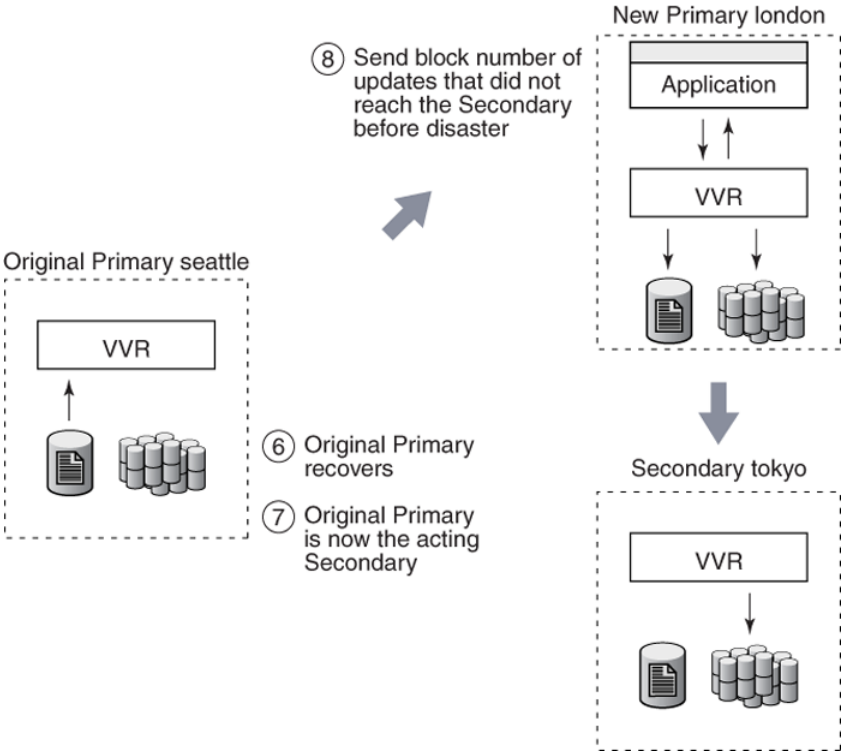
   It is possible that the Primary and Secondary data volumes are not up-to-date because all updates to the original Primary might not have reached the Secondary before the takeover. The failback process takes care of these writes by replaying the SRL or DCM of the original Primary. After the original Primary detects that a takeover has occurred, the new Primary uses information in the DCM or SRL of the original Primary to set bits in its DCM for any blocks that changed on the original Primary before the takeover. You can use the `vxrlink status` command to monitor the progress of the DCM replay.

2. Converting the original Primary to Secondary and synchronizing the data volumes on the original Primary with the data volumes on the new Primary using the `vradmin fbsync` command. This command replays the failback log to synchronize the data volumes. The blocks that changed on the original Primary are resynchronized with the new Primary after the DCM of the new Primary is replayed. During the resynchronization, the data from the data volumes on the new Primary is transferred to the data volumes on the original Primary.

   This step is not required if the `-autofb` option was used at the time of the takeover. The data on the original Primary data volumes is inconsistent for the duration of the replay. To keep a consistent copy of the original Primary data, take a snapshot of the data volumes before starting the replay. When using the `vradmin fbsync` command, you can also specify the `cache` or the `cachesize` option so that a space-optimized snapshot of the original Primary data volumes is automatically created.

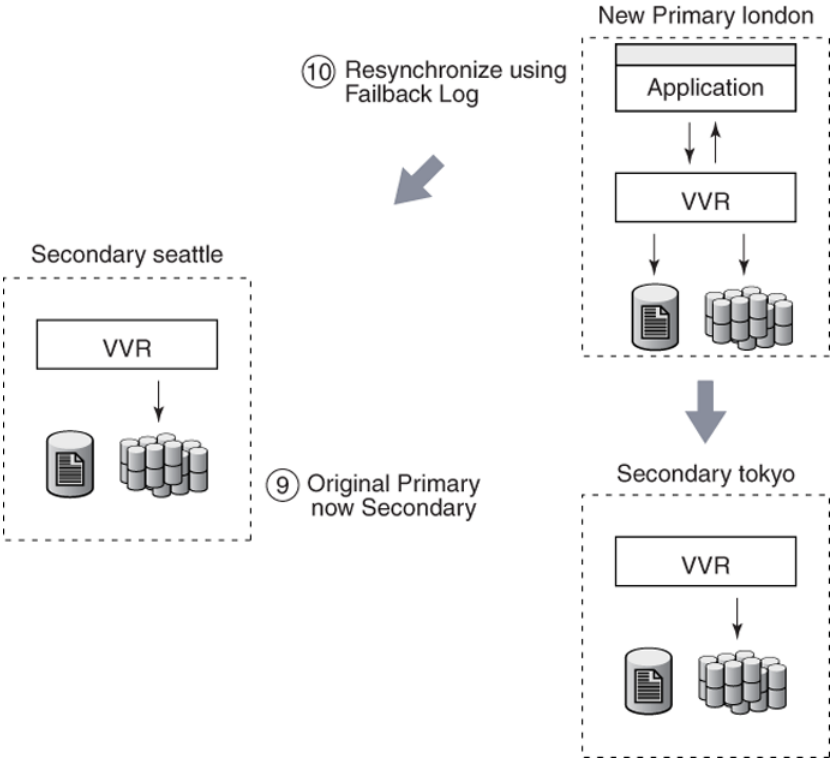
3. Migrating the Primary Role back to the original Primary and starting replication.

In the following illustration, the original Primary `seattle` has recovered and is now the acting Secondary. The new Primary `london` uses information in the DCM or SRL of the original Primary to set bits in its DCM for any blocks that changed on the original Primary before the takeover.



In the following illustration, the fast failback feature was enabled on the new Primary `london` when takeover was performed.

The original Primary `seattle` is being resynchronized using the failback log.



Example - Failing back to the original Primary using fast failback

In this example, the Primary host `seattle` has restarted after an unexpected failure. After the failure, the original Primary `seattle` was taken over by the Secondary host `london`. Each data volume on the Secondary `london` has a Data Change Map (DCM) associated with it. As a result, fast failback is enabled on `london`.

An application is running on `london` and incoming writes are being logged to its DCM. This example shows how to fail back to the original Primary `seattle` using the fast failback feature.

To fail back to the original Primary seattle using fast failback

1. Examine the original Primary and make sure you want to convert the original Primary to Secondary.

2. Convert the original Primary to Secondary and synchronize the data volumes in the original Primary RVG `hr_rvg` with the data volumes on the new Primary RVG `hr_rvg` on `london` using the fast failback feature. To synchronize the Secondary using fast failback, type the following command on the new Primary `london` or the original Primary `seattle`:

   ```
   # vradmin -g hrdg [-wait] fbsync hr_rvg \
      [cache=cacheobj | cachesize=size]
   ```

   When the synchronization completes, go to the next step. You can check the status of the synchronization using the `vxrlink status` command. The `-wait` option with the `vradmin fbsync` command can also be used to wait for the completion of the synchronization process.

   The *`cache`* attribute specifies a name for the precreated cache object, on which the snapshots for the volumes in the specified RVG will be created. Before using the cache attribute, you must create the *`cache`* object.

   See "Preparing the RVG volumes for snapshot operation" on page 253.

   The *`cachesize`* attribute specifies a default size for the cache object with respect to the source volume. You can specify only one of these attributes at one time with the `vradmin fbsync` to create one cache object for each snapshot.

   The parameters *`cache`* and *`cachesize`* are optional. If you do not specify either of these parameters then the `vradmin fbsync` will convert the original Primary to a Secondary and synchronize the data volumes on the original Primary with the data volumes on the new Primary, without creating the snapshots.

   This step is not required if the `-autofb` option was used at the time of takeover

3. At a convenient time, stop the application on the new Primary.

4 Migrate the Primary role from the new Primary host `london` to the original Primary host `seattle` by typing the following command on any host in the RDS:

```
# vradmin -g hrdg migrate hr_rvg seattle
```

Replication from the original Primary `seattle` to the original Secondary `london` is started by default.

5 Restart the application on the original Primary `seattle`. Because the application was stopped properly before the migration, an application recovery is not required.

Example - Failing back to the original Primary using fast failback in a multiple Secondaries setup

In this example, the setup consists of two Secondaries, `london` and `tokyo`. The Primary host `seattle` has restarted after an unexpected failure. After the failure, the original Primary `seattle` was taken over by the Secondary host `london`. Each data volume on the Secondary `london` has a Data Change Map (DCM) associated with it. As a result, fast failback is enabled on `london`.

If you had created RLINKs between the hosts `london` and `tokyo` when setting up the replication, you do not need to manually reconfigure the additional Secondary `tokyo` as a part of the RDS. It will automatically be added as a Secondary of the new Primary `london`.

An application is running on `london` and incoming writes are being logged to its DCM.

This example shows how to fail back to the original Primary `seattle` using the fast failback feature.

To fail back to the original Primary using fast failback in a setup with multiple Secondaries

1 Fail back to the original Primary using fast failback as usual except do not restart the application on the original Primary yet.

See "Example - Failing back to the original Primary using fast failback" on page 320.

2 After migration, you must synchronize the additional secondary `tokyo` with the original Primary `seattle`.

On the original Primary `seattle`:

- Synchronize the data volumes in the Secondary RVG `hr_rvg` on `tokyo` with the data volumes in the original Primary RVG `hr_rvg` using the difference-based synchronization and Storage Checkpoint. To do this, use the following command on any host in the RDS:

  ```
  # vradmin -g hrdg -c checkpt syncrvg hr_rvg tokyo
  ```

  The `-c` option when used with the `vradmin syncrvg` command automatically starts a Storage Checkpoint with the specified name, `checkpt`, in this example. After the data volumes are synchronized, the Storage Checkpoint is ended.

- Start replication to `tokyo` using the Storage Checkpoint created above:

  ```
  # vradmin -g hrdg -c checkpt startrep hr_rvg tokyo
  ```

3. Restart the application on the original Primary `seattle`. Because the application was stopped properly before the migration, an application recovery is not required.

Failing back using difference-based synchronization

Failing back to the original Primary using difference-based synchronization involves the following steps:

To fail back using difference-based synchronization

- Converting the original Primary to a Secondary of the new Primary.
- Synchronizing the data volumes on the original Primary with the data volumes on the new Primary using difference-based synchronization using Storage Checkpoint.
- Starting replication to the Secondary (original Primary) using Storage Checkpoint.
- Migrating the Primary Role to the original Primary and starting replication by default.

Converting an original Primary to a Secondary

VVR provides the `vradmin makesec` command to convert the original Primary to a Secondary. This command is only needed if fast failback was not enabled when the original takeover was performed. If fast failback was enabled, the original Primary will automatically be converted to a Secondary when DCM playback begins. Note that this command can be run only from the original Primary host where one of its original Secondaries has taken over the Primary role.

You can issue the `vradmin makesec` command in the failback procedure only if fast failback has not been enabled when taking over from the original Primary. Run this command when the original Primary restarts. Stop the application if it restarts automatically when the Primary restarts. The `vradmin makesec` command converts the original Primary to a Secondary RVG.

Tip: Before using the `vradmin makesec` command make sure you close all the applications running on the original Primary's data volumes. Also ensure that none of the data volumes are open.

The `vradmin makesec` command fails if the Secondary data volumes are not up-to-date or if there are some applications still running on the failed Primary's data volumes. Use the `-f` option to convert a failed Primary to a Secondary even when the Secondary data volumes are not up-to-date. If any of the failed Primary data volumes are open or have applications running on them, then using the `vradmin makesec` command with the `-f` option will fail. To proceed with the `vradmin makesec` command, first close the volumes or stop the applications as required.

To convert an original Primary to a Secondary:

```
# vradmin -g diskgroup makesec local_rvgname newprimary_name
```

The argument *`diskgroup`* is the disk group on the local host.

The argument *`local_rvgname`* is the name of the RVG on the local host, that is, the original Primary and represents its RDS.

The argument *`newprimary_name`* is the name of the new Primary host, that is, the previous Secondary host. Note that the *`newprimary_name`* argument must be the same as the host name displayed with the Primary-Primary configuration error in the output of the `vradmin -l printrvg` command.

Example - Failing back to the original Primary using difference-based synchronization

In this example, the Primary host `seattle` has restarted after an unexpected failure. After the failure, the original Primary `seattle` has been manually taken over by the Secondary host `london`. This example shows how to fail back to the original Primary `seattle` using difference-based synchronization.

See "Synchronizing volumes using difference-based synchronization" on page 200.

To fail back to the original Primary seattle using difference-based synchronization

1. Make the original Primary RVG `hr_rvg` on `seattle` the Secondary RVG of the new Primary `london` by typing the following command on the original Primary `seattle`:

   ```
   # vradmin -g hrdg makesec hr_rvg london
   ```

2. Synchronize the data volumes in the original Primary RVG `hr_rvg` with the data volumes in the new Primary RVG `hr_rvg` on `london` using the difference-based synchronization and Storage Checkpoint. To synchronize the Secondary based on differences using a Storage Checkpoint, type the following command on any host in the RDS:

   ```
   # vradmin -g hrdg -c checkpt_presync syncrvg hr_rvg seattle
   ```

3. Stop the application on the new Primary `london`.
4. Start replication to the Secondary RVG (original Primary) `hr_rvg` on `seattle` from the new Primary RVG `hr_rvg` on `london` using the Storage Checkpoint by typing the following command on any host in the RDS:

   ```
   # vradmin -g hrdg -c checkpt_presync startrep hr_rvg seattle
   ```

5. Migrate the Primary role from the new Primary host `london` to the original Primary host `seattle` by typing the following command on any host in the RDS:

   ```
   # vradmin -g hrdg migrate hr_rvg seattle
   ```

   Replication from the original Primary `seattle` to the original Secondary `london` is started by default.
6. Restart the application on the original Primary `seattle`. Because the application was stopped properly before the migration, an application recovery is not required.

Example - Failing back to the original Primary using difference-based synchronization in a multiple Secondaries setup

This example shows how to fail back to the original Primary `seattle` using the difference-based synchronization feature, in an RDS with multiple Secondaries.

To fail back to the original Primary using difference-based synchronization in a setup with multiple Secondaries

1. Fail back to the original Primary using difference-based synchronization as usual except do not restart the application on the original Primary yet.

   See "Example - Failing back to the original Primary using difference-based synchronization" on page 324.

2. On the original Primary `seattle`:

   - Synchronize the data volumes in the Secondary RVG `hr_rvg` on `tokyo` with the data volumes in the original Primary RVG `hr_rvg` using the difference-based synchronization and Storage Checkpoint. To do this, use the following command on any host in the RDS:

     ```
     # vradmin -g hrdg -c checkpt syncrvg hr_rvg tokyo
     ```

     The `-c` option when used with the `vradmin syncrvg` command automatically starts a Storage Checkpoint with the specified name, `checkpt`, in this example. After the data volumes are synchronized, the Storage Checkpoint is ended.

   - Start replication from `seattle` to `tokyo` using the command:

     ```
     # vradmin -g hrdg -c checkpt startrep hr_rvg tokyo
     ```

3. Restart the application on the original Primary `seattle`. Because the application was stopped properly before the migration, an application recovery is not required.

About choosing the Primary site after a site disaster or network disruption

VCS global clustering monitors and manages the replication jobs and clusters at each site. In the event of a site outage, global clustering controls the shift of replication roles to the Secondary site, bring up the critical applications and redirects client traffic from one cluster to the other.

Before Release 5.1SP1, if there was a disaster at the Primary site or a network disruption, the applications were taken offline on the original Primary and failed over to the Secondary. When the original Primary returned or the network disruption was corrected, you had the following options:

- Manually resynchronize the original Primary with the data from the new Primary, once the original Primary comes back up. The applications are only active on the new Primary site.
- Automatically resynchronize the original Primary with the data from the new Primary, once the original Primary comes back up. The applications are only active on the new Primary site.

Beginning in Release 5.1SP1, you have a third option. Applications can be active on both the original Primary and Secondary sites. After the original Primary returns or the network disruption is corrected, you have the option of specifying which site is the Primary going forward. This option is called the primary-elect feature, and it is enabled through the VCS global clustering.

The key difference between the primary-elect feature and the other options is that if a network disruption occurs, applications continue to run on the Primary site and they are also failed over to the Secondary. This feature lets you maintain application availability on both sites while the network is down.

Note: You cannot use the primary-elect feature and the automated bunker replay feature in the same environment. If you set the `AutoResync` attribute to `2` (to enable the primary-elect feature), the value of the `BunkerSyncTimeOut` attribute must be `0` to disable the automated bunker replay feature. Similarly, if you set the `BunkerSyncTimeOut` attribute to a non-zero value, the `AutoResync` attribute cannot be `2`.

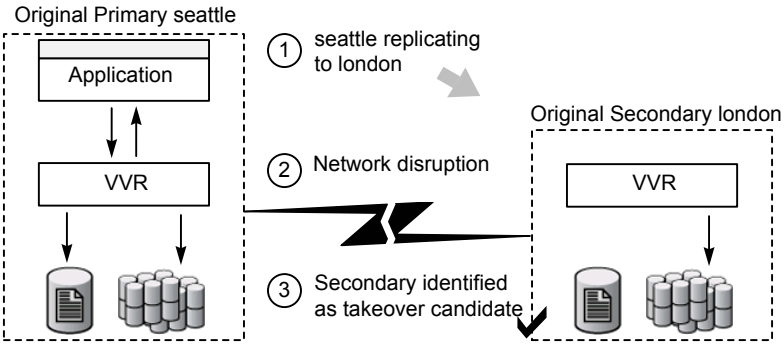
---

for detailed information on configuring and using the primary-elect feature.

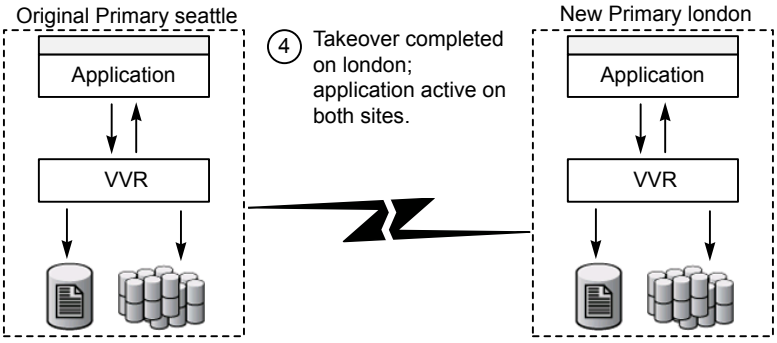
Application availability in the case of a network disruption

The following diagrams illustrate the primary-elect feature. This feature is most useful in the case of the network disruption. It ensures application availability on both sites, even though the network is down.

In this example, the Primary site (`seattle`) is replicating data to the Secondary host (`london`) when a network disruption occurs. `london` has been identified as the Secondary for takeover.



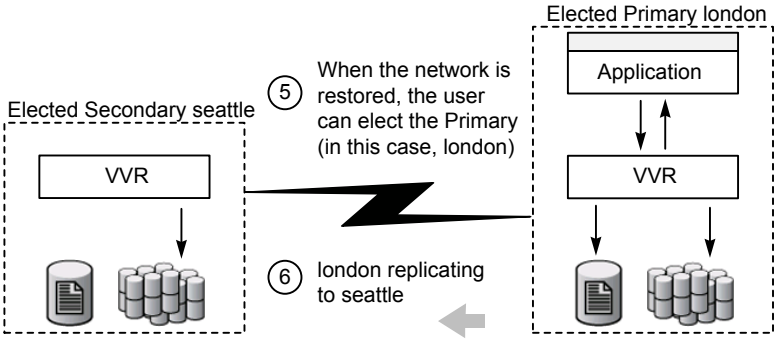
After the takeover, `london` becomes the new Primary. The applications come online on `london`.



Because this event is a network disruption and not a site failure, the original Primary site, `seattle`, is still active. In the primary-elect feature, the applications on the original Primary site are not taken offline. Instead, application data is now being written to both sites, `seattle` and `london`.

Note: In the case of a Primary site failure, the applications are taken offline, even if you choose the primary-elect feature. However, this feature allows you to online the applications on the original Primary outside of VCS control.

When the network connection is restored, you can elect which of the two sites continues as the Primary. In this example the elected Primary is `london` and the elected Secondary is `seattle`.



Any data that was written to the elected Secondary (`seattle`) during the network disruption is lost. Any data that was written to the elected Primary (`london`) is intact.

Configuring VCS global clustering so you can choose the Primary site

To configure VCS global clustering so you can choose the Primary site

1. Make the configuration writable and set the value of the `AutoResync` attribute to `2` for the RVGPrimary resource.

```
# haconf -makerw
# hares -modify RVGPrimary_resource AutoResync 2
```

2. Make sure that the `BunkerSyncTimeout` attribute of the RVGPrimary resource is set to `0` to disable the automated bunker replay feature. You cannot use the automated bunker replay feature and the primary-elect feature in the same environment.

3. Check the value and save the configuration.

```
# hares -value RVGPrimary_resource AutoResync
# haconf -dump -makero
```

Choosing the Primary site after a site disaster or network disruption

After a site disaster or network disruption is resolved, you have the following options:

- Choose the original Primary site as the Primary site going forward.

- Choose the original Secondary site as the Primary site going forward.

This section shows the procedure for each option. In these examples,`seattle` is the Primary site and replicates data to the Secondary site `london`. When `seattle` goes down (or there is a network disruption), `london` takes over as the new Primary. After the problem is resolved and `seattle` is back up, you can elect which site (`seattle` or `london`) is the Primary going forward.

Note: Before you try these procedures, make sure that the AutoResync attribute value is set to 2 to enable the primary-elect feature.

Example - Choosing the original Primary as the Primary going forward

In the following example, the original Primary site, `seattle`, will be chosen as the Primary site going forward. `london` is the elected Secondary site.

To choose the original Primary site as the Primary site going forward

1. On the original Primary, bring the applications in the application service group online so that applications are active on both sites during the primary-elect phase. Bringing the applications online is performed outside of VCS control.
2. On the elected Secondary site, offline the applications in the application service group (*app_grp*). In this example, `london` is the new Secondary site.

   ```
   hagrp -offline app_grp -sys london
   ```

   If you have multiple application service groups, you must repeat this step for each one.
3. On the elected Secondary, specify the original Primary site as the elected Primary site with the `-actionargs` argument of the `hares -action` command. In this example, `seattle` is the name of the global cluster that is associated with the elected Primary site, `seattle`.

   ```
   hares -action RVGPrimary_resource \
   ElectPrimary -actionargs seattle -sys london
   ```

4. Bring the service groups online that you took offline earlier.

Note: If the problem was caused by a network disruption, any application data that was written to `london` during the disruption is lost.

`seattle` is now the Primary site and `london` is the Secondary site. If necessary, bring the applications on `seattle` back online. `seattle` now replicates data to `london`.

Example - Choosing the original Secondary as the Primary going forward

In the following example, the original Secondary site, `london`, will be chosen as the Primary site going forward. `seattle` will be the elected Secondary site.

To choose the original Secondary site as the Primary site going forward

1. On the original Primary, bring the applications in the application service group online so that applications are active on both sites during the primary-elect phase. Bringing the applications online is performed outside of VCS control.
2. On the elected Secondary site, offline the applications in the application service group (*app_grp*). In this example, `seattle` is the new Secondary site.

   ```
   hagrp -offline app_grp -sys seattle
   ```

   If you have multiple application service groups, you must repeat this step for each one.
3. On the elected Secondary, specify the original Secondary site as the elected Primary site with the `-actionargs` argument of the `hares -action` command. In this example, `london` is the name of the global cluster that is associated with the elected Primary site, `london`.

   ```
   hares -action RVGPrimary_resource \
   ElectPrimary -actionargs london -sys london
   ```

4. Bring the service groups online that you took offline earlier.

Note: Any updates made on the elected Secondary during the primary-elect phase will be lost.

`london` is now the Primary site and `seattle` is the Secondary site. If necessary, bring the applications on `london` back online.`london` now replicates data to `seattle`.

Troubleshooting the primary-elect feature

You can troubleshoot the RVGPrimary agent's online agent function, or the VVR ElectPrimary command.

See "Troubleshooting failures in the RVGPrimary online agent function" on page 332.

See " Troubleshooting failures in VVR ElectPrimary command" on page 332.

Troubleshooting failures in the RVGPrimary online agent function

Use the following information to troubleshoot the online agent function for the primary-elect feature:

- Did not prepare the data volumes for space-optimized snapshot operations.
  Check the reason for the failure in the VCS engine log. Manually prepare all the data volumes inside the RVG using the `vxsnap -g dg prepare vol` command. Clear the application service group and bring it back online manually.
- Did not create space-optimized snapshots due to lack of space in the disk group, or any other reason.
  Check the reason for the failure in the VCS engine log. Ascertain and fix the cause for the failure. For example, if you do not have enough disk storage, provision more space in the disk group. Clear the application service group and bring it back online manually. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.
- Did not set the value of the autogrow option to on for the cache objects.
  Check the reason for the failure in the VCS engine log. Manually set 'autogrow' to 'on' for all the cache objects in the RVG, use the `vxcache -g dg set autogrow=on cacheobject` command. Clear the application service group and bring it back online manually.
- Did not convert the secondary RVG to a primary using the `vxrvg -E -F -r makeprimary rvg` command.
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Clear the application service group and bring it back online manually. If the cause of the failure cannot be determined, note the error message and the error code of the failing command from VCS engine logs. Contact Veritas technical support for assistance.

Note: The `-E` option of the vxrvg command is internal and is not intended for customer use.

Troubleshooting failures in VVR ElectPrimary command

Use the following information to troubleshoot failures in VVR ElectPrimary command:

- Did not offline the RVG resource
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Try to manually offline the RVG using the `/opt/VRTS/bin/hares -offline RVG_resource -sys system` command. Re-run the ElectPrimary command.
- Failure for the RVG resource to go offline in one minute
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Ensure that the RVG resource goes offline using the `/opt/VRTS/bin/hares -wait RVG_resource State OFFLINE -sys system` command. Re-run the ElectPrimary command. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.
- Did not restore the data volumes of the RVG from the space-optimized snapshots
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Re-run the ElectPrimary command. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.
- Did not destroy the space-optimized snapshots
  Check the reason for the failure in the VCS engine log. Ascertain the cause of the failure. Manually run the `vxrvg -g dg -P snap_prefix snapdestroy rvg` command to destroy the space-optimized snapshots. The primary election is complete and you do not need to run the ElectPrimary command again. If the failure's cause cannot be determined, note the error message and the error code of the failing command in the VCS engine logs. Contact Veritas technical support for assistance.

Primary-elect configuration limitations

A primary-elect configuration has the following limitations:

- The configuration must consist of one Primary site replicating to one Secondary site. You cannot have multiple VVR Secondary sites.
- The Primary site and the Secondary site must run VVR release 5.1 SP1 or later, and the disk group version must be 160.
- This feature is not supported when you replicate shared disk groups or if you have a bunker deployment.

Note: When an application is active on both sites, VVR cannot guarantee that the data is consistent between them. The application must merge data from both sites to ensure consistency.

Chapter 11

Replication using a bunker site

This chapter includes the following topics:

- Introduction to replication using a bunker site
- Sample bunker configuration
- Setting up replication using a bunker site
- Administering replication using a bunker site
- Using a bunker for disaster recovery
- Replication using a bunker site in a VCS environment
- Removing a bunker
- About bunker commands

Introduction to replication using a bunker site

Volume Replicator (VVR) offers different modes of replication, synchronous and asynchronous.

Replication using a bunker site enables you to combine the advantages of synchronous and asynchronous replication, without the overhead of maintaining two complete copies of your data on two Secondary sites.

Replication using a bunker site maintains a copy of the Primary SRL on a site near the Primary site, known as the bunker site. The copy of SRL can be used to bring the Secondary up-to-date if there is a disaster at the Primary site. A bunker site only requires additional storage for the bunker SRL. The bunker SRL is usually replicated using synchronous mode to provide zero data loss. Ideally, the bunker

site should be in a site sufficiently far away to not be within the same disaster zone as the Primary site, yet close enough to not impede the synchronous update of the bunker SRL. If both the Primary and the bunker are lost, the Secondary must start from a previous point in time. Therefore, choose the bunker site so that the likelihood of this is rare.

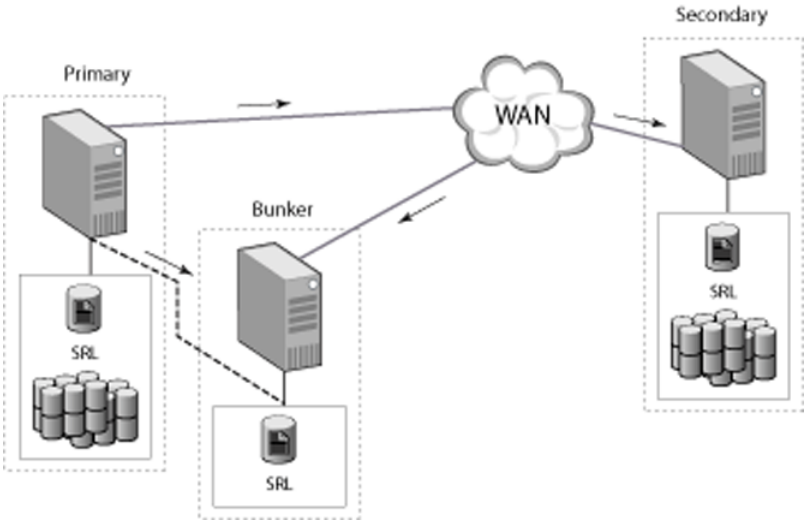
Replication using a bunker site can be performed using an IP network or using direct connectivity to the storage, such as IP over Fibre Channel, Direct Attached Storage (DAS) or Network Attached Storage (NAS). If replication is performed over IP, the Primary host sends writes to a host at the bunker site, known as the bunker host. The bunker host performs writes to the bunker SRL. If replication is done with direct storage, the disk group containing the bunker SRL is imported on the Primary host and the Primary host performs writes to both the bunker SRL and the Primary SRL.

About replication using a bunker site during normal operations

Under normal operating conditions, application writes are logged to the Primary SRL and synchronously replicated to the bunker and any other synchronous Secondaries. Thus, the bunker is in the role of a Secondary. A write is completed to the application as soon as it is logged to the Primary SRL, other synchronous Secondary sites, and the bunker SRL. VVR asynchronously writes the data to the Primary data volume and sends it to the asynchronous Secondary. When the Secondary acknowledges the writes, the SRL header is updated to indicate the status of the Secondary.

In a typical asynchronous replication setup, the network bandwidth is provisioned for the average application write rate. Therefore, the bunker SRL may contain some writes that the application considers complete, but that have not been applied to the Secondary.

For synchronous replication, the network bandwidth must be provisioned for peak application write rate.

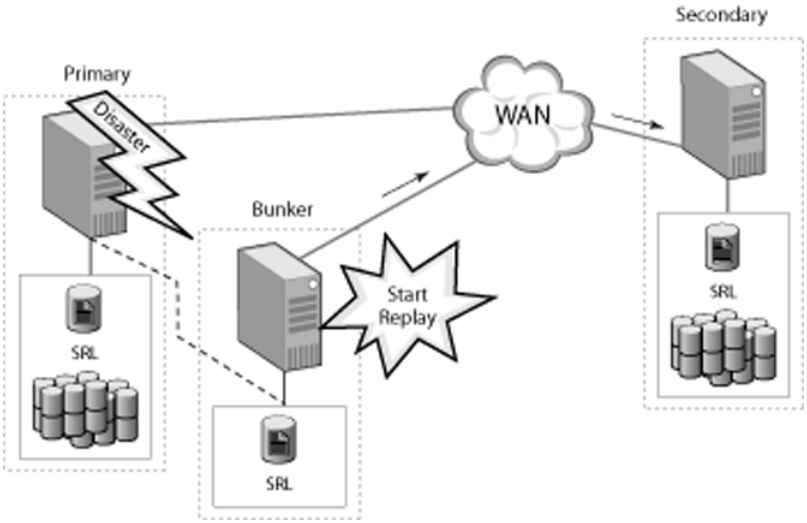


How the bunker site is used for disaster recovery

If the Primary site fails, the Secondary needs to take over. However, the Secondary may be behind the Primary. That is, there may be some writes that are completed to the application but that have not reached the Secondary data volumes; these writes are stored in the SRL on the bunker.

To recover from a disaster on the Primary, you can use the SRL from the bunker site to update the Secondary. You activate the bunker site, which converts it to the Primary role, and then the bunker can replay the pending writes from the bunker SRL to the Secondary.

The procedure is similar whether the bunker setup uses the IP protocol or uses direct storage. However, if the bunker setup uses direct storage, you must first import the disk group containing the bunker SRL onto the bunker host and recover the disk group. In both cases, you activate the bunker to connect the bunker host to the Secondary, and then replay the SRL to the Secondary.



After all of the pending writes are transferred to the Secondary, the Secondary is as up-to-date as the Primary. In normal circumstances, if the entire SRL is replayed, the Secondary can take over the role of the Primary with no data loss. However, in certain failure conditions, the bunker might not be able to bring the Secondary exactly up-to-date with the Primary. For example, if the RLINK between the Primary and the bunker was disconnected prior to the failure at the Primary.

Replication using a bunker site enables you to balance the Recovery Point Objective (RPO) with the Recovery Time Objective (RTO) depending on your needs. In the case of a disaster, completely replaying the bunker SRL to the Secondary provides zero RPO. However, the RTO depends on the time required to replicate pending writes from the bunker SRL to the Secondary site. If the Secondary is far behind the Primary at the time of the disaster, the RTO could be large.

By enabling replication using a bunker site, you can stop the replay after a period of time to recover as much data as possible within a target RTO. For example, if your Secondary is 2 hours behind the Primary, you can replay the full bunker SRL to achieve zero RPO but the RTO would then be about 2 hours. If your target RTO is 1 hour, you could begin bunker replay and then stop the replay after 1 hour.

If you need the application to be immediately available (RTO is zero), you can perform a normal Secondary takeover, without replaying the bunker at all. However, in this case, the pending writes in the bunker SRL are lost. In order to use the bunker SRL to update the Secondary, you must replay the bunker before performing takeover on the Secondary.

Note: The bunker can act in the Secondary role, to receive updates from the Primary, or it can act in the Primary role, to send updates to the Secondary during replay. However, it cannot perform both roles at the same time and therefore does not serve as a relay between a Primary and another Secondary.

After the Secondary has been updated (either the bunker replay has completed or the target RTO is reached and the bunker replay has been stopped), the Secondary can take over the role of the original Primary. The bunker for the original Primary cannot be used as a bunker for the original Secondary. Therefore, another suitable host near the new Primary can be configured as a bunker for the new Primary.

Best practices for setting up replication using a bunker site

It is recommended that you use the following best practices when setting up a bunker site for replication using a bunker site:

- The bunker SRL should be in a site far away to not be within the same disaster zone as the Primary site, yet close enough to not impede the synchronous update of the bunker SRL. Typically, the bunker site should be in the range of 100 kilometers (around 65 miles) or less.
- The bunker site should be provisioned appropriately to support the performance requirements of the application. Since the bunker SRL is usually replicated using synchronous mode to provide zero data loss, writes to the bunker SRL precede all writes to the data volumes in an RVG. The total write performance of an RVG is bound by the total write performance of the bunker SRL. For example, you can dedicate separate disks to bunker SRLs, and if possible dedicate separate controllers to the bunker SRL.

Sample bunker configuration

The examples regarding replication using a bunker site assume the following configuration:

```
# vradmin printrvg
        Replicated Data Set: hr_rvg
        Primary:
                HostName: seattle <localhost>
                RvgName: hr_rvg
                DgName: hrdg
        Secondary:
                HostName: london
                RvgName: hr_rvg
```

```
        DgName: hrdg
Bunker (Secondary):
        HostName: portland
        RvgName: hr_rvg
        DgName: hrdg2
```

The examples in the following sections show how to add a bunker to this configuration.

The bunker host, called `portland`, is added to the `hr_rvg`. The SRL `hr_srl` is present on `portland` in disk group `hrdg2`.

Setting up replication using a bunker site

Setting up replication using a bunker site involves the following steps:

- See "Adding a bunker to an RDS" on page 340.
- See "Changing replication settings for the bunker Secondary" on page 344.
- See "Starting replication to the bunker" on page 346.

Requirements for replication using a bunker site

Observe the following requirements for replication using a bunker site:

- Storage for the bunker SRL at the bunker site.
- Direct connectivity from the Primary to the bunker storage, or IP connectivity from the Primary to the bunker host.
- A system, referred to as the bunker host, with connectivity to the bunker SRL. The bunker host can be a relatively low-tier host, because it does not need to support running the application, but is used only to track and replay the writes to the bunker SRL.
  If the Primary will replicate to the bunker SRL using IP, the bunker host is required at all times.
  If the Primary will replicate to the bunker SRL using the `STORAGE` protocol, the bunker host is required only in the case of a Primary failure, to replay the pending writes from the bunker SRL to the Secondary. However, you must know the IP address at the time of setup.
- A VVR license is required for the bunker host.
- Network connection between bunker and Secondary.
  The network bandwidth over this connection need not be dedicated, because the connection is used only during the recovery process in case of a disaster.

- The bunker SRL must be the same size and the same name as the Primary SRL, or adding the bunker to the RDS fails.
- Replication using a bunker site is not supported in a Cluster Volume Manager (CVM) configuration.

Note: Replication using a bunker site supports cross-platform replication. For example, the bunker node can be running on Linux, while the Primary and Secondary nodes are running on Solaris.

Best practices for setting up replication using a bunker site

When you set up replication using a bunker site, consider the following best practices:

- The bunker SRL should be far enough from the Primary site so that it is not in the same disaster zone. However, the bunker should also be close enough not to impeded the synchronous update of the bunker SRL. Typically, the bunker site should be 100 km (65 miles) or less from the Primary site.
- Configure the bunker SRL to support the performance characteristics required by the application. Because the bunker SRL is usually replicated using synchronous mode to prevent zero data loss, all writes to the data volumes in an RVG are preceded by writes to the bunker SRL. The total write performance of an RVG is limited by the total write performance of the bunker SRL. For example, dedicate separate disks to bunker SRLs, and, if possible, also dedicate separate controllers to the bunker SRL.

Adding a bunker to an RDS

This section describes adding a bunker to an existing RDS. If the RDS already includes a Secondary, adding a bunker does not interrupt replication from the Primary to the Secondary. You can also add the bunker to the RDS before adding the Secondary. Each bunker can support one or more Secondaries. An RDS can only contain one bunker.

A bunker can be configured in one of the following ways:

- Using network (IP) connectivity to the bunker host
  If the bunker host has IP connectivity to the Primary, the Primary replicates to the bunker SRL using standard VVR replication over the network using TCP or UDP protocol.
- Using direct access to bunker storage
  This configuration uses any direct connectivity such as IP over Fiber Channel, Direct Attached Storage (DAS) or Network Attached Storage (NAS) between

the bunker storage and the Primary. In this case, the disk group containing the bunker SRL is imported on the Primary host, and the Primary writes to the bunker storage.

Note: You cannot add data volumes to a bunker.

In addition, bunkers are not supported for Cluster Volume Manager (CVM) in Release 5.1SP1.

To add a bunker when the bunker host is accessible by IP

The steps for adding a bunker are the same whether the Primary uses a private disk group.

1 Create a new disk group, hrdg2, containing only an SRL.

Note: The bunker SRL must be the same size and the same name as the Primary SRL, or adding the bunker fails.

---

2 To add the bunker, type the following command:

```
# vradmin -g hrdg -bdg hrdg2 addbunker hr_rvg seattle portland
```

where `hr_rvg` is the name of the RVG; `seattle` is the name of the Primary; and `portland` is the name of the bunker.

This command creates RLINKs between the bunker and the Primary, and also between the bunker and each Secondary in the RDS.

Note: Assume the following configuration:

```
# vradmin printrvg
Replicated Data Set: hr_rvg
Primary:
     HostName: seattle-v6 <localhost>
     RvgName: hr_rvg
     DgName: hrdg
Secondary:
     HostName: london-v6
     RvgName: hr_rvg
     DgName: hrdg
```

In this configuration, replication is already setup between `london-v6` and `seattle-v6`. The bunker host can be added to the RDS using it's IPv6 address or a host name which resolves to an IPv6 address. For example, the bunker host can be added to the RDS using the following command:

```
# vradmin -g hrdg -bdg hrdg2 addbunker hr_rvg seattle-v6 \
portland-v6
```

where *`hr_rvg`* is the name of the RVG; *`seattle-v6`* is the name of the Primary; and *`portland-v6`* is the name of the bunker.

To add a bunker when the bunker storage is directly accessible

1. Create a new disk group for the bunker, containing only an SRL. We recommend that this disk group has a different name than the main disk group for the RDS. For example, `hrdg2`.

   Note: The bunker SRL must be the same size and the same name as the Primary SRL, or adding the bunker fails.

   The disk group must be available to the Primary and the bunker host. That is, the disk group can be imported on either the Primary or the bunker host.

2. Deport the bunker disk group on the bunker `portland` and import it on the Primary `seattle`.

   If the bunker disk group has the same name as the main disk group name, import it with a temporarily different name on the Primary, using the following command:

   ```
   # vxdg import -t -n newdgname bunkerdgname
   ```

   See "Automating local cluster failover for a bunker" on page 352.

   If the VCS agents for VVR are not configured, you must deport the bunker disk group from the previous logowner and import it on the new logowner node each time the logowner fails over.

3 Add the bunker:

```
# vradmin -g hrdg -bdg hrdg2 addbunker hr_rvg seattle \
  portland protocol=STORAGE
```

where `hr_rvg` is the RVG name; `seattle` is the Primary name; `portland` is the bunker name.

4 To display the configuration, use the following command:

```
# vradmin printrvg
Replicated Data Set: hr_rvg
Primary:
     HostName: seattle <localhost>
     RvgName: hr_rvg
     DgName: hrdg
Secondary:
     HostName: london
     RvgName: hr_rvg
     DgName: hrdg
Bunker (Secondary):
     HostName: portland
     RvgName: hr_rvg
     DgName: hrdg2
```

Changing replication settings for the bunker Secondary

In normal operating conditions, the bunker acts in a Secondary role, and receives writes from the Primary. Like other Secondaries, the bunker Secondary has replication attributes, which determine the behavior of replication between the Primary and the Secondary. When you add a bunker to an RDS, it is configured as a Secondary and the replication attributes are set to the default values. For some of the attributes, the default values for a bunker Secondary are different than the default values for a standard Secondary.

To use values other than the defaults, change the replication settings between the Primary and the bunker Secondary using the `vradmin set` command. Each attribute setting could affect replication and must be set up with care. The replication attributes for a bunker Secondary are the same as the replication attributes for a standard Secondary.

See “About replication settings for a Secondary” on page 87.

The following table provides an overview of the replication attributes in the bunker scenario, including the default settings for a bunker Secondary.

`synchronous`

Sets the mode of replication for the bunker Secondary.

When the synchronous attribute is set to `override`, replication is done synchronously. In this mode, the normal application write is not affected on the Primary when the link between the Primary and the bunker is down. We recommend setting the synchronous attribute to `override` to provide zero data loss under normal conditions. The default is `override`.

When the synchronous attribute is set to `fail`, the application write fails when the link between the Primary and the bunker is down. This ensures that the bunker SRL is always as up-to-date as the Primary SRL and ensures zero RPO after bunker recovery on the Secondary if the Primary site crashes.

To set up the bunker to replicate asynchronously, set the `synchronous` attribute to `off`. In this mode, use latency protection to limit how much the bunker can be behind the Primary.

`srlprot`

Sets SRL protection.

`off` disables SRL protection. If the Primary SRL overflows for the bunker, the Secondary is detached, and the bunker cannot be used to track additional writes from the Primary. The default is `off`.

`override` enables SRL protection. If the Primary and Secondary are connected, new writes are stalled until space becomes available in the SRL. If the Primary and Secondary are disconnected, VVR disables SRL protection and lets the SRL overflow.

Because a bunker does not contain data volumes, a bunker cannot use a DCM to track changes in case of SRL overflow. Therefore, you cannot set the SRL protection to `dcm` or `autodcm` for a bunker.

`protocol`

Indicates the network connectivity between the bunker and Primary.

If the bunker is replicating over IP, the protocol can be set to `UDP` or `TCP`. The default is `TCP`.

If the storage is directly accessible by the Primary, for example, DAS or NAS, set the protocol to `STORAGE`.

`latencyprot`

Sets the latency protection for the bunker. Set the desired RPO value to make sure that the bunker never gets behind by more than this RPO.

`off`, `fail`, or `override`. The default is `off`.

`latency_high_mark`	Set when using latency protection.
`latency_low_mark`	See "About the latencyprot attribute" on page 94.
`packet_size`	Indicates the packet size used by VVR. The packet setting is only specified if the protocol is UDP. The default is 8400.
`bandwidth_limit`	Controls the amount of bandwidth used by VVR for replication to the bunker. The default bandwidth limit is none, indicating that VVR can use any available bandwidth.

Starting replication to the bunker

During normal operation, writes to the Primary SRL are replicated to the SRL on the bunker. If the Primary fails, you can use the bunker SRL to update the Secondary.

To start replication from the Primary to the bunker

- To start replication to the bunker, use the following command:

```
# vradmin -g hrdg -a startrep hr_rvg portland
```

You must use the `-a` option with the `startrep` command to start replication to a bunker. The `-a` option automatically synchronizes the bunker SRL with the Primary SRL. The synchronization starts with the first write still outstanding for any Secondary. Any writes in the Primary SRL that have been updated on all of the Secondaries are not transferred to the bunker SRL. Using the `startrep` command with either `-f` or `-b` options does not attach the RLINK for a bunker.

Note: The bunker SRL does not store Primary Storage Checkpoints; therefore, the bunker does not support attaching or resuming the Secondary from a Storage Checkpoint.

Reinitializing the bunker

If the bunker site falls behind the Secondaries, you can reinitialize the bunker without disrupting replication between the Primary and the Secondary in the RDS. For example, in the case when the bunker site fails and is down for a period of days.

You can detach the bunker RLINK and reattach it with the `vradmin -a startrep` command. This command reconnects the bunker and starts synchronization with the first write still outstanding for any Secondary. The bunker SRL starts receiving writes from this point onwards.

Administering replication using a bunker site

After you start the replication between the Primary and the bunker, administer the replication using the same commands used to administer replication to any other Secondary. For example, `pauserep`, `resumerep`, and `stoprep`.

See "Administering replication" on page 227.

To display the status of replication to a bunker, use the `vradmin repstatus` command.

See "Displaying consolidated replication status" on page 164.

Using a bunker for disaster recovery

The following sections describe how to use the bunker for disaster recovery if the Primary fails.

To use a bunker for disaster recovery

1. Update the Secondary from the bunker.

   See "Updating the Secondary from the bunker" on page 347.

2. After the Secondary is up-to-date, the Secondary can take over the role of Primary.

   See "About taking over from an original Primary" on page 308.

3. When the original Primary recovers, restore the Primary role to the original Primary.

   See "Restoring the original Primary in a bunker setup" on page 349.

Updating the Secondary from the bunker

When disaster strikes and the Primary host goes down, the Secondary can be updated using the bunker. You can update the Secondary from the bunker automatically or manually.

Note: If the Primary SRL has overflowed for a Secondary, or if the Secondary is inconsistent because it is resynchronizing, you cannot use the corresponding bunker SRL to recover the Secondary. Because the bunker does not have data volumes, it cannot use DCM to track overflows.

Updating the Secondary from the bunker automatically

You can automate the failover to the Secondary so that the VCS global clustering can promote the bunker site as the Primary and automatically replay the bunker to the Secondary.

See "About bunker replay in a VCS environment" on page 353.

If you do not configure automatically bunker replay, you must manually update the Secondary from the bunker.

Updating the Secondary from the bunker manually

VVR commands relating to replication using a bunker site can be issued on any VVR host associated to the RDS, except where indicated. The `activatebunker` and `deactivatebunker` commands must be issued on the bunker host.

To update the Secondary from the bunker manually

1. If the bunker is using the `STORAGE` protocol, import the disk group containing the bunker SRL onto the bunker host by issuing the following command on the bunker host:

   ```
   # vxdg -C import hrdg2
   ```

2. Activate the bunker, by issuing the following command on the bunker host:

   ```
   # vradmin -g hrdg2 activatebunker hr_rvg
   ```

   This command converts the bunker RVG from receiving mode (Secondary) to replicating mode (Primary).

   You only need to issue the `activatebunker` command once, even if you are updating multiple Secondaries.

3. Start replication from the bunker host to the Secondary:

   ```
   # vradmin -g hrdg2 -b startrep hr_rvg london
   ```

   This command connects the RLINKs that were created when the bunker was added to the RDS and begins replaying the bunker SRL.

   If you have more than one Secondary using this bunker, repeat the `vradmin startrep` command for each Secondary.

4. Monitor the status of the replication from bunker to Secondary:

   ```
   # vradmin -g hrdg2 repstatus hr_rvg
   ```

5 When the Secondary is up-to-date, stop replication to the Secondary. You can also stop the replication before the replay is finished, for example, if the Primary is restored.

```
# vradmin -g hrdg2 stoprep hr_rvg london
```

6 After the bunker has been used to do the replay and it is not needed for any more replays, the bunker must be deactivated.

Note: Deactivate the bunker only after all the replays from the bunker have been stopped.

To deactivate the bunker, issue the following command on the bunker host:

```
# vradmin -g hrdg2 deactivatebunker hr_rvg
```

You only need to issue this command once.

The Secondary is now up-to-date and can take over as Primary.

See "About taking over from an original Primary" on page 308.

Restoring the original Primary in a bunker setup

In most cases, when the original Primary recovers after a failure, you would want to restore the RDS to the original configuration. In a bunker setup, how you restore the Primary role to the original Primary depends on the status of the bunker replay.

Refer to the indicated method to restore the Primary in the following situations:

- If the original Primary recovers during the replay of a bunker SRL.
  See "Recovering the original Primary during bunker replay" on page 349.
- If the original Primary recovers after the original Secondary has taken over the Primary role.
  See "Failing back to the original Primary" on page 350.

Recovering the original Primary during bunker replay

If the original Primary recovers during the replay of a bunker SRL, the original Secondary has not taken over the Primary role. You can restore operation to the original Primary without completing the replay and the Secondary takeover of the Primary role.

After the bunker is activated and is replaying the bunker SRL, the bunker is acting in the Primary role. If the original Primary recovers and connects while the bunker is active, the RDS shows a configuration error of multiple Primaries in the RDS.

Deactivating the bunker clears the configuration error, and restores the original Primary as the only Primary in the RDS.

To restore the original primary

1. Stop the replication from the bunker to the Secondary:

   ```
   # vradmin -g hrdg2 stoprep hr_rvg
   ```

2. Deactivate the bunker. Issue the following command on the bunker host:

   ```
   # vradmin -g hrdg2 deactivatebunker hr_rvg
   ```

   After the original Primary has recovered and connected, replication resumes from the Primary.

Primary replay to the Secondary resumes from the point in the SRL indicating the last write received by the Secondary. The SRL indicates the writes that have been replayed from the bunker and does not resynchronize those writes. For example, suppose the bunker SRL contained 10 GB when the Primary failed. After 7 GB of the writes were replayed to the Secondary, the Primary recovered. The Primary only needs to synchronize the 3 GB of pending data.

If the bunker setup is using IP protocol, replication from the Primary to the bunker also resumes automatically.

If the bunker storage is connected to the Primary using the `STORAGE` protocol, then the disk group containing the bunker SRL is imported on the bunker host during the replay. When the Primary recovers, this disk group must be made available to the Primary host again.

To make the bunker disk group available to the Primary

1. Deport the disk group from the bunker host:

   ```
   # vxdg deport hrdg2
   ```

2. Import the disk group on the Primary host:

   ```
   # vxdg import hrdg2
   ```

   Replication from the Primary to the bunker now resumes.

Failing back to the original Primary

If the original Secondary has taken over the role of the Primary, fail back the Primary role to the original Primary. Before failing back to the original Primary, make sure all of the writes on the new Primary have been replayed on the original Primary.

If the original Secondary is converted to be the new Primary before all the writes on the bunker SRL are replayed, the failback process synchronizes the remaining writes. The failback process detects the status of the bunker replay, and does not synchronize any writes in the bunker SRL that have already been replayed. For example, suppose the bunker SRL contained 10 GB when the Primary failed. After 7 GB of the writes were replayed to the Secondary, the replay was stopped and the Secondary converted to the new Primary. When the original Primary recovers, the failback process only needs to synchronize the 3 GB of pending data.

See "Failing back to the original Primary" on page 316.

After the failback completes, and the Primary role is restored to the original Primary, you should restart replication to the bunker. The Primary RLINK to the bunker is detached when the original Primary becomes the Secondary of the new Primary as part of the failback process. Therefore, after the original Primary again becomes the Primary, you must reestablish the RLINK between the bunker and the Primary.

See "Restoring the bunker setup after failback to original Primary " on page 351.

Restoring the bunker setup after failback to original Primary

After the original Primary is restored and the failback is complete, restore the bunker setup so that the bunker is again replicating the SRL of the original Primary.

If the bunker storage is connected to the Primary using the `STORAGE` protocol, then the disk group containing the bunker SRL is imported on the bunker host during the replay. When the Primary recovers, this disk group must be made available to the Primary host again.

To restore the bunker setup when using the STORAGE protocol

1. Deport the disk group from the bunker host:

   ```
   # vxdg deport hrdg2
   ```

2. Import the disk group on the Primary host:

   ```
   # vxdg import hrdg2
   ```

3. To deactivate the bunker, if it is not already deactivated, issue the following command on the bunker host:

   ```
   # vradmin -g hrdg2 deactivatebunker hr_rvg
   ```

4. Restart replication from the Primary to the bunker:

   ```
   # vradmin -g hrdg -a startrep hr_rvg portland
   ```

To restore the bunker setup when using the IP protocol

1. Deactivate the bunker, if it is not already deactivated. Issue the following command on the bunker host:

```
# vradmin -g hrdg2 deactivatebunker hr_rvg
```

2. Restart replication from the Primary to the bunker.

```
# vradmin -g hrdg -a startrep hr_rvg portland
```

Replication using a bunker site in a VCS environment

You can configure replication using a bunker site in a VCS environment.

Automating local cluster failover for a bunker

You can set up the VCS agents to automate failover of the bunker when the Primary fails over within a local cluster. This step is not required if the bunker is set up using the IP protocol. For IP protocol, the bunker setup is the same whether the Primary is a single node or a VCS cluster.

When a bunker is set up using the `STORAGE` protocol, the disk group containing the bunker RVG is imported on the Primary host. If the Primary RVG is in a VCS cluster, the bunker RVG must remain online on the same node on which the parent application RVG is online.

In a private disk group environment, the RVG resource handles the failover process. If the host on which the RVG resource is online fails, the RVG resource fails over to another host within the cluster. The RVG resource ensures that the bunker RVG also fails over, so that the bunker RVG continues to be on the same host with the parent RVG.

To set up automated failover of the bunker RVG, specify the bunker RVG, the bunker disk group, and the bunker host using the following attributes of the RVG resource in the application service group or the RVGLogowner agent:

StorageDG	The name of the bunker disk group.
StorageRVG	The name of the bunker RVG.
StorageHostIds	A space-separated list of the hostids of each node in the bunker cluster.

The attributes are the only specific attributes that differ for an RDS containing a bunker. The rest of the configuration for the VCS Agents for VVR is the same as for any other RDS.

See "Requirements for configuring VVR in a VCS environment" on page 108.

Using the StorageHostIds attribute

If the bunker site is a cluster, make sure that the bunker RVG group is never online when the bunker disk group is imported on the Primary cluster. Otherwise, the bunker disk group would be imported on two hosts at the same time, which results in split brain.

If an automatic failover occurs in the Primary cluster, the agent refers to the StorageHostIds attribute to help ensure that the bunker RVG is not imported on both a bunker host and a host in the Primary cluster at the same time. The Primary cluster does not import the bunker disk group if the disk group is already imported on a host in the bunker cluster.

To determine the hostid, issue the following command on each node:

```
# vxdctl list
        Volboot file
        version: 3/1
        seqno:   0.5
        cluster protocol version: 60
        hostid:  vvrnode1
        defaultdg:  pdg
```

If the hostid of a bunker node changes, you must change the StorageHostIds attribute to reflect the new value using the following command:

```
# hares modify RvgPriResName StorageHostIds value
```

About bunker replay in a VCS environment

If a disaster occurs at the Primary site, the data at the Secondary site may not be up-to-date. If the Primary has a bunker site associated with it, you can use the bunker to synchronize the Secondary before making it the Primary site.

The procedure for bunker replay in the VCS environment is the same as in the case of a non-clustered Primary. The RVGPrimary agent handles bunker replay automatically. You do not have to perform any manual steps.

You can choose how you want to use bunker replay by configuring the agent attribute `BunkerSyncTimeOut`.

See "How the RVGPrimary works in a bunker setup" on page 130.

Note: You cannot use the automated bunker replay feature and the primary-elect feature in the same environment. If you set the `BunkerSyncTimeOut` attribute to a non-zero value, the `AutoResync` attribute cannot be `2`. Similarly, if you set the `AutoResync` attribute to `2` (to enable the primary-elect feature), the value of the `BunkerSyncTimeOut` attribute must be `0` to disable the automated bunker replay feature.

For VCS to automatically replay a bunker, the `bunker_target` flag must be set on the RLINK from the Secondary to the bunker node. When the `bunker_target` flag is set on an RLINK, it indicates that this is the RLINK from the Secondary to the bunker node.

When you configure a bunker, the `vradmin addbunker` command sets this flag correctly on the RLINK from the Secondary to the bunker node. However, if you have upgraded from a release prior to Volume Replicator 5.1SP1, and you had a bunker in the configuration before the upgrade, the `bunker_target` flag would not be set on the RLINK from the Secondary the bunker node. In this case, you must run the following command on the Secondary to set the `bunker_target` flag:

```
# vxedit -g dg set bunker_target=on RLINK_from_secondary_to_bunker
```

Removing a bunker

When a bunker is no longer required in an RDS, you can remove the bunker from the RDS.

Note: Before removing a bunker, you must stop replication to the specified bunker, using the `vradmin stoprep` command.

Warning: The operation performed by the `vradmin delbunker` command is irreversible. Adding the bunker back to the RDS initializes the bunker SRL and any previous writes in the bunker SRL are lost.

To remove a bunker, use the following command from any host in the RDS:

```
# vradmin -g dg [-f] delbunker rvgname bunkersitename
```

About bunker commands

The following `vradmin` commands are the only operations supported for the bunker host:

```
vradmin changeip
        addbunker
        delbunker
        set rvg
        startrep
        stoprep
        resumerep
        pauserep
        activatebunker
        deactivatebunker
        printrvg
        printvol
        repstatus
        resizesrl
```

Chapter 12

Troubleshooting VVR

This chapter includes the following topics:

- Recovery from RLINK connect problems
- Recovery from configuration errors
- Recovery on the Primary or Secondary

Recovery from RLINK connect problems

This section describes the errors that may be encountered when connecting RLINKs. To be able to troubleshoot RLINK connect problems, it is important to understand the RLINK connection process.

Connecting the Primary and Secondary RLINKs is a two-step operation. The first step, which attaches the RLINK, is performed by issuing the `vradmin startrep` command. The second step, which connects the RLINKs, is performed by the kernels on the Primary and Secondary hosts.

When the `vradmin startrep` command is issued, VVR performs a number of checks to ensure that the operation is likely to succeed, and if it does, the command changes the state of the RLINKs from DETACHED/STALE to ENABLED/ACTIVE. The command then returns success.

If the command is successful, the kernel on the Primary is notified that the RLINK is enabled and it begins to send messages to the Secondary requesting it to connect. Under normal circumstances, the Secondary receives this message and connects. The state of the RLINKs then changes from ENABLED/ACTIVE to CONNECT/ACTIVE.

If the RLINK does not change to the CONNECT/ACTIVE state within a short time, there is a problem preventing the connection. This section describes a number of possible causes. An error message indicating the problem may be displayed on the console.

- If the following error displays on the console:

  ```
  VxVM VVR vxrlink INFO V-5-1-5298 Unable to establish connection
   with remote host <remote_host>, retrying
  ```

  Make sure that the `vradmind` daemon is running on the Primary and the Secondary hosts; otherwise, start the `vradmind` daemon by issuing the following command:

  ```
  # /etc/init.d/vras-vradmind.sh start
  ```

  For an RLINK in a shared disk group, make sure that the virtual IP address of the RLINK is enabled on the logowner.

- If there is no self-explanatory error message, issue the following command on both the Primary and Secondary hosts:

  ```
  # vxprint -g diskgroup -l rlink_name
  ```

  In the output, check the following:
  The *remote_host* of each host is the same as *local_host* of the other host.
  The *remote_dg* of each host is the same as the disk group of the RVG on the other host.
  The *remote_dg_dgid* of each host is the same as the *dgid* (disk group ID) of the RVG on the other host as displayed in the output of the *vxprint -l diskgroup* command.
  The *remote_rlink* of each host is the same as the name of the corresponding RLINK on the other host.
  The *remote_rlink_rid* of each host is the same as the *rid* of the corresponding RLINK on the other host.
  Make sure that the network is working as expected. Network problems might affect VVR, such as prevention of RLINKs from connecting or low performance. Possible problems could be high latency, low bandwidth, high collision counts, and excessive dropped packets.

- For an RLINK in a private disk group, issue the following command on each host.
  For an RLINK in a shared disk group, use `vxprint -Vl | grep logowner` to find the logowner node, then issue the following command on the logowner on the Primary and Secondary.

  ```
  # ping remote_host
  ```

  Note: This command is only valid when ICMP ping is allowed between the VVR Primary and the VVR Secondary.

After 10 iterations, type Ctrl-C. There should be no packet loss or very little packet loss. To ensure that the network can transmit large packets, issue the following command on each host for an RLINK in a private disk group.
For an RLINK in a shared disk group, issue the following command on the logowner on the Primary and Secondary:

```
# ping -s 8192 remote_host
```

The packet loss should be about the same as for the earlier ping command.

- Issue the `vxiod` command on each host to ensure that there are active I/O daemons. If the output is `0 volume I/O daemons running`, activate I/O daemons by issuing the following command:

  ```
  # vxiod set 10
  ```

- VVR uses well-known ports to establish communications with other hosts. Issue the following command to display the port number:

  ```
  # vxprint -g diskgroup -l rlink_name
  ```

  Issue the following command to ensure that the heartbeat port number in the output matches the port displayed by `vxprint` command:

  ```
  # vrport
  ```

  Confirm that the heartbeat port has been opened by issuing the following command:

  ```
  # netstat -an | grep port-number
  ```

  where `port-number` is the port number being used by the heartbeat server as displayed by the `vrport` command.
  The output looks similar to this:

  ```
  udp      0     0      0,0,0,0:port-number
  ```

- Check for VVR ports on the Primary and Secondary sites.
  Run the `vrport` utility and verify that ports are same at both ends.
  Check whether the required VVR ports are open. Check for UDP 4145, TCP 4145, TCP 8199, and the anonymous port. Enter the following commands:

```
# netstat -an --udp | grep 4145
udp        0      0 :::4145                     :::*
# netstat -an --tcp | grep 4145
tcp        0      0 :::4145                     :::*                    LISTEN
```

```
# netstat -an --tcp | grep 8199
tcp        0      0 :::8199                     :::*                    LISTEN
```

Perform a `telnet` test to check for open ports. For example, to determine if port 4145 is open, enter the following:

```
# telnet <remote> 4145
```

- Use the `netstat` command to check if `vradmind` daemons can connect between the Primary site and the Secondary site:

```
# netstat -an --tcp | grep 8199 | grep ESTABLISHED
tcp        0      0 10.209.85.18:56872      10.209.85.19:8199       ESTABLISHED
tcp        0      0 10.209.87.175:8199      10.209.85.23:42860      ESTABLISHED
tcp        0      0 10.209.87.175:8199      10.209.87.176:9335      ESTABLISHED
```

Recovery from configuration errors

Configuration errors occur when the configuration of the Primary and Secondary RVGs is not identical. Each data volume in the Primary RVG must have a corresponding data volume in the Secondary RVG of exactly the same size; otherwise, replication will not proceed. If a volume set is associated to the RDS, the configuration of the volume set must also match on the Primary and on the Secondary.

Errors in configuration are detected in two ways:

- When an RLINK is attached for the first time, the configuration of the Secondary is checked for configuration errors. If any errors are found, the `attach` command fails and prints error messages indicating the problem. The problem is fixed by correcting the configuration error, and then retrying the attach.
- Changes that affect the configuration on the Primary or Secondary may cause the Secondary to enter the PAUSE state with the `secondary_config_err` flag set. The problem is fixed by correcting the configuration error, and then resuming the RLINK.

Errors during an RLINK attach

During an RLINK attach, VVR checks for errors in the configuration of data volumes. VVR also checks for errors in the configuration of volume sets, if the RDS has a volume set associated to the RVG.

Data volume errors during an RLINK attach

When an RLINK is attached, VVR checks whether for each data volume associated to the Primary RVG, the Secondary RVG has an associated data volume of the same size that is mapped to its counterpart on the Primary. The following example illustrates an attempted attach with every possible problem and how to fix it. Before the `attach`, the Primary has this configuration:

```
TY     Name                Assoc        KSTATE     LENGTH       STATE

rv     hr_rvg         -                 DISABLED     -          EMPTY

rl     rlk_london_hr_rvg  hr_rvg        DETACHED     -          STALE

v      hr_dv01            hr_rvg        ENABLED    12800        ACTIVE

pl     hr_dv01-01         hr_dv01       ENABLED    12800        ACTIVE

sd     disk01-05          hr_dv01-01    ENABLED    12800         -

v      hr_dv02            hr_rvg        ENABLED    12800        ACTIVE

pl     hr_dv02-01         hr_dv02       ENABLED    12880        ACTIVE

sd     disk01-06          hr_dv02-01    ENABLED    12880

v      hr_dv03            hr_rvg        ENABLED    12880        ACTIVE

pl     hr_dv03-01         hr_dv03       ENABLED    12880        ACTIVE

sd     disk01-07          hr_dv03-01    ENABLED    12880        -

v      hr_srl             hr_rvg        ENABLED    12880        ACTIVE

pl     hr_srl-01          hr_srl        ENABLED    12880        ACTIVE

sd     disk01-08          hr_srl-01     ENABLED    12880 0      -
```

The Secondary has the following configuration:

```
TY     Name                Assoc        KSTATE     LENGTH       STATE

rv     hr_rvg              -            ENABLED    -        -   ACTIVE
```

```
rl     rlk_seattle_hr_rvg hr_rvg       ENABLED    -        -   ACTIVE

v      hr_dv01            hr_rvg       ENABLED    12700    -   ACTIVE

pl     hr_dv01-01         hr_dv01      ENABLED    13005    -   ACTIVE

sd     disk01-17          hr_dv01-01   ENABLED    13005    0    -

v      hr_dv2             hr_rvg       ENABLED    12880    -   ACTIVE

pl     hr_dv02-01         vol2         ENABLED    13005    -   ACTIVE

sd     disk01-18          hr_dv02-01   ENABLED    13005    0    -

v      hr_srl             hr_rvg       ENABLED    12880    -   ACTIVE

pl     hr_srl-01          hr_srl       ENABLED    13005    -   ACTIVE

sd     disk01-19          hr_srl-01    ENABLED    13005    0    -
```

Note that on the Secondary, the size of volume `hr_dv01` is small, `hr_dv2` is misnamed (must be `hr_dv02`), and `hr_dv03` is missing. An attempt to attach the Primary RLINK to this Secondary using the `attach` command fails.

```
# vxrlink -g hrdg -f att rlk_london_hr_rvg
```

The following messages display:

```
VxVM VVR vxrlink INFO V-5-1-3614 Secondary data volumes detected
    with rvg hr_rvg as parent:
VxVM VVR vxrlink ERROR V-5-1-6789 Size of secondary datavol hr_dv01
    (len=12700) does not match size of primary (len=12800)
VxVM VVR vxrlink ERROR V-5-1-3504 primary datavol hr_dv02 is not
    mapped on secondary, yet
VxVM VVR vxrlink ERROR V-5-1-3504 primary datavol hr_dv03 is not
    mapped on secondary, yet
```

To fix the problem, issue the following commands on the Secondary:

1. Resize the data volume `hr_dv01`:

```
# vradmin -g hrdg resizevol hr_rvg hr_dv01 12800
```

2. Rename the data volume `hr_dv2` to `hr_dv02`:

```
# vxedit -g hrdg rename hr_dv2 hr_dv02
```

3. Associate a new volume, `hr_dv03`, of the same size as the Primary data volume hr_dv03.

```
# vxassist -g hrdg make hr_dv03 12800
# vxvol -g hrdg assoc hr_rvg hr_dv03
```

Alternatively, the problem can be fixed by altering the Primary to match the Secondary, or any combination of the two. When the Primary and the Secondary match, retry the attach.

On the Primary:

```
# vxrlink -g hrdg -f att rlk_london_hr_rvg
VxVM VVR vxrlink WARNING V-5-1-12397 This
command should only be used if primary and all
secondaries arealready synchronized. If this is not
the case detach the rlink and use autosync or
checkpoint options to attach.
VxVM VVR vxrlink INFO V-5-1-3614 Secondary data
volumes detected with rvg hr_rvg as parent:
VxVM VVR vxrlink INFO V-5-1-6183 vol1: len=12800 primary_datavol=hr_dv01
VxVM VVR vxrlink INFO V-5-1-6183 vol1: len=12800 primary_datavol=hr_dv02
VxVM VVR vxrlink INFO V-5-1-6183 vol1: len=12800 primary_datavol=hr_dv03
```

Volume set errors during an RLINK attach

If a volume set is associated to an RDS, the name of the volume set on the Primary must have the same name as the volume set on the Secondary, and the volume sets must have the same configuration of component volumes.

When an RLINK is attached, VVR checks whether for each volume set associated to the Primary RVG, the Secondary RVG has an associated volume set of the same name. Also, VVR checks whether the volume sets on the Primary and on the Secondary have the same component volumes with the same names, lengths, and indices. (The volume names of the component volumes can be different on the Primary and Secondary if they are mapped, as for independent volumes.) If any of

the component volumes do not exist on the Secondary or have a mismatched name, length, or index, the RLINK attach command fails with the appropriate error message.

See "Volume set configuration errors during modification of an RVG" on page 365.

If the volume set does not exist on the Secondary, but all the component volumes exist on the Secondary with the correct names and lengths, VVR creates the volume set on the Secondary and associates it to the RDS. This does not cause a configuration error.

Errors during modification of an RVG

After the initial setup and attach of a Secondary RLINK, incorrect modifications such as adding, resizing, and renaming volumes can cause configuration errors, if they result in a mismatch between the volumes on the Primary and on the Secondary. If an RVG has an associated volume set, modifications to the volume set can also cause configuration errors. These include incorrectly adding, removing, or renaming component volumes of the associated volume set; adding component volumes with different indices on the Primary and Secondary; or renaming the associated volume set.

When a modification of an RVG causes a configuration error, the affected RLINK is PAUSED with the `secondary_config_err` flag set. This prevents replication to the Secondary until the problem is corrected.

Run the *`vxrlink verify rlink`* command at either node to check whether this has occurred. When the configuration error has been corrected, the affected RLINK can be resumed.

Missing data volume error during modifcation of an RVG

If a data volume is added to the Primary RVG and the Secondary has no corresponding data volume, the RLINK state changes to PAUSED with the `secondary_config_err` flag set. Executing the `vxrlink verify` command produces the following:

On the Primary:

```
# vxrlink -g hrdg verify rlk_london_hr_rvg
RLINK              REMOTE HOST    LOCAL_HOST    STATUS    STATE
rlk_london_hr_rvg   london          seattle     ERROR     PAUSE
ERROR: hr_dv04 does not exist on secondary (london)
```

On the Secondary:

```
# vxrlink -g hrdg verify rlk_seattle_hr_rvg
RLINK                REMOTE HOST     LOCAL_HOST        STATUS
```

```
STATE
rlk_seattle_hr_rvg   seattle         london            ERROR
PAUSE
ERROR: hr_dv04 does not exist on secondary (local host)
```

To correct the problem, either create and associate `hr_dv04` on the Secondary or alternatively, dissociate `vol04` from the Primary, and then resume the Secondary RLINK. To resume the Secondary RLINK, use the *`vradmin resumerep rvg_name`* command.

If `hr_dv04` on the Primary contains valid data, copy its contents to `hr_dv04` on the Secondary before associating the volume to the Secondary RVG.

Data volume mismatch error during modification of an RVG

If a Primary data volume is increased in size, but the Secondary data volume is not, a configuration error results.

On the Primary:

```
# vxassist growby hr_dv04 100
# vxrlink -g hrdg verify rlk_london_hr_rvg
RLINK                REMOTE HOST    LOCAL_HOST       STATUS      STATE
rlk_london_hr_rvg    london         seattle          ERROR       PAUSE
ERROR: hr_dv04 too small (12800). Primary is 12900
```

On the Secondary:

```
# vxrlink -g hrdg verify rlk_seattle_hr_rvg
RLINK                REMOTE HOST    LOCAL_HOST       STATUS      STATE
rlk_seattle_hr_rvg   seattle        london           ERROR       PAUSE
ERROR: hr_dv04 too small (12800). Primary is 12900
```

To correct the problem, increase the size of the Secondary data volume, or shrink the Primary data volume:

```
# vradmin -g hrdg resizevol hr_rvg hr_dv04 12900
```

After resizing a data volume, resume the Secondary RLINK by issuing the following command on any host in the RDS:

```
# vradmin -g hrdg resumerep hr_rvg
```

Data volume name mismatch error during modification of an RVG

If a volume is renamed on the Primary but not on the Secondary, a configuration error results and the RLINK will be disconnected. Use the `vxprint -lP` command to view the RLINK flags. If the `secondary_config_err` flag is set, use one of the following commands to determine if there is a data volume name mismatch error.

On the Primary:

```
# vxrlink -g hrdg verify rlk_london_hr_rvg
RLINK                REMOTE HOST   LOCAL_HOST      STATUS      STATE
rlk_london_hr_rvg    london        seattle         ERROR       PAUSE
ERROR: hr_dv04 on secondary has wrong primary_datavol name (hr_dv04,
should be hr_dv05)
```

On the Secondary:

```
# vxrlink -g hrdg verify rlk_seattle_hr_rvg
RLINK                REMOTE HOST   LOCAL_HOST       STATUS      STATE
rlk_seattle_hr_rvg   seattle       london           ERROR       PAUSE
ERROR: hr_dv04 on secondary has wrong primary_datavol name (hr_dv04,
should be hr_dv05)
```

To fix this error, do one of the following:

- Rename either the Primary or Secondary data volume, and resume the RLINK using the *`vradmin resumerep rvg_name`* command.
  OR
- Set the `primary_datavol` field on the Secondary data volume to refer to the new name of the Primary data volume as follows, and resume the RLINK using the *`vradmin resumerep rvg_name`* command.
  On the Secondary:

  ```
  # vxedit -g hrdg set primary_datavol=hr_dv05 hr_dv04
  ```

  where `hr_dv05` is the new name on the Primary

Volume set configuration errors during modification of an RVG

If a volume set is associated to the RDS, the name of the volume set on the Secondary must have the same name as the volume set on the Primary in order for replication to occur. In addition, the volume set on the Secondary must have the same component volumes, with the same names, lengths and indices as on the Primary.

If a component volume is resized on the Primary but not on the Secondary, a data volume mismatch error results. Resize the volume and resume replication.

See "Data volume mismatch error during modification of an RVG" on page 364.

The configuration of the Secondary is checked for configuration errors, when an RLINK is attached for the first time. If any errors are found, the `vradmin startrep` command fails and prints error messages indicating the problem. Correct the configuration error, and then retry the command.

Configuration errors may also occur when you modify a volume set or its component volumes. Run the *`vxrlink verify rlink`* command at either node to check whether this has occurred. Correct the configuration error, and then resume the RLINK.

Volume set name mismatch error

If the volume set name differs on the Primary and the Secondary, the following error displays:

```
VxVM VVR vxrlink ERROR V-5-1-0 VSet name vset_name of secondary datavol
vol_name does not match VSet name vset_name of primary datavol
vol_name
```

To correct the problem, rename the volume set on either the Primary or the Secondary, using the following command:

```
# vxedit -g diskgroup rename vset_name new_vset_name
```

Volume index mismatch error

If the indices for the component volumes on the Primary volume set and the Secondary volume set are different, the following error displays:

```
VxVM VVR vxrlink ERROR V-5-1-0 VSet index (index_name) of secondary datavol
vol_name does not match VSet index (index_name) of primary datavol
vol_name
```

To correct the problem, perform the following steps on the Secondary:

1. Dissociate each volume from the volume set using the following command:

   ```
   # vxvset -g diskgroup rmvol vset_name compvol_name
   ```

   When you remove the last volume, the volume set is also removed.

2. Create the volume set using the following command:

   ```
   # vxvset -g diskgroup -o index make vset_name \
     compvol_name index
   ```

3. Associate each of the remaining volumes to the volume set, specifying the index of the corresponding volumes on the Primary using the following command:

   ```
   # vxvset -g diskgroup -o index addvol vset_name  \
     compvol_name index
   ```

Component volume mismatch error

If a data volume is removed from the volume set on the Primary RVG only or added to the volume set on the Secondary RVG only, the following error displays:

```
Secondary datavol vol_name is associated to VSet vol_name
whereas primary datavol is not associated to any Vset
```

Similarly, if a data volume is removed from the volume set on the Secondary RVG only or added to the volume set on the Primary RVG only, the following error displays:

```
Primary datavol vol_name is associated to VSet whereas secondary
datavol vol_name is not associated to any Vset
```

To correct the problem, add or remove data volumes from either the Secondary or the Primary volume sets. The volume sets on the Primary and the Secondary should have the same component volumes.

To add a data volume to a volume set, do one of the following:

- To add a data volume to a volume set in an RVG:

  ```
  # vradmin -tovset vset_name addvol rvg_name vol_name
  ```

- To remove a data volume in a volume set in an RVG:

  ```
  # vradmin -fromvset vset_name delvol rvg_name vol_name
  ```

Recovery on the Primary or Secondary

This section describes how to recover from various types of disasters, such as a Primary host crash or an error on the Primary or Secondary data volumes.

About recovery from a Primary-host crash

When a Primary host recovers from a failure, VVR automatically recovers the RVG configuration. When the Primary recovers, VVR recovers the Primary SRL and all volumes in the RVG. Information about the recent activity on the SRL and the data volume is maintained in the SRL header. VVR uses this information to speed up recovery, which is automatic on reboot.

Recovering from Primary data volume error

If a write to a Primary data volume fails, the data volume is detached. The RVG continues to function as before to provide access to other volumes in the RVG. Writes to the failed volume return an error and are not logged in the SRL.

RLINKs are not affected by a data volume failure. If the SRL was not empty at the time of the volume error, those updates will continue to flow from the SRL to the Secondary RLINKs. Any writes for the failed volume that were completed by the application but not written to the volume remain in the SRL. These writes are marked as pending in the SRL and are replayed to the volume when the volume is recovered. If the volume is recovered from the backup and restarted, these writes are discarded.

If the data volume had a permanent failure, such as damaged hardware, you must recover from backup. Recovery from this failure consists of two parts:

- Restoring the Primary data volume from backup
- Resynchronizing any Secondary RLINKs

If the RVG contains a database, recovery of the failed data volume must be coordinated with the recovery requirements of the database. The details of the database recovery sequence determine what must be done to synchronize Secondary RLINKs.

Detailed examples of recovery procedures are given in the examples:

- See “Example - Recovery with detached RLINKs” on page 369.
- See “Example - Recovery with minimal repair” on page 369.
- See “Example - Recovery by migrating the primary” on page 370.

If the data volume failed due to a temporary outage such as a disconnected cable, and you are sure that there is no permanent hardware damage, you can start the

data volume without dissociating it from the RVG. The pending writes in the SRL are replayed to the data volume.

See "Example - Recovery from temporary I/O error" on page 370.

Example - Recovery with detached RLINKs

In this example, all the RLINKs are detached before recovery of the failure begins on the Primary. When recovery of the failure is complete, including any database recovery procedures, all the RLINKs must be synchronized using a Primary Storage Checkpoint.

Perform the steps on the Primary. In this example, the Primary host is `seattle`.

To recover from failure

1. Detach all RLINKs

```
# vxrlink -g hrdg det rlk_london_hr_rvg
```

2. Fix or repair the data volume.

   If the data volume can be repaired by repairing its underlying subdisks, you need not dissociate the data volume from the RVG. If the problem is fixed by dissociating the failed volume and associating a new one in its place, the dissociation and association must be done while the RVG is stopped.

3. Make sure the data volume is started before restarting the RVG.

```
# vxvol -g hrdg start hr_dv01
# vxrvg -g hrdg start hr_rvg
```

4. Restore the database.
5. Synchronize all the RLINKs using block-level backup and checkpointing.

Example - Recovery with minimal repair

This example does the minimum to repair data volume errors, leaving all RLINKs attached. In this example, restoring the failed volume data from backup, and the database recovery is done with live RLINKs. Because all the changes on the Primary are replicated, all the Secondaries must be consistent with the Primary after the changes have been replicated. This method may not always be practical because it might require replication of large amounts of data. The repaired data volume must also be carefully tested on every target database to be supported.

Perform the steps on the Primary. In this example, the Primary host is `seattle`.

To recover from failure

1. Stop the RVG.

```
# vxrvg -g hrdg stop hr_rvg
```

2. Dissociate the failed data volume from the RVG.
3. Fix or repair the data volume or use a new volume.

   If the data volume can be repaired by repairing its underlying subdisks, you need not dissociate the data volume from the RVG. If the problem is fixed by dissociating the failed volume and associating a new one in its place, the dissociation and association must be done while the RVG is stopped.
4. Associate the volume with the RVG.
5. Make sure the data volume is started before restarting the RVG. If the data volume is not started, start the data volume:

```
# vxvol -g hrdg start hr_dv01
```

6. Start the RVG:

```
# vxrvg -g hrdg start hr_rvg
```

7. Restore the database.

Example - Recovery by migrating the primary

As an alternative recovery method, the Primary role can be transferred to a Secondary host.

See "Migrating the Primary" on page 300.

After takeover, the original Primary with the failed data volume will not become `acting_secondary` until the failed data volume is recovered or dissociated.

Example - Recovery from temporary I/O error

If the I/O error on the data volume is temporary and you are sure that all the existing data is intact, you can start the data volume without dissociating it from the RVG. For example, if the SCSI cable was disconnected or there was a power outage of the storage. In this case, follow the steps below.

To recover from a temporary I/O error

1. Fix the temporary failure.
2. Start the data volume:

```
# vxvol -g hrdg start hr_dv01
```

   Any outstanding writes in the SRL are written to the data volume.

Primary SRL volume error cleanup and restart

If there is an error accessing the Primary SRL, the SRL is dissociated and the RLINKs are detached. The state of the Primary and Secondary RLINKs is changed to STALE. The RVG state does not change, but the RVG is put into PASSTHRU mode that allows update of the Primary volume to continue until the error is fixed.

See "About RVG PASSTHRU mode" on page 372.

The SRL must be repaired manually and then associated with the RVG. While the SRL is being repaired, no attempt is made to send data to the RLINKs. After the SRL is replaced, all RLINKs must be completely synchronized. Attach the RLINKs and perform a complete synchronization of the Secondaries.

On the Primary (seattle):

To cleanup after a Primary SRL error

1. Dissociate the SRL from the RVG.

```
# vxvol -g hrdg dis hr_srl
```

2. Fix or replace the SRL volume.
3. Make sure that the repaired SRL is started before associating it with the RVG. If the repaired SRL is not started, start it:

```
# vxvol -g hrdg start hr_srl
```

4. Associate a new SRL with the RVG. After associating the new SRL, the RVG PASSTHRU mode no longer displays in the output of the command `vxprint -lV`.

```
# vxvol -g hrdg aslog hr_rvg hr_srl
```

5. Perform a complete synchronization of the Secondary.

   See "Synchronizing the Secondary and starting replication" on page 152.

About RVG PASSTHRU mode

Typically, writes to data volumes associated with an RVG go to the RVG's SRL first, and then to the RLINKs and data volumes. If the Primary SRL is ever detached because of an access error, then the Primary RVG is put into PASSTHRU mode. In PASSTHRU mode, writes to the data volume are passed directly to the underlying data volume, bypassing the SRL. No RLINKs receive the writes. Use `vxprint -l` on the RVG to see if the `passthru` flag is set. Associating a new SRL will clear PASSTHRU mode, and the Secondary node RVGs must be synchronized.

Primary SRL volume error at reboot

If the Primary SRL has an error during reboot, there is a possibility that the disks or arrays containing the SRL have not yet come online. Because of this, instead of placing the RVG in PASSTHRU mode, VVR does not recover the RVG. When the SRL becomes available, issue the following commands to recover the RVG and the RLINK:

```
# vxrvg -g diskgroup recover rvg_name
# vxrlink -g diskgroup recover rlink_name
```

After this error has occurred and you have successfully recovered the RVG, if you dissociate a volume from the RVG, you may see the following message:

```
VxVM vxvol WARNING V-5-1-5273 Because there could be outstanding writes
in the SRL, the data volume being dissociated should be considered
out-of-date and inconsistent
```

You can ignore this message.

If the SRL is permanently lost, create a new SRL.

See "Recovering from SRL header error" on page 373.

In this case, it is possible that writes that had succeeded on the old SRL and acknowledged to the application, were not yet flushed to the data volumes and are now lost. Consequently, you must restore the data volumes from backup before proceeding. Because this causes the data volumes to be completely rewritten, it is recommended that you detach the RLINKs and synchronize them after the restore operation is complete.

Primary SRL volume overflow recovery

Because the size of the Primary SRL is finite, prolonged halts in replication activity to any RLINK can exceed the log's ability to maintain all the necessary update history to bring an RLINK up-to-date. When this occurs, the RLINK in question is

marked as STALE and requires manual recovery before replication can proceed. A STALE RLINK can only be brought up-to-date by using automatic synchronization or a block-level backup and Storage Checkpoint. The other RLINKs, the RVG, and the SRL volume are all still operational.

SRL overflow protection can be set up to prevent SRL overflow, and is the default. Instead of allowing the RLINK to become STALE, dcm logging is initiated. At a later time when the communication link is not overloaded, you can incrementally resynchronize the RLINK using the `vradmin resync rvg` command.

Primary SRL header error cleanup and recovery

An SRL header failure on the Primary is a serious error. All RLINKs are lost and must be recovered using a Primary Storage Checkpoint. Because information about data volume errors is kept in the SRL header, the correct status of data volumes cannot be guaranteed under all occurrences of this error. For this reason, we recommend that you mirror the SRL.

If an SRL header error occurs during normal operation and you notice it before a reboot occurs, you can be certain that any data volumes that have also (simultaneously) failed will have a status of DETACHED. If the system is rebooted before the `vxprint` command shows the volumes to be in the DETACHED state, the status of any failed data volumes may be lost. Both these cases involve multiple errors and are unlikely, but it is important to understand that the state of Primary data volumes can be suspect with this type of error.

When a Primary SRL header error occurs, writes to the RVG continue; however, all RLINKs are put in the STALE state. The RVG is operating in PASSTHRU mode.

Recovering from SRL header error

Recovering from an SRL header error requires dissociating the SRL from the RVG, repairing the SRL, and completely synchronizing all the RLINKs.

To recover from an SRL header error

1. Stop the RVG.

```
# vxrvg -g hrdg stop hr_rvg
```

2. Dissociate the SRL from the RVG.

```
# vxvol -g hrdg dis hr_srl
```

3. Repair or restore the SRL. Even if the problem can be fixed by repairing the underlying subdisks, the SRL must still be dissociated and reassociated to initialize the SRL header.

4 Make sure the SRL is started, and then reassociate the SRL:

```
# vxvol -g hrdg start hr_srl
# vxvol -g hrdg aslog hr_rvg hr_srl
```

5 Start the RVG:

```
# vxrvg -g hrdg start hr_rvg
```

6 Restore the data volumes from backup if needed. Synchronize all the RLINKs.

See "Methods to synchronize the Secondary" on page 152.

Secondary data volume error cleanup and recovery

If an I/O error occurs during access of a Secondary data volume, the data volume is automatically detached from the RVG and the RLINKs are disconnected. A subsequent attempt by the Primary to connect to the Secondary fails and a message that the Secondary volumes are stopped is displayed. The Primary is unaffected and writes continue to be logged into the SRL. After the Secondary data volume error is fixed and the data volume is started, the RLINKs automatically reconnect.

If there is no suitable Primary or Secondary Storage Checkpoint, detach the RLINKs on both the Primary and Secondary, and then synchronize the RLINKs.

See "Restoring the Secondary from online backup" on page 280.

Recovery using a Secondary Storage Checkpoint

This section explains how to recover from a Secondary data volume error using a Secondary Storage Checkpoint.

On the Secondary (london):

1 Repair the failed data volume. You need not dissociate the data volume if the problem can be fixed by repairing the underlying subdisks.

2 Make sure that the data volume is started:

```
# vxvol -g hrdg start hr_dv01
```

3 Restore data from the Secondary Storage Checkpoint backup to all the volumes. If all volumes are restored from backup, the Secondary will remain consistent during the synchronization. Restore the RLINK by issuing the following command:

```
#  vxrlink -g hrdg -c sec_chkpt restore rlk_seattle_hr_rvg
```

Cleanup using a Primary Storage Checkpoint

On the Secondary (`london`):

1. Repair the failed data volume as above. Be sure that the data volume is started before proceeding:

   ```
   # vxvol -g hrdg start hr_dv01
   ```

2. Detach the RLINK to enable writing to the Secondary data volumes:

   ```
   # vxrlink -g hrdg det rlk_seattle_hr_rvg
   ```

3. Restore data from the Primary Storage Checkpoint backup to all data volumes. Unlike restoration from a Secondary Storage Checkpoint, the Primary Storage Checkpoint data must be loaded onto all Secondary data volumes, not just the failed volume. If a usable Primary Storage Checkpoint does not already exist, make a new Storage Checkpoint.

   See "Example—Synchronizing the Secondary using block-level backup" on page 506.

4. Reattach the RLINK.

   ```
   # vxrlink -g hrdg att rlk_seattle_hr_rvg
   ```

On the Primary (`seattle`):

Detach the RLINK and then reattach it from the Primary Storage Checkpoint using the following commands:

```
# vxrlink -g hrdg det rlk_london_hr_rvg
# vxrlink -g hrdg -c primary_checkpoint att rlk_london_hr_rvg
```

Secondary SRL volume error cleanup and recovery

The Secondary SRL is used only during atomic recovery of an RLINK and when an IBC is active. If I/O errors occur during recovery of the Secondary SRL, the recovery fails, the SRL volume is automatically detached, and the RLINK is forced to the pause state. Manual intervention is required to repair the physical problem, reattach the SRL, and resume the RLINK. Upon resumption, an automatic recovery of the RVG is retried and if it succeeds, update activity can continue. The only problem occurs if the Primary SRL overflows before the repair is complete, in which case a full synchronization is required.

If an error occurs in the data portion of the SRL, the RLINK is forced to the PAUSE state with the `secondary_paused` flag set. The SRL is not dissociated.

If an error occurs in the SRL header, the Secondary RVG is forced to the FAIL state and the SRL is dissociated.

On the Secondary (`london`):

1. Dissociate the SRL, fix it, and then re-associate it. The dissociation and re-association are necessary even if the problem can be fixed by repairing the underlying subdisks because this sequence initializes the SRL header.

   ```
   # vxvol -g hrdg dis hr_srl
   ```

   Fix or replace the SRL. Be sure the SRL is started before associating it:

   ```
   # vxvol -g hrdg start hr_srl
   # vxvol -g hrdg aslog hr_rvg hr_srl
   ```

2. Run the RLINK resume operation to clear the `secondary_log_err` flag.

   ```
   # vxrlink -g hrdg resume rlk_seattle_hr_rvg
   ```

Secondary SRL header error cleanup and recovery

An SRL header failure on the Secondary puts the Secondary RVG into the fail state, and sets the RLINK state to the PAUSE state on both the Primary and Secondary. Because information about data volume errors is kept in the SRL header, the correct state of data volumes is not guaranteed in all cases. If a Secondary SRL header failure occurs during normal operation and is noticed before a reboot occurs, any data volumes that also failed will have a state of DETACHED. If the system is rebooted before the `vxprint` command shows the volumes to be in the DETACHED state, the status of any failed data volumes may be lost. Both these cases involve multiple errors and are unlikely, but it is important to understand that the state of Secondary data volumes can be suspect with this type of error.

To cleanup and recover the SRL header failure

1. Dissociate the SRL volume.

   ```
   # vxvol -g hrdg dis hr_srl
   ```

2. Repair the SRL volume. Even if the problem can be fixed by repairing the underlying subdisks, the SRL volume must still be dissociated and re-associated to initialize the SRL header.

3. Start the SRL volume. Then, re-associate it.

   ```
   # vxvol -g hrdg start hr_srl
   # vxvol -g hrdg aslog hr_rvg hr_srl
   ```

4 Start the RVG.

```
# vxrvg -g hrdg start hr_rvg
```

5 If the integrity of the data volumes is not suspect, just resume the RLINK.

```
# vxrlink -g hrdg resume rlk_seattle_hr_rvg
```

OR

If the integrity of the data volumes is suspect, and a Secondary Storage Checkpoint backup is available, restore from the Secondary Storage Checkpoint.

```
# vxrlink -g hrdg det rlk_seattle_hr_rvg
# vxrlink -g hrdg -f att rlk_seattle_hr_rvg
# vxrlink -g hrdg -w pause rlk_seattle_hr_rvg
```

Restore the Secondary Storage Checkpoint backup data on to the data volumes.

```
# vxrlink -g hrdg -c secondary_checkpoint restore \
  rlk_seattle_hr_rvg
```

OR

If the integrity of the data volumes is suspect and no Secondary Storage Checkpoint is available, synchronize the Secondary using a block-level backup and Primary Storage Checkpoint.

See "Example—Synchronizing the Secondary using block-level backup" on page 506.

As an alternative, you can also use automatic synchronization.

```
# vxrlink -g hrdg det rlk_seattle_hr_rvg
```

On the Secondary, restore the Primary Storage Checkpoint backup data to the data volumes.

```
# vxrlink -g hrdg -f att rlk_seattle_hr_rvg
```

On the Primary (`seattle`):

```
# vxrlink -g hrdg -c primary_checkpoint att \
  rlk_london_hr_rvg
```

Secondary SRL header error at reboot

If the secondary SRL has an error after a reboot, it is not possible to recover it, even if the SRL subsequently becomes available. Ignore the following message:

```
VxVM VVR vxrvg ERROR V-5-1-5268 RVG rvg_name cannot be recovered
because SRL is not accessible. Try recovering the RVG after the
SRL becomes available using vxrecover -s command
```

To reset the SRL volume

1. Dissociate the SRL:

   ```
   # vxvol -g hrdg -f dis srl
   ```

   Ignore the following messages:

   ```
   VxVM vxvol WARNING V-5-1-5291 WARNING: Rvg rvgname has not been
   recovered because the SRL is not available. The data volumes may
   be out-of-date and inconsistent
   VxVM vxvol WARNING V-5-1-5296 The data volumes in the rvg rvgname
   cannot be recovered because the SRL is being dissociated.
   Restore the data volumes from backup before starting the applications
   ```

2. Create a new SRL volume, *new_srl* and continue as follows:

   ```
   # vxvol -g hrdg aslog rvg_name new_srl
   # vxrlink -g hrdg recover rlink_name
   # vxrlink -g hrdg -f att rlink_name
   # vxrvg -g hrdg start rvg_name
   ```

   If replication was frozen due to receipt of an IBC, the data in the SRL is lost but there is no indication of this problem. To see whether this was the case, examine the /var/log/messages file for a message such as:

   ```
   WARNING: VxVM VVR vxio V-5-0-259 Replication frozen for rlink
   <rlink>
   ```

   If this is the last message for the RLINK, that is, if there is no subsequent message stating that replication was unfrozen, the Primary RLINK must be completely resynchronized.

Chapter 13

Tuning replication performance

This chapter includes the following topics:

- Overview of replication tuning
- SRL layout
- Tuning Volume Replicator

Overview of replication tuning

The important factors that affect VVR performance are the layout of the SRL and the sizing of the VVR buffers. This chapter explains how to decide on the lay out of the SRL, and size the VVR buffers. It also describes how to choose the value of other VVR tunables.

SRL layout

This section explains how the SRL affects application performance and how a good SRL layout can improve performance.

How SRL affects performance

Incoming writes to a data volume on the Primary are written to the SRL first, and then to the data volume. VVR manages writes in the same way, irrespective of the replication settings, including the mode of replication. Note that writes to different data volumes within the RVG are all written in the same SRL. Therefore, the SRL throughput may affect performance. The use of the SRL may not degrade performance too badly, for the following reasons:

- The writes to SRL are sequential, whereas, the writes to the data volumes are spatially random in most cases. Typically, sequential writes are processed faster than the random writes.
- The SRL is not used to process read operations performed by the application. If a large percentage of the operations are read operations, then the SRL is not busy at these times.

If the rate at which the application writes to the data volumes is greater than the rate at which the SRL can process writes, then the application could become slow. The following sections explain how to lay out the SRL to improve performance.

Striping the SRL

Striping the SRL over several physical disks to increase the available bandwidth can improve performance.

Choosing disks for the SRL

It is recommended that there be no overlap between the physical disks comprising the SRL and those comprising the data volumes, because all write requests to VVR result in a write to both the SRL and the requested data volume. Any such overlap is guaranteed to lead to major performance problems, as the disk head thrashes between the SRL and data sections of the disk. Slowdowns of over 100% can be expected.

Mirroring the SRL

It is recommended that the SRL be mirrored to improve its reliability. The loss of the SRL immediately stops replication. The only way to recover from this is to perform a full resynchronization, which is a time-consuming procedure to be avoided whenever possible. Under certain circumstances, the loss of the SRL may even cause loss of the data volumes. The risk of this failure can be minimized by mirroring the SRL.

Tuning Volume Replicator

This section describes how to adjust the tunable parameters that control the system resources used by VVR. Depending on the system resources that are available, adjustments may be required to the values of some tunable parameters to optimize performance.

Note that in a shared disk group environment, each of the VVR buffer spaces must be set to the same value on each node.

For instructions on changing the value of tunables, refer to the *Veritas InfoScale™ Replication Administrator's Guide*.

VVR buffer space

VVR uses the following buffers to replicate data:

- Write buffer space on the Primary
- Readback buffer space on the Primary
- Buffer space on the Secondary

Write buffer space on the Primary

VVR processes writes differently depending on whether it is replicating in a private disk group or a shared disk group. Also, in a shared disk group environment, VVR processes writes differently when replicating in synchronous and asynchronous mode.

When a write is issued, a write buffer is allocated from the write buffer space on the Primary. In a private disk group, the buffer is not released until the data has been written to the Primary SRL and sent to all the Secondaries in synchronous mode. If the Secondaries in asynchronous mode cannot keep up with the application write rate, the data to be sent to the Secondary starts accumulating in the write-buffer space on the Primary. As a result, write-buffer space on the Primary becomes low. Then, VVR begins to free some buffers and postpones sending the data to the Secondaries in asynchronous mode. As a result, more space is freed up for incoming write requests so that they are not delayed.

If the disk group is shared and the write is issued on the logowner, VVR allocates a write buffer from the write buffer space on the logowner.

If the disk group is shared and VVR is replicating in synchronous mode, and the write is issued on the non-logowner, VVR sends the write to the logowner. On the logowner, VVR receives the write in the write ship buffer space and then copies it to the write buffer space. This process is called write shipping. In a shared disk group that uses write shipping, the write buffer is freed in the same way as for a private disk group.

If the disk group is shared and VVR is replicating in asynchronous mode, and the write is issued on the non-logowner, VVR exchanges metadata information about the write with the logowner. After VVR receives the metadata information on the non-logowner, VVR performs the writes locally on the non-logowner. This process is called metadata shipping.

Readback buffer space on the Primary

When VVR is ready to send the freed requests to the Secondary, the freed requests are read back from the SRL. The data from the SRL is read back in to the pre-rvg readback buffer space on the Primary.

The need to read back data from the SRL has an impact on write latency because more non-sequential I/O is performed on the SRL.

See the section "Choosing the mode of replication" in the *Veritas InfoScale™ Replication Administrator's Guide*.

Reading back data from the SRL also increases the load on the system and slows the rate at which data is sent to the Secondaries.

Note that the write buffer is freed only if the mode of replication is asynchronous; the writes do not have to be read back from the SRL when replicating in synchronous mode.

Buffer space on the Secondary

The writes from the Primary are received in to the buffer space on the Secondary. The write is then written to the Secondary data volume from this buffer space. A write on the Primary can complete before the write to the Secondary data volume completes, even in synchronous mode of replication. However, if the Secondary is low on buffer space, it rejects new writes from the Primary thereby slowing down the Primary. On the Primary this appears as an inability to send requests over the network. The results are identical to those pertaining to insufficient network bandwidth.

For Secondaries in asynchronous mode, there may be no limit to how far Secondary data volumes can fall behind unless certain mechanisms are in force.

See the section "Choosing latency and SRL protection" in the *Veritas InfoScale™ Replication Administrator's Guide*.

Hence, if all the Secondaries are replicating in asynchronous mode, the application may not slow down; if there are Secondaries in synchronous mode, the write rate of the application reduces.

Tunable parameters for the VVR buffer spaces

The amount of buffer space available to VVR affects the application and replication performance. You can use the following tunables to manage buffer space according to your requirements:

- `vol_rvio_maxpool_sz`
- `vol_min_lowmem_sz`

- `vol_max_rdback_sz`
- `vol_max_nmpool_sz`

The amount of buffer space available to VVR affects the application and replication performance. You can use the following tunables to manage buffer space according to your requirements:

- `vol_rvio_maxpool_sz`
- `vol_min_lowmem_sz`
- `vol_max_wrspool_sz`
- `vol_max_rdback_sz`
- `vol_max_nmpool_sz`

Use the `vxmemstat` command to monitor the buffer space used by VVR. The following sections describe each of the above tunables.

For instructions on changing the value of tunables, refer to the *Veritas InfoScale™ Replication Administrator's Guide.*

Tunable parameters for the write buffer space on the Primary in a private disk group

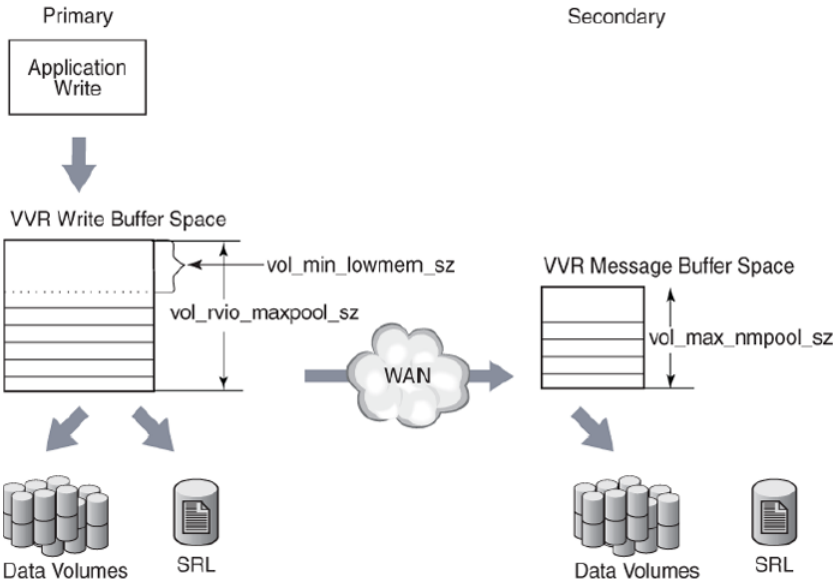
The following tunable parameters control the write buffer space on the Primary in a private disk group:

- vol_rvio_maxpool_sz
- vol_min_lowmem_sz

The amount of buffer space that can be allocated within the operating system to handle incoming writes is defined by the tunable vol_rvio_maxpool_sz, which defaults to 128MB.

Figure 13-1 shows the buffering process during a write operation.

Figure 13-1 How VVR uses buffers between the Primary and Secondary



If the available buffer space is not sufficient to process the write request, writes are held up. VVR must wait for current writes to complete and release the memory being used before processing new writes.

Furthermore, when the buffer space is low, VVR frees buffers early, requiring VVR to read back the write from the SRL.

See "Write buffer space on the Primary" on page 381.

Both these problems can be alleviated by increasing `vol_rvio_maxpool_sz`. By setting the `vol_rvio_maxpool_sz` to be large enough to hold the incoming writes, you can increase the number of concurrent writes and reduce the number of readbacks from the SRL. When decreasing the value of the `vol_rvio_maxpool_sz` tunable, stop all the RVGs on the system on which you are performing this operation.

When deciding whether or not a given write is freed early and read back later, VVR looks at the amount of buffer space available, and frees the write if the amount is below a threshold defined by the tunable `vol_min_lowmem_sz`. If this threshold is too low, it results in buffers being held for a long time. New writes cannot be performed because of lack of buffer space.

The `vol_min_lowmem_sz` tunable is 4MB (4194304 bytes).

You can raise the threshold by increasing the value of the tunable `vol_min_lowmem_sz`. It should be set to at least 3 x N x I, but not less than 520K, where N is the number of concurrent writes to replicated volumes, and I is the

average I/O size, rounded up to 8 kilobytes. The `vol_min_lowmem_sz` tunable is auto-tunable and depending on the incoming writes, VVR will increase or decrease the tunable value. The value that you specify for the tunable, using the `vxtune` utility or the system-specific interface, will be used as an initial value and could change depending on the application write behavior.

Note that this tunable is used only when replicating in asynchronous mode because SRL is not read back when replicating in synchronous mode.

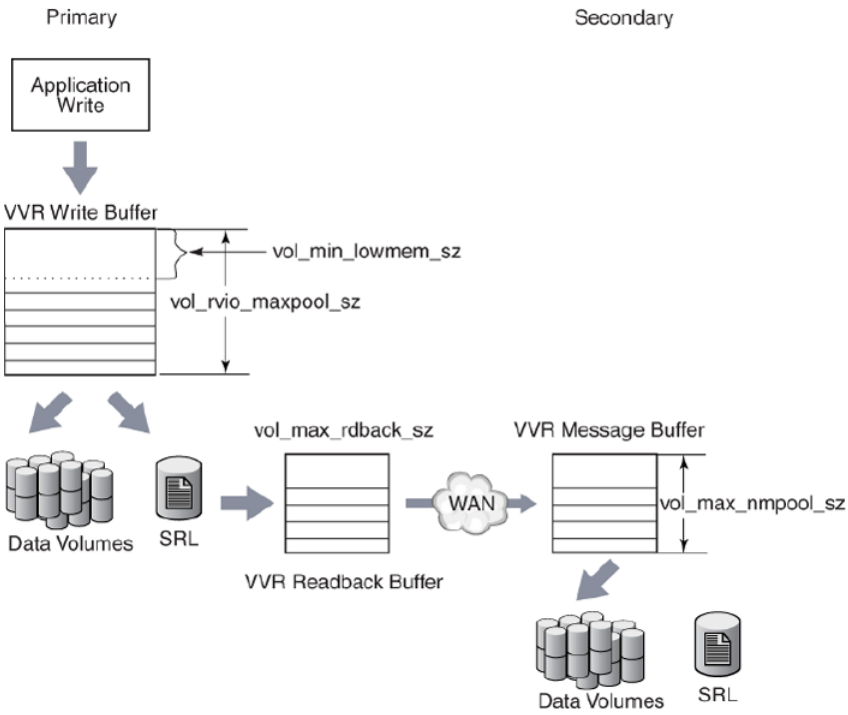
Use the `vxrvg stats` command to determine the maximum concurrency (N) and average write size (I).

Tunable parameter for the readback buffer space

The amount of buffer space available for readbacks is defined by the tunable, `vol_max_rdback_sz`, which defaults to 128 megabytes. To accommodate reading back more data, increase the value of `vol_max_rdback_sz`. You may need to increase this value if you have multiple Secondaries in asynchronous mode for one or more RVGs. Changing the value `vol_max_rdback_sz` will change the readback pool size of all the RVG's configured on the host.

Figure 13-2 shows how the `vol_max_rdback_sz` tunable is involved when VVR reads back data.

Figure 13-2 How VVR uses buffers during a readback



Use the `vxmemstat` command to monitor the buffer space. If the output indicates that the amount of space available is completely used, increase the value of the `vol_max_rdback_sz` tunable to improve readback performance. When decreasing the value of the `vol_max_rdback_sz` tunable, pause replication to all the Secondaries to which VVR is currently replicating.

Tunable parameters for the buffer space on the Primary in a shared disk group

In a shared disk group environment, the following tunable parameters control the buffer space on the Primary when replicating in asynchronous mode:

- `vol_rvio_maxpool_sz`
- `vol_min_lowmem_sz`
- `vol_max_rdback_sz`

In asynchronous mode, the tunable parameters work the same way as for a private disk group.

See "Tunable parameters for the write buffer space on the Primary in a private disk group" on page 383.

The `vol_rvio_maxpool_sz` tunable applies to all nodes. The `vol_min_lowmem_sz` `vol_max_rdback_sz` tunables are only applied on the logowner node. However, these tunables should also be set to the same values on all nodes, because any node may become the logowner at some time.

In a shared disk group environment, the following tunable parameters control the buffer space on the Primary when replicating in synchronous mode:

- `vol_max_wrspool_sz`
- `vol_rvio_maxpool_sz`

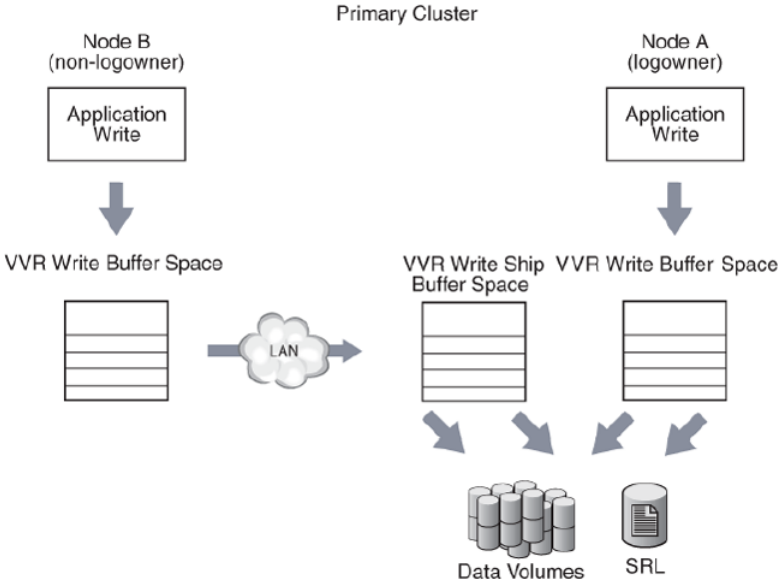
When replicating in synchronous mode, the `vol_rvio_maxpool_sz` tunable works as the same way as for a private disk group, except that it won't prevent readbacks.

See "Tunable parameters for the write buffer space on the Primary in a private disk group" on page 383.

This tunable should be set on all nodes in the shared disk group. In addition, the amount of buffer space that can be allocated on the logowner to receive writes sent by the non-logowner is defined by the write ship buffer space tunable `vol_max_wrspool_sz`, which defaults to 64MB. This tunable should be set to the same value on all nodes, because any node may become the logowner at some time.

Figure 13-3 shows how VVR uses buffers on a Primary in a shared disk group environment.

Figure 13-3 How VVR uses buffers on a Primary in a shared disk group for synchronous RLINKs



Tunable parameters for the buffer space on the Secondary

The amount of buffer space available for requests coming in to the Secondary over the network is determined by the VVR tunable, `vol_max_nmpool_sz`, which defaults to 64 megabytes. VVR allocates separate buffer space for each Secondary RVG, the size of which is equal to the value of the tunable `vol_max_nmpool_sz`. The buffer space on the Secondary must be large enough to prevent slowing the network transfers excessively.

If the buffer is too large, it can cause problems. When a write arrives at the Secondary, the Secondary sends an acknowledgment to the Primary so that the Primary knows the transfer is complete. When the write is written to the data volume on the Secondary, the Secondary sends another acknowledgment, which tells the Primary that the write can be discarded from the SRL. However, if this second acknowledgment is not sent within one minute, the Primary disconnects the RLINK. The RLINK reconnects immediately but this causes disruption of the network flow and potentially other problems. Thus, the buffer space on the Secondary should be sized in such a way that no write can remain in it for one minute. This size depends on the rate at which the data can be written to the disks, which is dependent on the disks themselves, the I/O buses, the load on the system, and the nature of the writes (random or sequential, small or large).

If the write rate is W megabytes/second, the size of the buffer should be no greater than W * 50 megabytes, that is, 50 seconds' worth of writes.

There are various ways to measure W. If the disks and volume layouts on the Secondary are comparable to those on the Primary and you have I/O statistics from the Primary before replication was implemented, these statistics can serve to arrive at the maximum write rate.

Alternatively, if replication has already been implemented, start by sizing the buffer space on the Secondary to be large enough to avoid timeout and memory errors.

While replication is active at the peak rate, run the following command and make sure there are no memory errors and the number of timeout errors is small:

```
# vxrlink -g diskgroup -i5 stats rlink_name
```

Then, run the `vxstat` command to get the lowest write rate:

```
# vxstat -g diskgroup -i5
```

The output looks similar to this:

```
                    OPERATIONS        BLOCKS        AVG TIME(ms)
TYP NAME           READ     WRITE      READ  WRITE   READ  WRITE

Mon 29 Sep 2003 07:33:07 AM PDT
vol srl1             0      1245         0   1663   0.0    9.0
vol archive          0       750         0    750   0.0    9.0
vol archive-L01      0       384         0    384   0.0    5.9
vol archive-L02      0       366         0    366   0.0   12.1
vol ora02            0       450         0    900   0.0   11.1
vol ora03            0         0         0      0   0.0    0.0
vol ora04            0         0         0      0   0.0    0.0

Mon 29 Sep 2003 07:33:12 AM PDT
vol srl1             0       991         0   1389   0.0   20.1
vol archive          0       495         0    495   0.0   10.1
vol archive-L01      0       256         0    256   0.0    5.9
vol archive-L02      0       239         0    239   0.0   14.4
vol ora02            0       494         0    988   0.0   10.0
vol ora03            0         0         0      0   0.0    0.0
vol ora04            0         0         0      0   0.0    0.0
```

For each interval, add the numbers in the blocks written column for data volumes, but do not include the SRL. Also, do not include any subvolumes. For example, `archive-L01`, and `archive-L02` are subvolumes of the volume `archive`. The statistics of the writes to the subvolumes are included in the statistics for the volume

archive. You may vary the interval, the total time you run the test, and the number of times you run the test according to your needs. In this example, the interval is 5 seconds and the count is in blocks, hence on a machine with 2 kilobytes of block size, the number of megabytes per interval, M, is (total * 2048)/(1024*1024), where total is the sum for one interval. Hence, for one second the number of megabytes is M/5 and the size of the buffer is (M/5)*50. If there is more than one Primary, do not increase the buffer size beyond this number.

The writes to the SRL should not be considered part of the I/O load of the application. However, in asynchronous mode, the Secondary writes the incoming updates to both the Secondary SRL and the data volumes, so it may be necessary to make the value of vol_max_nmpool_sz slightly larger. However, to avoid the problems discussed at the beginning of this section, the calculated vol_max_nmpool_sz value should still ensure that writes do not remain in the pool for more than one minute.

DCM replay block size

When the Data Change Map (DCM) is being replayed, data is sent to the Secondary in blocks. The tunable vol_dcm_replay_size indicates the size of the DCM replay blocks. The default value of vol_dcm_replay_size is 256K. This tunable cannot be changed using the vxtune command. You can use the operating system specific method to change the value.

Heartbeat timeout

VVR uses a heartbeat mechanism to detect communication failures between the Primary and the Secondary hosts. The RLINKs connect after the heartbeats are exchanged between the Primary and the Secondary. The RLINK remains connected while the heartbeats continue to be acknowledged by the remote host. The maximum interval during which heartbeats can remain unacknowledged is known as the heartbeat timeout value. If the heartbeat is not acknowledged within the specified timeout value, VVR disconnects the RLINK.

The tunable vol_nm_hb_timeout enables you to specify the heartbeat timeout value. The default is 10 seconds. For a high latency network, increasing the default value of the tunable vol_nm_hb_timeout prevents the RLINKs from experiencing false disconnects.

In Release 5.1SP1 and later, you can tune the heartbeat timeout value with vxtune.

Memory chunk size

The tunable voliomem_chunk_size enables you to configure the granularity of memory chunks used by VVR when allocating or releasing system memory. A

memory chunk is a contiguous piece of memory that can be written to the disk in one operation. If the write size of the application is larger than the memory chunk size then the write is split resulting in multiple operations, which can reduce performance.

The default memory chunk size is 32K. For applications performing large writes, increase the size of `voliomem_chunk_size` to improve replication performance. The maximum value of `voliomem_chunk_size` is 32K.

UDP replication tuning

When you use UDP as the replication protocol, VVR uses its own network flowcontrol. VVR increases or decreases the rate at which data is sent depending on the number of timeouts or memory errors it gets per second. If the number of errors is greater, VVR decreases the sending rate to avoid network congestion. If there are only a few (or no) errors, VVR continues to increase the sending rate by a fixed amount every second. For a lossy network, a large number of errors may occur, which prevents VVR from increasing the sending rate. However, these errors are not due to network congestion so VVR should continue to increase the sending rate.

You specify the error tolerance VVR uses by setting two tunables: `vol_rp_increment` and `vol_rp_decrement`. VVR increases its sending rate if timeouts or memory errors per second are not more than the `vol_rp_increment` value. VVR decreases its sending rate if timeouts or memory errors per second are more than `vol_rp_decrement`. The default value of both tunables is `8`.

In the case of a lossy network, the sending rate does not increase because the number of errors per second could be more than `vol_rp_increment` or `vol_rp_decrement`, and the sending rate can decrease further. This impacts replication performance. If RLINK statistics show a higher number of errors and VVR is not using available bandwidth, you may be able to improve replication performance by tuning `vol_rp_increment` and `vol_rp_decrement` to higher value like 16 or 32. In Release 5.1SP1 and later, you can change both tunables using the `vxtune` command. This tuning is required only on the Primary host.

Tuning the number of TCP connections

If you use TCP as the replication protocol, VVR supports multi-connection. The number of connections is auto-tuned depending on latency between the Primary host and the Secondary host. If the network pipe is very thick, the auto-tuned number of connections may not be enough to saturate the pipe.

You can specify the number of TCP connections using the tunable `nmcom_max_connections`.

You may need to tune nmcom_max_connections in the following situations:

- VVR is not using available bandwidth
- The latency between the Primary host and Secondary host is high (greater than 20 ms)

To tune the number of TCP connections

1. Determine the number of active connections. Enter:

```
# /etc/vx/diag.d/vxkprint |grep active_connections
```

2. If active_connections value is less than 8, tune nmcom_max_connections. Continue with the following steps.
3. Pause replication.
4. Tune the nmcom_max_connections value to 8. In Release 5.1SP1 and later, you can tune this value with the vxtune command. Enter:

```
# vxtune nmcom_max_connections 8
```

Note: This tuning is required only on the Primary host.

5. Resume replication to make the new value effective.
6. Use the vrstat command to observe the bandwidth use. If use increases and there is still room grow, tune nmcom_max_connections to 16 or 32 connections.

Message slots on the Secondary

VVR uses a number of memory slots on the Secondary site to store incoming network I/O. By default, number of messages slots is 1024. If the network has high packet reordering, the default number of message slots may not be enough. Packet will get dropped if they are out of order, and this will impact replication performance.

You can use network health check tools like Iperf to find packet reordering, or you can also use extended RLINK statistics. Enter the following command:

```
# vxrlink -g dgname -i10 -e stats rlink_name
Errors :
 ------
 No memory available                              : 0
 No message slots available                       : 0
 No memory available in nmcom pool on Secondary   : 0
 Timeout                                          : 0
```

```
Missing packet                                  : 0
Missing message                                 : 0
Stream                                          : 0
Checksum                                        : 0
Unable to deliver due to transaction            : 0
```

Errors in the following categories indicate that packet reordering is happening in the network:

- No message slots available
- Missing message

You can specify the number of message slots with the `nmcom_max_msgs` tunable. For a high packet reordering network, increasing default value to 2048 or 3072 may improve replication performance. In Release 5.1SP1 and later, you can tune this value with the `vxtune` command.

VVR and network address translation firewall

VVR uses a heartbeat mechanism to detect communication failures between the Primary and the Secondary hosts. VVR uses IP addresses in the heartbeat message to send heartbeat acknowledgments.

When replicating over a Network Address Translation (NAT) based firewall, VVR must use the translated IP address, instead of the IP address in the heartbeat message. If the IP address in the heartbeat message is used, the heartbeat acknowledgment is dropped at the firewall and replication does not start.

The tunable `vol_vvr_use_nat` directs VVR to use the translated address from the received message so that VVR can communicate over a NAT-based firewall. Set this tunable to `1` only if there is a NAT-based firewall in the configuration.

Tuning VVR compression

The following tunable parameters control the compression settings on a per-system basis:

- `vol_cmpres_enabled`
- `vol_cmpres_threads`

For information on changing the value of tunables using the `vxtune` command, see the *Veritas InfoScale™ Replication Administrator's Guide*.

Tunable parameter for enabling compression

The compression feature can be enabled using one of the following methods:

- On a per-Secondary basis using the `vradmin set` command.
- On a per-system basis using the global VVR compression tunable.

For more information on enabling compression on a per-Secondary basis using the `vradmin set` command, see the section titled "Choosing the compression mode" in the *Veritas InfoScale™ Replication Administrator's Guide*.

The global VVR compression tunable, `vol_cmpres_enabled`, lets you enable or disable compression on a per-system basis. Compression is disabled by default. The `vol_cmpres_enabled` tunable can be set either on the Primary or the Secondary host, however it comes into effect only on the Primary. The per-Secondary compression setting does not override the global setting. The compression-related statistics are maintained only on the logowner node of the Primary cluster.

Note: The `vol_cmpres_enabled` tunable is not cluster-aware. In clustered environments, it is recommended that you set the tunable to 1 on all the nodes of the Primary and Secondary clusters to prevent disruption of the tunable setting in the event that the logowner node changes.

Note: If the global VVR compression tunable is enabled, but the per-Secondary setting is not, compression will be enabled but the `vradmin repstatus` and`vxprint` command outputs will not indicate whether compression is enabled or disabled. Also, the `vrstat` and `vxrlink -e stats` commands will not display the compression-related statistics. To be able to view the compression mode and compression-related statistics, you should use the per-Secondary setting to enable compression.

Tunable parameter for setting compression threads

The `vol_cmpres_threads` tunable is a per-system tunable that lets you set the number of compression threads on the Primary host or the number of decompression threads on the Secondary host between 1 and 64. The default value is 10. You can tune this setting dependent on your CPU usage.

Tuning considerations for VVR compression

The following are tuning considerations for VVR compression:

- VVR compression results in additional CPU utilization on the Primary and Secondary hosts when enabled. It is recommended that you provision for this CPU overhead by allocating a dedicated CPU.

- Reducing the number of compression and decompression threads on the Primary and Secondary hosts reduces the amount of CPU consumption.
- In cases where CPU utilization is very high, it is recommended that you either disable compression or reduce the number of compression and decompression threads that are set on the Primary and Secondary hosts.

Section 3

Getting started with File Replicator

- Chapter 14. Introducing File Replicator
- Chapter 15. Administering File Replicator

Chapter 14

Introducing File Replicator

This chapter includes the following topics:

- About File Replicator
- How File Replicator works

About File Replicator

File Replicator (VFR) enables cost-effective periodic replication of data over IP networks, giving organizations an extremely flexible storage independent data availability solution for disaster recovery and off-host processing. It enables you to maintain a consistent copy of application data at a remote location. It replicates the application writes on the file system at the source location to a remote location across any distance. If a disaster occurs at the source location, you can use the copy of the application data at the remote location and restart the application at the remote location. The host at the source location on which the application is running is known as the Primary host, and the host at the target location is known as the Secondary host.

With the flexibility of scheduling the replication intervals to match the business requirements, File Replicator tracks all updates to the file system and replicates these updates at the end of the configured time interval. VFR leverages data deduplication provided by Veritas File System (VxFS) to reduce the impact that replication can have on scarce network resources. VFR is available as one of the Veritas Replicator options with Storage Foundation and associated products on Linux. Apart from disaster recovery, VFR can be used for content distribution, off host processing, and data migration.

Note: VFR is supported on SUSE Linux Enterprise Server (SLES) and Red Hat Enterprise Server (RHEL) Linux for Storage Foundation, Storage Foundation and High Availability and Storage Foundation Cluster File System High Availability (SFCFS). The minimum file system layout Version must be 9 on both source and target systems.

Features of VFR

File Replicator (VFR) includes the following features:

- Supports periodic replication of a subset of a file system ranging from a single file to an entire file system.
- Supports reversible data transfer. The target of replication may become the source at runtime, with the former source system becoming a target.
- Supports automatic recovery from the last good successfully replicated point in time image.
- Periodically replicates changes. The interval is configurable by the user.
- Supports deduplication to increase storage efficiency on the target system.
- Supports protection of the target file system from accidental writes.

File Replicator limitations

File Replicator has the following limitations:

- Replication of compressed files on the source system does not result in compressed files on the target system.
- VFR is only supported on source and target systems with Veritas File System (VxFS) file systems with disk layout version 9.
- File systems are identified only by mount point and file system type. If a different file system is mounted in place of the original file system used, VFR may not detect the new file system.
- VFR replicates in one direction, however the replication direction can be manually changed in the event of a disaster.
- VFR does not provide any security options and must run on a private secure network if security is desired.
- The source and destination file systems must have the same file system block size.
- Cluster mounts used on a target must be managed by VCS.

- If replication is enabled for a file system, it should not be resized to a smaller size using `fsvoladm resize` command.

How File Replicator works

File Replicator replicates a set of files on a single file system. It replicates the files or a specified consistency group to a single target. The replicated files are sent over the network on a periodic basis with a period ranging from 15 minutes to a 180 minutes. The target is updated to match the current status of the source files as of the most recent start of replication using Storage Checkpoints. The changes since the last replication iteration are efficiently identified using the FCL (File Change Log) and Storage Checkpoints.

A replication job is directional. The job definition includes the source and destination, both host addresses, and the mount points. It also includes the interval at which incremental replication is scheduled. At each interval, a file system point in time image is taken and the file/directory changes are sent to the target as a stream of deltas where it is applied to have the target image mirror the point in time image of the source.

A consistency group consists of an include and exclude path and pattern list. An include list can have both paths and patterns and specifies a combination of files and directories within a file system that needs to be replicated as a single consistent unit. An exclude list specifies files and directories that must not be replicated. Both include and exclude path and pattern lists are optional. If no paths are specified in the include path list and no patterns are specified in the include pattern list, the entire file system is replicated, with the exception of any files or directories which match an exclude path or pattern.

The following directories and files from the root of the file system are not replicated and may not be specified in the exclude list:

- lost+found
- .placement_policy.xml
- quotas
- quotas.grp

At system startup, if enabled, the replication scheduler and the replication daemons are started. Once the replication job is started using the `vfradmin` command, the scheduler daemon enables the FCL if it is not already enabled, takes a file system Storage Checkpoint, and starts the replication process against the newly created Storage Checkpoint. Old Storage Checkpoints are removed once they are no longer useful.

The replication daemon is useful on machines that act as a replication target. It listens for incoming connections and applies the changes on the target file system. After every iteration, a file system Storage Checkpoint is taken. The old Storage Checkpoint is deleted once a new one is created.

The Storage Checkpoints used by VFR are data full checkpoints and will consume extra storage on both the source and the target file systems. Also note that the FCL is enabled by VFR and it is not advisable that you disable some FCL events or disable the FCL entirely. VFR will detect such events and will use a less efficient method to identify the changed files and directories. VFR will also enable the missing FCL events and turn FCL back on.

About error recovery after a site disaster or network disruption using VFR

Transient errors, such as network failures, memory allocation failures, and force unmounts are handled automatically with the next replication job run. If the last replication job run was incomplete, the target file system is rolled back to the last known good Storage Checkpoint and all of the changes are reapplied.

Note: The file system must be unmounted to promote the last Storage Checkpoint. For a cluster file system, the file system must be unmounted from all of the cluster nodes. Thus, if the file system is mounted with the cluster option, it must be managed by Cluster Server (VCS). If the file system is locked with `mntlock` mount option, it will be unlocked before file system is unmounted.

In the event of machine failure, if enabled, the replication daemons will be started by the init scripts at the system boot time. The scheduler daemon periodically scans the Veritas File System (VxFS) mount points and starts the replication job if they were started when system went down. When scheduler daemon is killed, it must be started manually using the init script or the `vfradmin` command.

See “Performing a VFR switchover” on page 421.

See “Performing a VFR failover after a disaster” on page 424.

See “Recovering a failed site if the failed source node comes up again” on page 426.

See “Recovering a failed site if a new node is assigned as the target” on page 428.

About File Replicator log files

The log files related to file replication are placed in the `/var/VRTSvxfs/replication/log/` directory. The scheduler daemon, `vxfstaskd`, creates a log file and logs important messages in the `vxfstaskd.log` file. The

scheduler daemon also logs critical messages to the syslog. The replication process logs in a file named after the mount point and the job name. The replication target daemon, `vxfsrepld`, logs important messages in the `vxfsrepld.log` file. If `ulimit` is set appropriately before daemons are started, the core dump, if any, will be stored in the `/var/VRTSvxfs/replication/diag/` directory. Old log files are deleted from the directory.

Chapter 15

Administering File Replicator

This chapter includes the following topics:

- About vfradmin utility
- Starting the vxfstaskd and vxfsrepld file replication daemons
- Protecting a target file system
- Creating a file replication job
- Creating a consistency group
- Managing a file replication job
- Displaying file replication job information
- Modifying a file replication job
- Modifying a consistency group
- Working with pattern lists
- Deleting a file replication job
- Deleting a consistency group
- Performing a VFR switchover
- Performing a VFR failover after a disaster
- Recovering a failed site if the failed source node comes up again
- Recovering a failed site if a new node is assigned as the target

About vfradmin utility

File Replicator (VFR) lets you configure and control a system to automatically replicate the data in a source file system to a matching file system on any system over the network. This replication is performed incrementally after the initial synchronization by transferring the deltas between successive versions of the data. The vfradmin utility lets you manage and control the file replication feature. The utility lets you view and modify the configuration of a replication job as well as start or stop the replication job.

The configuration and the status files for an individual file system are saved under the lost+found directory on that file system. This allows the replication system to access the data when storage devices are moved between nodes in a clustered environment.

Note: Deleting the lost+found directory on the file system also deletes the configuration and status files for the replication job, and you will not be able to start the replication for the file system. The scheduler will drop the task for running the replication job.

Before starting a replication job, you must run the scheduler daemon (vxfstaskd) and the replication target daemon (vxfsrepld).

Starting the vxfstaskd and vxfsrepld file replication daemons

The vxfstaskd daemon is a scheduler daemon that takes scheduled Storage Checkpoints, starts the replication process against newly created Storage Checkpoints, and cleans up unused Storage Checkpoints. The daemon periodically scans Veritas File System (VxFS) mount points for replication job configuration and automatically schedules any enabled jobs. The vxfstaskd daemon must be started on the source system before a replication job can be scheduled.

The replication scheduler and target daemon can be started and stopped by either using the init script located at /etc/init.d/vxfs_replication or by running the vfradmin command directly. To start both the daemons using the init script, run following command on both the source and the target system:

```
# /etc/init.d/vxfs_replication start
```

By default, the init script will not start the daemons at system bootup or when run manually. If you want to start the replication daemons, edit the replication

configuration file at `/etc/vx/vxfs_repltab` and set VXFS_REPLICATION_START=1 . The replication daemon, by default uses TCP port 56987. If this port is not free in the environment, specify a desired port in VXFS_REPLICATION_TGT_PORT=*port number*.

To stop the replication scheduler and the target daemon, run the following command:

```
# /etc/init.d/vxfs_replication stop
```

To start the `vxfstaskd` scheduler daemon manually, type the following command on the source system:

```
# vfradmin sched start
```

To stop the `vxfstaskd` daemon, type the following command on the source system:

```
# vfradmin sched stop [-f]
```

The `vfradmin sched stop` command stops the scheduler process after all the in progress replication jobs are complete. If the `-f` option is specified, all currently scheduled replication jobs are aborted. Specifying the `-f` option may leave the file system Storage Checkpoints mounted and may leave the target file system in an intermediate state.

The `vxfsrepld` daemon must be running on the systems that function as replication targets. This daemon listens on the replication port and applies the delta changes sent by the source system. The `vxfsrepld` daemon must be started on the target system before a replication job can be started.

To start the `vxfsrepld` replication target daemon manually, type the following command on the target system:

```
# vfradmin vxfsrepld start [-p port]
```

where *port* specifies the port on which the `vxfsrepld` daemon will listen for the incoming connection.

Note: By default, the replication daemon uses port 56987. If this port is not free in the environment, you must specify a different port number, using the `-p` option.

To stop the `vxfsrepld` daemon, type the following command on the target system:

```
# vfradmin vxfsrepld stop [-f]
```

The `vfradmin vxfsrepld stop` command stops the `vxfsrepld` daemon after all the in-progress replication jobs are complete. If the `-f` option is specified, all in

progress replication jobs are aborted. This may result in a file system in an intermediate state.

Note: If either daemon processes are killed and they are not being managed by VCS, they must be manually re-started using the init script or the `vfradmin` command.

Protecting a target file system

File Replicator relies on the assumption that the target file system will not be modified by anything other than the replication job. Doing so could cause replication to fail. To prevent this accidental write, a file system can be mounted with protection.

To mount a file system with protection, run the following command:

```
# mount -t vxfs -oprotected=on device_name target_mntpt
```

where *device_name* is the name of your device and *target_mntpt* is the mount point of the target file system.

When protected, a file system will behave as if it were mounted read-only. However, VFR will still be able to apply updates. This will also prevent some administrative commands from running on the file system.

The mount option should be added to the `fstab` or VCS configuration so that the file system will always be protected. If a writable copy of the file system is needed, a Storage Checkpoint can be taken and mounted read/write while the file system is still protected.

During VFR manual failover of the replication target, the old target file system needs to be remounted with the `-oprotected` option set to `off` before any applications are started.

Creating a file replication job

To create a replication job, you must configure the replication job on both the source and the target systems with the same parameters. The `vfradmin job create` command creates the configuration for the replication job. You must specify the `-s` option when creating the replication job on the source system and the `-t` option when creating the replication job on the target system.

Note: When creating a new replication job, the target file system must be empty, otherwise the full-sync of the replication job will fail with the following error:

```
target file system not empty for full sync
```

Note: If a replication job is not created on the target system, or is created with different parameters, the replication job will fail with the following error:

```
job configuration error
```

Note: Creating a replication job does not start the replication automatically. On the source system, use the `vfradmin job start` command to start the replication job.

To create a replication job, type the following command on both the source and the target systems:

```
# vfradmin job create [-p tgt_port] [-s|-t] [-d 0|1] [-n]\
  [-g cgrp_name] job_name src_addr src_mnpt tgt_addr tgt_mntpt freq
```

where:

tgt_port is the target port. If no port is specified, the default port 56987 is used. The replication target daemon must be running with the same port, otherwise the replication job will fail with a "connection refused" error.

cgrp_name is the name of the consistency group you want to associate with the replication job. If no consistency group is specified, the entire file system is replicated.

job_name is the name of the replication job you are creating.

src_addr is the IP address or DNS name of the source system. This address is used as the source for connections.

Note: The *src_addr* should not map to the localhost address (127.0.0.1) in `/etc/hosts` on the system. If this is the case, the replication job will fail with the following error:

```
Invalid Argument
```

---

src_mntpt is the mount point on the source system.

tgt_addr is the IP address or DNS name of the target system.

tgt_mntpt is the mount point on the target system.

freq is the frequency in which the replication job runs. The frequency parameter is specified in minutes and valid values range from 15 minutes to 180 minutes.

The `-d` option lets you to turn debugging off or on. The default value is 1.

When a replication job is created with the `-n` option, VFR will not use the File Change Log (FCL) feature to detect changes in the file system for replication. If FCL is turned off, VFR will not enable FCL.

Creating a consistency group

To create a consistency group, you must configure the consistency group on both the source and the target systems with the same paths. The `vfradmin cgrp create` command creates a consistency group. Only one consistency group can be associated with a replication job. If no consistency group is associated with a replication job, the entire file system is replicated. After creating a consistency group, you must set up the include and exclude lists, and then use the `vfradmin job set cgrp` command to associate the consistency group with a file system.

To create a consistency group, type the following command on both the source and the target systems:

```
# vfradmin cgrp create cgrp_name mntpt
```

where:

cgrp_name is the name of the consistency group you are creating.

mntpt is the mount point on the source or the target system.

Prior to associating a consistency group with a replication job, you must specify the paths and patterns to include or exclude using the `vfradmin cgrp include add` and `vfradmin cgrp exclude add` commands.

See "Modifying a consistency group" on page 416.

To associate a consistency group with a replication job, type the following command on both the source and the target systems:

```
# vfradmin job set cgrp job_name mnpt cgrp_name
```

where:

job_name is the name of the replication job that you want to associate the consistency group with.

mnpt is the mount point on the source or the target system.

cgrp_name is the name of the consistency group that you want to associate with the replication job.

Alternately, you can associate a consistency group while creating a replication job by specifying the `-g` option when using the `vfradmin job create` command.

Managing a file replication job

You can perform the following file replication job management tasks. The commands must be run on the source system.

Starting a file replication job

Use the `vfradmin job start` command to start replication. The command initiates replication immediately based on the frequency interval settings.

At each frequency interval, a fresh file system Storage Checkpoint is taken and replication is started against the new Storage Checkpoint. If a previous replication run has not completed, a new Storage Checkpoint is not taken and the current run is skipped.

To start a replication job, type the following command:

```
# vfradmin job start job_name \
mntpt
```

where *job_name* is the name of the replication job you want to start and *mntpt* is the mount point on the source system.

Note: Running the `vfradmin job start` command on an aborted replication job will automatically restart the job.

Recovering from a file replication job failure

Use the `vfradmin job sync` command to recover from the secondary site in the event that the primary file system is completely destroyed. The command lets you start a replication job and stops the replication job after one iteration (full or incremental) is complete.

The command can also be used to schedule a replication job using a script or a cron job.

Pausing a file replication job

Use the `vfradmin job pause` command to pause the current replication process immediately without waiting for the running iteration to complete. It does not drop the replication job from the schedule.

To pause a replication job, type following command:

```
# vfradmin job pause job_name \
mntpt
```

where*job_name* is the name of the replication job you want to pause and *mntpt* is the mount point on the source system.

Resuming a paused file replication job

Use the `vfradmin job resume` command to resume the replication process that was paused. The command replicates the data from the point where it was paused.

To resume a replication job, type following command:

```
# vfradmin job resume job_name \
mntpt
```

where*job_name* is the name of the replication job you want to resume and *mntpt* is the mount point on the source system.

Stopping a file replication job

Use the `vfradmin job stop` command to drop a replication job from the schedule and wait for running iterations to complete. This step may take time if the network is slow or if a large amount of data has changed since the last replication run. The `-n` option can be used to drop a replication job from schedule without waiting for it to complete.

To stop a replication job, type the following command:

```
# vfradmin job stop [-n] job_name \
mntpt
```

where *job_name* is the name of the replication job you want to stop and *mntpt* is the mount point on the source system.

Aborting a file replication job

Use the `vfradmin job abort` command to force cancellation of a replication job, even if the job is in progress. Aborting a replication job may leave Storage Checkpoints mounted on the source system and the target file system may be left in an intermediate state.

To abort a replication job, type the following command:

```
# vfradmin job abort job_name \
mntpt
```

where *job_name* is the name of the replication job you want to abort and *mntpt* is the mount point on the source system.

Waiting for job cycles to complete

Use the `vfradmin job wait` command to wait for the current iteration of a replication job to complete. This command can be used if a script requires the completion of the current replication cycle.

Displaying file replication job information

This section describes the VFR commands that you can use to view information about currently configured replication jobs.

Displaying file replication jobs

The `vfradmin job list` command displays the name of the currently configured replication job for a specified mount point. Specifying the `-v` option prints the current configuration information for the replication job, which includes the mode, source mount point, target mount point, frequency, and other configuration information. Specifying the `-j` option allows you to display configuration information for a specific replication job.

To display the replication job associated with a mount point, type the following command:

```
# vfradmin job list [-v] [-j job_name] mntpt
```

where *mntpt* is the mount point on the source or the target system and *job_name* is the name of the replication job you want to display.

Displaying the status of a file replication job

The `vfradmin job get status` command displays the current status of the replication job.

To display the status of a replication job, type the following command:

```
# vfradmin job get status job_name mntpt
```

where *job_name* is the name of the replication job, for which you want to display the status, and *mntpt* is the mount point on the source or the target system.

The status includes current and intermediates states.

Table 15-1 lists the current states.

Table 15-1 Current states

State	Description
idle	The replication job is in the middle of two replication iterations and if scheduled, it is waiting for the next schedule.
full-sync-running	The replication job is currently doing the first full-sync to the target file system.
full-sync-failed	The replication job failed to complete the first full-sync. The reason for the failure is displayed. This could be because of a job configuration mismatch on the source and the target, network failure or any other error. The replication log files on both the source and the target machine display additional information.

Table 15-1 Current states *(continued)*

State	Description
full-sync-paused	The replication job was paused when full-sync was running. The full-sync operation will continue when the job is resumed.
running	The replication job is currently identifying the changes since the last run and sending them over to the target machine.
failed	The replication job failed to complete the incremental sync. The associated error is displayed. The scheduler, replication job log file of the source machine, and the replication daemon target log file on the target machine display additional information.
paused	The replication job was paused either when the replication was running or when it was waiting for its next schedule. When the job resumes, the replication will continue from where it was paused.

Table 15-2 lists the intermediate states.

Table 15-2 Intermediate states

State	Description
full-sync-started	The first full -sync is started and Storage Checkpoint is being created for the full-sync.
full-sync-prepare	The Storage Checkpoint has been created for the full-sync and is now being mounted.
full-sync-complete	The full-sync is now complete and Storage Checkpoint is now being un-mounted.
full-sync-restarted	This is the same as the full-sync-started state, except that the earlier full-sync attempt failed and the Storage Checkpoint created earlier will be reused.
full-sync-restart-prepare	This is the same as the full-sync-prepare state, except that the earlier full-sync attempt failed and the Storage Checkpoint created earlier will be mounted. The replication iteration will be started in recovery mode.
started	The incremental replication run has started and the Storage Checkpoint is being created for the run.
prepare	The Storage Checkpoint for the incremental sync has been created and is now being mounted.

Table 15-2 Intermediate states *(continued)*

State	Description
complete	The incremental replication run is now complete and the Storage Checkpoint is now being un-mounted.
restarted	This is the same as the started state, except that the last incremental sync failed and the Storage Checkpoint created earlier will be reused.
restart-prepare	This is the same as the prepare state, except that the last incremental sync failed and the Storage Checkpoint created earlier will be mounted. The replication iteration will be started in recovery mode.

Displaying file replication job statistics

The `vfradmin job get stats` command displays recent and aggregated statistics of the replication job. Data included in the statistics is dependent on the options specified. If no options are specified, the command displays counters over the lifetime since replication was configured and started. Lifetime counters accumulate after every replication attempt, lagging by one time interval. Displayed fields are: files changed and synced, directories changed and synced, named attribute files synced, file data synced, errors, stats start time, last update time, and completion time.

The `-v` option displays the most recent replication interval statistics, which includes files changed, file data synchronized, errors, various time stamps, and the transfer rate. The `-fh` option formats the file data synced field in human readable format. The `vfradmin job get stats` command can only be run on the source system.

To display statistics for a replication job, type the following command:

```
# vfradmin job get stats [-v] [-fh] job_name mntpt
```

where *job_name* is the name of the replication job, for which you want to display statistics, and *mntpt* is the mount point of the source system.

Displaying a file replication Storage Checkpoint

The `vfradmin job get ckpt` command displays the most recent Storage Checkpoint used by the replication job. If the last run of the replication job was successful, the same Storage Checkpoint name will exist on the target system with the same data. Displaying the most recent Storage Checkpoint provides information

on which Storage Checkpoint replication job is currently working, or was last worked on.

To display a replication Storage Checkpoint, type the following command:

```
# vfradmin job get ckpt job_name mntpt
```

where *job_name* is the name of the replication job, for which you want to display the Storage Checkpoint, and *mntpt* is the mount point of the source or the target system.

Displaying consistency groups

The `vfradmin cgrp list` command lists all the consistency groups associated with a specified mount point. Specifying the `-v` option prints all the details of the consistency group. If the `-g` option is specified, only information related to the named consistency group is displayed.

To display the consistency groups associated with a mount point, type the following command:

```
# vfradmin cgrp list [-v] [-g cgrp_name] mntpt
```

where *cgrp_name* is the name of a specific consistency group you want to display, and *mntpt* is the mount point for which you want to display associated consistency groups.

Modifying a file replication job

Some configuration parameters including the source mount point, the target mount point, the host, and the job direction can not be changed if a replication job is currently scheduled. All other configuration changes including the source and the target IP address, replication frequency, the target port, and the debugging level can be changed when a replication job is scheduled. Unless otherwise specified, all replication job modification commands must be run on both the source and the target systems to maintain replication job configuration consistency, otherwise the replication job will fail with a job configuration mismatch error.

To update the frequency at which file system data is replicated, type the following command:

```
# vfradmin job set freq job_name mntpt freq
```

where *job_name* is the name of the replication job you want to modify, *mntpt* is the mount point of the source or the target system, and *freq* is the new frequency in minutes at which the replication job will run.

To update the source address of the replication job, type the following command:

```
# vfradmin job set srcaddr job_name mntpt  src_addr
```

where *job_name* is the name of the replication job you want to modify, *mntpt* is the mount point of the source or the target file system, and *src_addr* is the new source address for the replication job.

To update the target address of the replication job, type the following command:

```
# vfradmin job set tgtaddr job_name mntpt  tgt_addr
```

where *job_name* is the name of the replication job you want to modify, *mntpt* is the mount point of the source or the target file system, and *tgt_addr* is the new target address for the replication job.

To update the host name where the replication job can come online, type the following command:

```
# vfradmin job set host job_name mntpt host
```

where *job_name* is the name of the replication job you want to modify, *mntpt* is the mount point of the source or the target system, and *host* is the updated host name.

The *host* must be the same host name as returned by the `uname -n` command. You must update the host name to manually failover the replication job from one cluster system to another. This command is only required on the source system if a new system will become the source for the replication job. Similarly, this command is only required on the target system if a new system will become the target for the replication job.

To update the target port of the replication job, type the following command:

```
# vfradmin job set port -p tgt_port job_name mntpt
```

where *tgt_port* is the target port you want to use, *job_name* is the name of the replication job you want to modify, and *mntpt* is the mount point of the source or the target system.

If you run this command, the `vxfsrepld` daemon on the target system must be started with the updated port.

To update the replication job source mount point, type the following command:

```
# vfradmin job set srcmntpt [-t tgt_mntpt] job_name newsrc_mntpt
```

where *job_name* is the name of the replication job you want to modify and *tgt_mntpt* is the mount point of the target file system. To change the source mount point on target system, the target mount point must be specified using the `-t` *tgt_mntpt* option. The file system must be mounted on *newsrc_mntpt* on the source system

for the command to succeed. On the source system, the replication job must not be associated with a consistency group.

To update the replication job target mount point, type the following command:

```
# vfradmin job set tgtmntpt [-s src_mntpt] job_name newtgt_mntpt
```

where *job_name* is the name of the replication job you want to modify and *src_mntpt* is the mount point of the source file system. To change the target mount point on source system, the source mount point must be specified using the `-s` *src_mntpt* option. The file system must be mounted on *newtgt_mntpt* on the target system for this command to succeed. On the target system, the replication job must not be associated with a consistency group.

To update the replication job direction on the source or the target, type the following command:

```
# vfradmin job set mode [-s|-t] job_name mntpt
```

where *job_name* is the name of the replication job you want to modify and *mntpt* is the mount point on the source or the target system. You must specify the `-s` option when updating the replication job mode on the old target system. This specifies that the current machine should now behave as the source system for the replication job. You must specify the `-t` option when updating the replication job mode on the old source system. This specifies that the current machine should behave as the target system for the replication job. This command swaps the source and the target mount points and IP addresses in the replication job configuration.

Changing the replication job mode can be used with other commands in disaster recovery scenarios.

Note: For a given replication job, when a previous source machine becomes the target machine, statistics information is deleted from the source machine. New source will build the statistics information from the next replication iteration.

To set the debug level of the replication job, type the following command:

```
# vfradmin job set dbg -d [0|1] job_name mntpt
```

where *job_name* is the name of the replication job you want to modify and *mntpt* is the mount point on the source or the target system. By default, the debug level is set to 1. Setting the debug level to 1 adds diagnostic information in the job log file. The diagnostic information can provide useful information for debugging any errors. Setting the debug level to 0 will turn off logging of diagnostic information.

To turn the File Change Log (FCL) feature on or off for a replication job, type the following command:

```
# vfradmin job set fcl job_name mntpt [on|off]
```

When used with "on" option, the command modifies the replication job to use the FCL feature to detect changes in the file system. When used with "off" option the replication job will not use the FCL feature to detect changes in the file system.

Modifying a consistency group

You can modify a consistency group by adding or removing a path (file or directory) or pattern to or from the include and exclude path and pattern lists. A consistency group can be modified only if it is not associated with any replication job.

Use the following commands to modify a consistency group:

- `vfradmin cgrp include add`
- `vfradmin cgrp include remove`
- `vfradmin cgrp exclude add`
- `vfradmin cgrp exclude remove`

The `vfradmin cgrp set mntpt` command allows you to modify the consistency group mount point. You must use the `vfradmin job clear cgrp` command to disassociate the consistency group from a file replication job.

To disassociate a consistency group from a replication job, type the following command on both the source and the target systems:

```
# vfradmin job clear cgrp job_name mnpt cgrp_name
```

where:

job_name is the name of the replication job that you want to disassociate the consistency group from.

mnpt is the mount point on the source or the target system.

cgrp_name is the name of the consistency group that you want to disassociate from the replication job.

After all desired modifications are made to the consistency group, it must be re-associated with the replication job. To associate a consistency group with a replication job, type the following command on both the source and the target systems:

```
# vfradmin job set cgrp job_name mnpt cgrp_name
```

where:

job_name is the name of the replication job that you want to associate the consistency group with.

mnpt is the mount point on the source or the target system.

cgrp_name is the name of the consistency group that you want to associate with the replication job.

To add a path (file or directory) to the include list, type the following command on both the source and the target systems:

```
# vfradmin cgrp include add cgrp_name mntpt include-path1 include-path2
```

where:

cgrp_name is the name of the consistency group you are modifying.

mntpt is the mount point on the source or the target system.

include-path1 and *include-path2* are the full paths including hte mount points of the files or directories you want to add to the include list.

To add a path (file or directory) to the exclude list, type the following command on both the source and the target systems:

```
# vfradmin cgrp exclude add cgrp_name mntpt exclude-path1 exclude-path2
```

where:

cgrp_name is the name of the consistency group you are modifying.

mntpt is the mount point on the source or the target system.

exclude-path1 and *exclude-path2* are the full paths including mount points of the files or directories you want to add to the exclude list.

Note: The exclude path should be a subset of the include path.

To remove a path (file or directory) from the include list, type the following command on both the source and the target systems:

```
# vfradmin cgrp include remove cgrp_name mntpt include-path1 \
include-path2
```

where:

cgrp_name is the name of the consistency group you are modifying.

mntpt is the mount point on the source or the target system.

include-path1 and *include-path2* are the full paths including mount points of the files or directories you want to remove from the include list.

To remove a path (file or directory) from the exclude list, type the following command on both the source and the target systems:

```
# vfradmin cgrp exclude remove cgrp_name mntpt exclude-path1 \
exclude-path2
```

where:

cgrp_name is the name of the consistency group you are modifying.

mntpt is the mount point on the source or the target system.

exclude-path1 and *exclude-path2* are the full paths including mount points of the files or directories you want to remove from the exclude list.

To modify the consistency group mount point, type the following command on both the source and the target systems:

```
# vfradmin cgrp set mntpt cgrp_name old_mntpt new_mntpt
```

where:

cgrp_name is the name of the consistency group you are modifying

old_mntpt is the old mount point on the source or the target system.

new_mntpt is the new mount point on the source or the target system.

The file system must be mounted at *new_mntpt*. If the file system is not mounted at *new_mntpt*, the command will fail.

Working with pattern lists

We can have patterns in a consistency group just as we have paths.

Patterns are formed using the following symbols:

`*`	Match zero or more characters, except /
`?`	Match exactly one character, except /
`[]`	Match any one of the characters specified within the brackets. Character ranges can be specified within the brackets using a hyphen character.
`**`	Match all files and zero or more directories or subdirectories

All the files and directories matching the specified pattern are replicated. If an include or exclude path and an include or exclude pattern conflict, the include or exclude path will take precedence. For example, if the exclude pattern is `/mnt1/dir1/dir2` and the include pattern is `/mnt1/dir1/dir2/dir3/*.txt`, none of the files in `dir3` are replicated.

Table 15-3 lists the operations you can do with patterns lists.

Table 15-3 Operations on pattern lists

Operations	Description
Add a pattern (file or directory name pattern) to the include pattern list	Type the following command on both the source and the target systems: `# vfradmin cgrp include add cgrp_name mntpt -p \ include-pattern1" -p include-pattern2"` Where *cgrp_name* is the name of the consistency group you are modifying. *mntpt* is the mount point on the source or the target system. *include-pattern1* and *include-pattern2* indicates the full path including the mount points of the name patterns of files or directories that you want to add to the include pattern list. You can have one or more instances of `-p` depending upon the number of patterns used.
Add a pattern (file or directory name pattern) to the exclude pattern list	Type the following command on both the source and the target systems: `# vfradmin cgrp exclude add cgrp_name mntpt -p \ exclude-pattern1" -p exclude-pattern2"` Where *cgrp_name* is the name of the consistency group you are modifying. *mntpt* is the mount point on the source or the target system. *exclude-pattern1* and *exclude-pattern2* indicates the full path including the mount points of the name patterns of files or directories that you want to add to the exclude pattern list. You can have one or more instances of `-p` depending upon the number of patterns used. **Note:** The exclude path must be a subset of the include path.

Table 15-3 Operations on pattern lists *(continued)*

Operations	Description
Remove a pattern (file or directory name pattern) from the include pattern list	Type the following command on both the source and the target systems: `# vfradmin cgrp include remove cgrp_name mntpt -p \ include-pattern1" -p include-pattern2"` Here, *include-pattern1* and *include-pattern2* indicates the full path including the mount points of the name patterns of files or directories that you want to remove from the include pattern list. You can have one or more instances of `-p` depending upon the number of patterns used.
Remove a pattern (file or directory name pattern) from the exclude pattern list	Type the following command on both the source and the target systems: `# vfradmin cgrp exclude remove cgrp_name mntpt -p \ exclude-pattern1" -p exclude-pattern2"` Here, *exclude-pattern1* and *exclude-pattern2* indicates the full path including the mount points of the name patterns of files or directories that you want to remove from the exclude pattern list. You can have one or more instances of `-p` depending upon the number of patterns used.

Deleting a file replication job

The `vfradmin job destroy` command deletes a replication job. This command completely removes the specified job from the configuration, cleans up any saved job-related statistics, and removes any Storage Checkpoints. To successfully delete a replication job, you must run this command on both the source and the target systems when the replication job is not running.

To delete a replication job, type the following command on the source and the target systems:

```
# vfradmin job destroy job_name mntpt
```

where *job_name* is the name of the replication job you want to delete and *mntpt* is the mount point on the source or the target system.

Deleting a consistency group

The `vfradmin cgrp destroy` command deletes a consistency group. To successfully delete a consistency group, you must run this command on both the source and the target systems.

To delete a consistency group, type the following command on the source and the target systems:

```
# vfradmin cgrp destroy cgrp_name mntpt
```

where *cgrp_name* is the name of the consistency group you want to delete and *mntpt* is the mount point on the source or the target system.

Performing a VFR switchover

The following procedure performs a VFR switchover.

To perform a VFR switchover

- Promote the job that you want to become the new replication source for the switchover:

  ```
  # vfradmin job promote [-q] job_name mntpt
  ```

  where *job_name* is the name of the replication job that you want to promote and *mntpt* is the mount point of the system on which the replication job is currently running. This command reverses the job's role, such that the source becomes the target, and the target becomes the source. This command can be run from either the current replication source or the target.

  The `vfradmin job promote` command is by default an interactive command. You can specify the `-q` option to suppress the interactivity.

Note: Any configuration changes made since the last check point will be lost and may require user intervention to restart replication.

The following example performs a switchover.

Example of a VFR switchover

1 View the configured jobs of the source, `/mnt1`, before performing the promote:

```
# vfradmin job list -v -j promojob /mnt1
Replication Job: 'promojob' (ENABLED)
=====================================
Source Mount Point:     /mnt1
Source Address:         host1
Consistency group:      -
Target Address:         host2:56987
Target Mount Point:     /mnt2
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host1
Machine Role:           SOURCE
```

2 View the configured jobs of the target, `/mnt2`, before performing the promote:

```
# vfradmin job list -v -j promojob /mnt2
Replication Job: 'promojob'
===========================
Source Mount Point:     /mnt1
Source Address:         host1
Consistency group:      -
Target Address:         host2:56987
Target Mount Point:     /mnt2
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host2
Machine Role:           TARGET
```

3 Promote the target, `/mnt2`, so that it becomes the source:

```
# vfradmin job promote promojob /mnt2
UX:vxfs vfradmin: INFO: V-3-20000: Current replication direction:
UX:vxfs vfradmin: INFO: V-3-20000:  host1:/mnt1 ->
  host2:/mnt2
UX:vxfs vfradmin: INFO: V-3-20000: If you continue this command,
  replication direction will change to:
UX:vxfs vfradmin: INFO: V-3-20000:  host2:/mnt2 ->
  host1:/mnt1
UX:vxfs vfradmin: INFO: V-3-20000: Do you want to continue (Y/N)? y
UX:vxfs vfradmin: INFO: V-3-20000: Job 'promojob' promoted successfully.
```

4 Verify that the old target, `/mnt2`, is now the source:

```
# vfradmin job list -v -j promojob /mnt2/
Replication Job: 'promojob' (ENABLED)
=====================================
Source Mount Point:     /mnt2
Source Address:         host2
Consistency group:      -
Target Address:         host1:56987
Target Mount Point:     /mnt1
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host2
Machine Role:           SOURCE
```

5 Verify that the old source, `/mnt1`, is now the target:

```
# vfradmin job list -v -j promojob /mnt1
Replication Job: 'promojob'
===========================
Source Mount Point:     /mnt2
Source Address:         host2
Consistency group:      -
Target Address:         host1:56987
Target Mount Point:     /mnt1
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host1
Machine Role:           TARGET
```

Performing a VFR failover after a disaster

The following procedure performs a VFR failover after a disaster by forcing a promote of the target.

To perform a VFR failover after a disaster

- Promote the job that you want to become the new replication source for the failover:

  ```
  # vfradmin job promote -f [-q] job_name mntpt
  ```

  where *job_name* is the name of the replication job that you want to promote and *mntpt* is the mount point of the system on which the replication job was running. You must execute this command on the target node.

  Executing this command causes the job to enter the faulted state, and the target becomes the source after the force promote completes.

  The `vfradmin job promote` command is by default an interactive command. You can specify the `-q` option to suppress the interactivity.

Note: Any configuration changes made since the last check point will be lost and may require user intervention to restart replication.

The following example performs a VFR failover after a disaster.

Example of a VFR failover after a disaster

1 View the configured jobs of the target, `/mnt2`, before performing the failover:

```
# vfradmin job list -v -j promojob /mnt2/
Replication Job: 'promojob'
===========================
Source Mount Point:     /mnt1
Source Address:         host1
Consistency group:      -
Target Address:         host2:56987
Target Mount Point:     /mnt2
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host2
Machine Role:           TARGET
```

2 Attempt to promote the target, `/mnt2`:

```
# vfradmin job promote -q promojob /mnt2
UX:vxfs vfradmin: ERROR: V-3-20000: Unable to promote the job promojob.
  Use the -f option to force promotion
```

Due to a disaster occuring, you cannot promote the target.

3 Force the promote by specifying the `-f` option:

```
# vfradmin job promote -qf promojob /mnt2
```

4 Verify that the old target, `/mnt2`, is now the source:

```
# vfradmin job list -v -j promojob /mnt2/
Replication Job: 'promojob' (FAULTED)
=====================================
Source Mount Point:     /mnt2
Source Address:         host2
Consistency group:      -
Target Address:         host1:56987
Target Mount Point:     /mnt1
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host2
Machine Role:           SOURCE
```

Recovering a failed site if the failed source node comes up again

The following procedure recovers a failed site if the failed source node comes up again.

To recover a failed site if the failed source node comes up again

- Start the recovery of the site:

  ```
  # vfradmin job recover job_name mntpt
  ```

  where *job_name* is the name of the replication job that you want to recover on the failed site and *mntpt* is the mount point of the system on which the replication job failed. You can execute this command only on the new source.

The following example recovers a failed site if the failed source node comes up.

Example of recovering a failed site if the failed source node comes up

1. View the configured jobs of the failed node, `/mnt1`, before performing the recovery:

```
# vfradmin job list -v -j promojob /mnt1
Replication Job: 'promojob' (ENABLED)
=====================================
Source Mount Point:     /mnt1
Source Address:         host1
Consistency group:      -
Target Address:         host2:56987
Target Mount Point:     /mnt2
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host1
Machine Role:           SOURCE
```

2. View the configured jobs of the new source, `/mnt2`, before performing the recovery:

```
# vfradmin job list -v -j promojob /mnt2
Replication Job: 'promojob' (FAULTED)
=====================================
Source Mount Point:     /mnt2
Source Address:         host2
Consistency group:      -
Target Address:         host1:56987
Target Mount Point:     /mnt1
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host2
Machine Role:           SOURCE
```

3. Start the recovery from the new source, `/mnt2`:

```
# vfradmin job recover promojob /mnt2
UX:vxfs vfradmin: INFO: V-3-20000: Job 'promojob' recovered successfully.
```

4 Verify that `/mnt2` is now the source:

```
# vfradmin job list -v -j promojob /mnt2/
Replication Job: 'promojob' (ENABLED)
=====================================
Source Mount Point:     /mnt2
Source Address:         host2
Consistency group:      -
Target Address:         host1:56987
Target Mount Point:     /mnt1
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host2
Machine Role:           SOURCE
```

You can see that `/mnt2` is now the source because its job is now enabled instead of faulted.

5 View the configured jobs of the failed node, `/mnt1`, after performing the recovery:

```
# vfradmin job list -v -j promojob /mnt1
Replication Job: 'promojob'
===========================
Source Mount Point:     /mnt2
Source Address:         host2
Consistency group:      -
Target Address:         host1:56987
Target Mount Point:     /mnt1
Job Frequency:          15
Job Debugging:          ENABLED
Use FCL:                on
Job host:               host1
Machine Role:           TARGET
```

Recovering a failed site if a new node is assigned as the target

The following procedure recovers a failed site if a new node is assigned as the target.

To recover a failed site if a new node is assigned as the target

1. Stop the faulted job from the new source:

   ```
   # vfradmin job stop [-n] job_name mntpt
   ```

   where *job_name* is the name of the replication job that you want to stop and *mntpt* is the mount point of the system on which the replication job is currently running.

2. Create a new job:

   ```
   # vfradmin job create [-p tgt_port] [-s|-t] [-d 0|1]
   [-n] [-g cgrp_name] job_name
   src_addr src_mntpt tgt_addr tgt_mntpt freq
   ```

3. Start the new replication job:

   ```
   # vfradmin job start job_name mntpt
   ```

   where *job_name* is the name of the replication job that you want to start and *mntpt* is the mount point on the source system.

Section 4

Analyzing your environment with Volume Replicator Advisor

- Chapter 16. Introducing Volume Replicator Advisor (VRAdvisor)
- Chapter 17. Collecting the sample of data
- Chapter 18. Analyzing the sample of data
- Chapter 19. Installing Volume Replicator Advisor (VRAdvisor)

Chapter 16

Introducing Volume Replicator Advisor (VRAdvisor)

This chapter includes the following topics:

- Audience
- Related Veritas InfoScale documents
- Overview of VRAdvisor
- How VRAdvisor works

Audience

This guide is intended for system administrators who are responsible for installing, configuring, and setting up replication using VVR.

This guide assumes that the user has:

- A basic understanding of system administration.
- Working knowledge of the VVR product.

This document guides you through the process of installing VRAdvisor and then evaluating various parameters using the data collection and data analysis process. This document describes procedures using both the graphical user interface and the command-line interface, as applicable, on the different platforms.

Related Veritas InfoScale documents

For more information on any of the topics presented in this guide, refer to the Veritas Volume Manager (VxVM) documentation sets and the *Storage Foundation Release Notes*.

Overview of VRAdvisor

Volume Replicator Advisor (VRAdvisor) is a planning tool that helps you determine an optimum Volume Replicator (VVR) configuration. This document assumes that you understand the concepts of VVR.

This document provides information on installing and using this tool on different platforms. Wherever applicable, the information that is specific to a platform has been appropriately indicated. For Windows, note that the Veritas Volume Manager (VxVM) has been renamed to Storage Foundation for Windows (VSFW) from Release 4.1 onwards.

Planning is the key to successfully configuring VVR. To set up an optimum configuration, you must understand the components of VVR and their interactions with each other. In addition, you must consider the factors that are specific to your environment while planning your VVR configuration.

The important factors to consider while planning include:

- Needs and constraints of the business
- Application characteristics
- Mode of replication
- Network characteristics

These factors are dependent on each other and these dependencies must be considered during planning. For example, if your business requires that the data on the Secondary to be as up to date with the Primary as possible, you must choose synchronous mode and provide enough network bandwidth to handle the peak application write rate on the Primary. Or, if the available network bandwidth is less than the peak write rate of the application, you must choose asynchronous mode of replication. Also, the size of the Storage Replicator Log (SRL) must be able to handle the Secondary outages and network outages for the given application characteristics. VRAdvisor considers these dependencies and enables you to determine the parameters to suit your VVR environment.

VRAdvisor does the following:

- Collects a sample of data that reflects the application characteristics.

- Analyzes the sample of the application characteristic and calculates the size of the SRL and the network bandwidth required for replication.
- Enables you to perform a What-if Analysis by varying the needs and constraints of your business, based on your future requirements.

Note that the replication log of VVR is referred to as SRL (Storage Replicator Log) on UNIX and as Replicator Log on Windows. The terms SRL and Replicator Log are used interchangeably in the document.

How VRAdvisor works

Using VRAdvisor for planning involves collecting a sample of data that represents the application write rate and analyzing this sample of data based on factors, such as the network bandwidth and network outage. VRAdvisor considers the worst case situations when analyzing data, which results in an optimum configuration for VVR. Working with VRAdvisor data collection, data analysis, and what-if analysis.

See "Data collection" on page 433.

See "Data analysis" on page 433.

See "What-if analysis" on page 434.

Data collection

VRAdvisor uses a sample of data for analysis; the sample of data must be available in an appropriate format required by VRAdvisor. To collect a sample of data that represent the application write rate, we recommend that you collect the sample of data for a period of seven to fourteen days. Make sure that the collection period includes times of peak usage for your application, so that the collected data reflects your environment.

In the data collection mode, VRAdvisor collects the sample of data in the appropriate format required by VRAdvisor. You can also collect the sample of data using the data collection script provided. The data collection script uses the appropriate command at the operating system level to collect the data, and also converts the data to the appropriate format required by VRAdvisor.

See "About collecting the sample of data" on page 435.

Data analysis

In the data analysis mode, VRAdvisor analyzes the sample of data that you have collected, based on the following factors specified by you:

- Available network bandwidth

- Network outage duration
- Secondary outage duration

After analyzing the data, VRAdvisor displays a graphical as well as textual representation of the results in a separate window.

See "About analyzing the sample of data" on page 447.

What-if analysis

The What-if analysis feature enables you to perform additional calculations, to plan for future requirements or alternative scenarios. You can vary the parameters and recalculate the results according to different criteria. For example, you can vary the network bandwidth parameter to see what effect it would have on the SRL size. Or, you can specify a potential SRL size and see how much network bandwidth would be required for that SRL size.

See "Performing What-if analysis" on page 456.

Chapter 17

Collecting the sample of data

This chapter includes the following topics:

- About collecting the sample of data
- Collecting the sample of data on UNIX
- Collecting the sample of data on Windows

About collecting the sample of data

You need to collect data write samples that can be used with the VRAdvisor wizard. VRAdvisor uses the sample of data for analysis.

Best practices for collecting the sample of data

When collecting the sample of data, observe the following best practices:

- We recommend that you collect the samples of data using the volumes that are to be part of the VVR configuration you are planning to set up.
- To collect a representative sample of data, we recommend that you collect the sample of data over a period of 7 to 14 days.

  Note: The data must be collected for a minimum of seven days.

- Make sure that the collection period includes times of peak usage for your application, so that the collected data reflects your actual requirements.

- In a shared disk group, collect a sample of data on each node. Start the collection at the same time on each node, for the same period. The resulting files can then be analyzed together.

VRAdvisor calculates an optimum size of the Storage Replicator Log (SRL) and the network for your VVR configuration using a sample of the write statistics.

Depending on the operating system on which you are collecting data, you can either collect the sample of data using the VRAdvisor wizard, commands, or the data collection script. For details, refer to the section for your platform.

See "Collecting the sample of data on UNIX" on page 436.

See "Collecting the sample of data on Windows" on page 444.

Collecting the sample of data on UNIX

VRAdvisor can be used to collect a sample of data for analysis. You can collect data using the VRAdvisor wizard, vxstat, or the data collection script. To use the VRAdvisor wizard to collect data, you must install VRAdvisor on the system for which you are collecting data. If you do not plan to install VRAdvisor on the system, you can use the data collection script to collect data and use VRAdvisor on another system to analyze the results.

On UNIX, collect the sample of data using one of the following methods:

- Collecting data using the VRAdvisor wizard
- Collecting data using the vxstat command
- Collecting data using the data collection script

Prerequisite for collecting the sample of data

Before collecting the sample of data, observe the following prerequisite:

- If you use Veritas Volume Manager (VxVM) volumes, make sure that you import the disk group containing the required volumes onto your system.

Supported locales

VRAdvisor requires the data to be collected in a supported locale. Before using any of the methods for collecting data, set the locale to a supported locale. For UNIX, VRAdvisor supports the following locales:

English:

C

en_US.UTF-8

ios_8859_1

Japanese:

ja

ja_JP.PCK

ja_JP.UTF-8

ja_JP.eucJP

Any of the methods of data collection include a date stamp with each sample of data. When you use VRAdvisor to analyze the sample data file, VRAdvisor uses the date stamp to determine the data collection interval. To enable VRAdvisor to automatically determine the data collection interval, the date must be displayed in one of the following formats.

Date formats for English locales:

Fri Oct 1 14:37:13 2004

Fri Oct 1 14:37:13 PDT 2004

Friday October 1 17:37:13 PDT 2004

Date formats for Japanese locales:

2004年 09月 24日 16時 21分 23秒
2004年 09月 24日 (金) 16時 20分 59秒

If VRAdvisor cannot determine the data collection interval, it prompts you to specify the data collection interval.

Collecting data using the VRAdvisor wizard

The VRAdvisor wizard supports collecting data for all disks or disk groups on the host on which VRAdvisor is installed.

To collect data using the VRAdvisor wizard

1. Set the locale to a supported locale. For example:

```
# export LC_ALL=C
```

2. Change directory as follows:

```
# cd /opt/VRTSvradv/bin
```

3 Launch the VRAdvisor wizard on Solaris, using the following command:

```
# ./vradvgui
```

4 In the Welcome page, select **Data Collection**, and then click **Next**.

5 Specify the requirements for the data to be collected.

See "Specifying the data to be collected" on page 438.

6 Click **Next**. The Confirmation message appears.

7 To start the data collection process immediately, click **Yes**. To go back and make any changes, click **No**.

The Data Collection Summary page indicates that the data collection has started. It also displays a summary of the specifications you entered for the data collection.

8 Click **Finish**. VRAdvisor continues to collect data for the specified duration, although the wizard window closes.

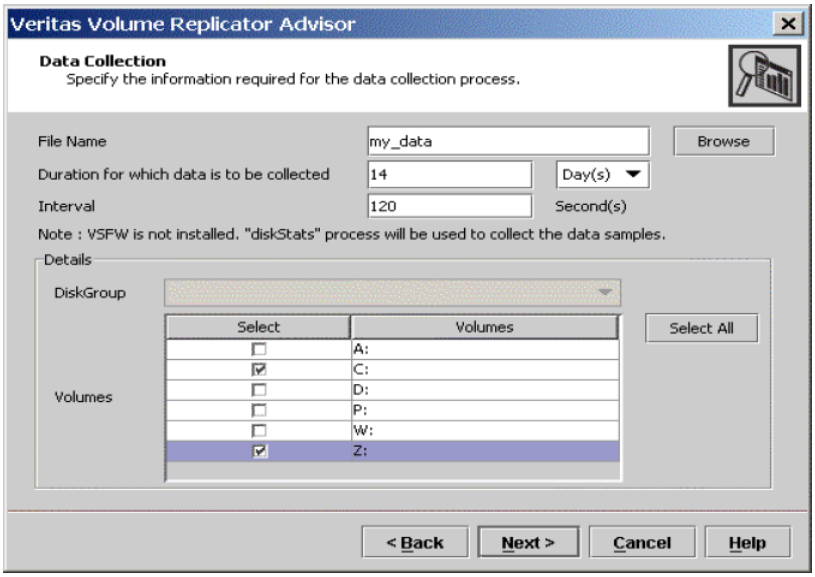
The data collection wizard displays an appropriate error message if it is unsuccessful in starting the data collection process. Select **Cancel**, fix the reported error and launch the data collection wizard again.

After the data collection completes, the file specified by **File Name** contains the sample of data in a format that can be used for analysis by VRAdvisor. You can proceed to analyze the collected data using VRAdvisor.

See "About analyzing the sample of data" on page 447.

Specifying the data to be collected

Use the VRAdvisor wizard to collect the sample data for analysis. Specify the data to be collected on the Data Collection page.



Complete the Data Collection page, by providing the following information:

File Name	Enter the name of the file where the data write samples will be collected. Make sure the name is not being used by another application. If a file already exists with that filename, or if the path is incorrect, a message is displayed.
Duration for which data is to be collected	Enter the duration in days or hours. The default value is 14 days. The maximum duration is 30 days.
Interval	Enter a value, in seconds, to indicate the frequency at which you want the data to be collected. The default value is 120 seconds.
Details	If you have VxVM installed, select the appropriate disk group in Disk Group. If you do not have VxVM installed, the Disk Group selection is not available. On Windows, the Disk Group field is not available. Only volumes with drive letters are displayed. Select the required volumes individually, or click **Select All** to select all of the available volumes in the selected disk group.

The Data Collection Summary page indicates that the data collection has started. It also displays a summary of the specifications you entered for the data collection.

Collecting data using the vxstat command

If you do not want to install VRAdvisor and VxVM is installed on your system, use the `vxstat` command to collect data.

To collect data using vxstat

1. Set the locale to a supported locale. For example:

   ```
   # export LC_ALL=C
   ```

2. To collect the data in the format required for analysis, use the following command with exactly the parameters shown:

   ```
   # vxstat -g dgname -i interval -c count volumes > filename
   ```

   where:

   interval is the data collection interval in seconds.

   count is the number of samples to collect.

   volumes is a list of volume names separated by spaces.

   For example, use the following command to collect a sample of data every 120 seconds and to collect 5040 samples. The volumes are the data volumes `hr_dv01` and `hr_dv02` in the disk group `hrdg`. The results are collected in the file `vra_in`.

   ```
   # vxstat -g hrdg -i 120 -c 5040 hr_dv01 hr_dv02 > vra_in
   ```

   After the data collection completes, the file *filename* (for example, *vra_in*) contains the sample of data in the vxstat format, which can be used for analysis by VRAdvisor. You can proceed to analyze the collected data using VRAdvisor.

   See "About analyzing the sample of data" on page 447.

Collecting data using the data collection script

To collect sample data for analysis by VRAdvisor, you need not have VRAdvisor installed on the system. Use the data collection script to collect data for AIX, HP-UX, Linux, or Solaris systems.

The data collection script performs the following tasks:

- collects sample data using the appropriate method for the host. If VxVM is installed, the script uses the `vxstat` command to collect data. If VxVM is not

installed, the script uses the appropriate operating system command to collect data:

- AIX: uses `lvmstat` to collect data for all logical volumes.
- HP-UX, Linux: uses `sar` to collect data for all disks.
- Solaris: uses `iostat` to collect data for all disks.

- stores the collected data and its associated metadata in a format that can be analyzed by VRAdvisor.
- sends notifications by email when an error occurs or when the data collection has successfully completed.

To collect data using the script

1. Set the locale to a supported locale. For example:

   ```
   # export LC_ALL=C
   ```

2. Copy the data collection script and ALL required files to the system on which you plan to collect data, from one of the following locations:

   - `storage_foundation/tools/vradvisor/scripts/` directory of the Veritas product disc
   - `scripts` directory of a Solaris system where VRAdvisor is installed. By default, the `/opt/VRTSvradv/scripts` directory.
   - `scripts` folder of a Windows system where VRAdvisor is installed. By default, the folder `Program Files/VERITAS/Volume Replicator Advisor/scripts`

Make sure to copy all the files in the directory.

3 To collect the data in the format required for analysis, use the following script:

```
# sh vra_data_collection.sh [-g dgname] [-i interval] \
 [-t duration] [-d dirname] [-v volume_list] \
 [-m maillist]
```

where:

dgname is a comma-separated list of disk groups for which data needs to be collected, if VxVM is installed. The default is all disk groups on the host.

interval is the data collection interval in seconds. The default is 120 seconds.

duration is the duration for which to collect data, in hours or days. Specify the suffix "h" for hours, "d" for days. The default is 14 days.

volume_list is a comma-separated list of volume names for which data needs to be collected, if VxVM is installed. The default is all volumes on the host.

dirname is the directory in which to store the collected data files. If no directory is specified, the files are stored in the current working directory.

maillist is a comma-separated list of email-ids to receive notifications. By default, the script sends an email notification to the root user of the host.

To show help for the script options

- For help on the script options, use the following command:

```
# sh vra_data_collection.sh -h
```

Files for the collected data

After the data collection completes, the collected data file contains the sample of data in a format that can be used for analysis by VRAdvisor. The file names are as follows:

- *hostname_dgname_timestamp*.vxstat

  Indicates the file contains `vxstat` output. This is the default if VxVM is installed on the system where data is collected. Any associated metadata is in a file with the same name plus the extension `.meta`.

- *hostname_dgname_timestamp*.vra

  Indicates that the data is in VRAdvisor CSV format. The script converts the data to this format if a command other than `vxstat` was used to collect the data. Any associated metadata is in a file with the same name plus the extension `.meta`.

After the data collection completes, you can proceed to analyze the collected data using VRAdvisor.

See "About analyzing the sample of data" on page 447.

Examples of collecting data with the data collection script

The following examples show collecting data with the collection script.

Example 1

Collect data every minute for 30 hours.

```
# sh vra_data_collection.sh -i 60 -t 30h
```

Example 2

Store the collected data in directory `samples` and notify `abc@example.com`.

```
# sh vra_data_collection.sh -d samples -m abc@example.com
```

Example 3

Collect data for the VxVM disk groups `dg1` and `dg2`.

```
# sh vra_data_collection.sh -g dg1,dg2
```

Example 4

Collect data for the VxVM volumes `vol1`, `vol2`, and `vol3` in the disk group `dg1`.

```
# sh vra_data_collection.sh -g dg1 -v vol1,vol2,vol3
```

Example 5

Collect data for the VxVM volumes `vol1` and `vol2` in the disk group `dg1`, every 5 minutes, for 15 days. Store collected data in directory `samples`. Notify `abc@example.com` and `pqr@example.com`.

```
# sh vra_data_collection.sh -d samples -i 300 -t 15d -g dg1 \
  -v vol1,vol2 -m abc@example.com,pqr@example.com
```

Collecting the sample of data on Windows

VRAdvisor can be used to collect and analyze a sample of data. You can collect data using the VRAdvisor wizard or the `diskStats` command. To use VRAdvisor to collect data, you must install VRAdvisor on your system. If you do not plan to install VRAdvisor on your system, use the `diskStats` command to collect data.

On Windows, collect the sample of data using one of the following methods:

- Collecting the sample of data using the VRAdvisor wizard
- Collecting the sample of data using the diskStats command

Prerequisite for collecting the sample of data

Before collecting the sample of data, observe the following prerequisite:

- If you are using VSFW volumes, then ensure that you import the disk group containing the required volumes onto your system.

Collecting the sample of data using the VRAdvisor wizard

This section describes collecting the sample of data using the VRAdvisor wizard.

To collect data using the VRAdvisor wizard

1. To launch the VRAdvisor wizard on Windows, select **Start > Programs > VERITAS > Volume Replicator Advisor > VRAdvisor Wizard**.
2. In the Welcome page, select **Data Collection** and then click **Next**.

   The Data Collection page appears.
3. Specify the requirements for the data to be collected.

   On Windows, only the `diskStats` command will be used to collect data.

   See "Specifying the data to be collected" on page 438.
4. Click **Next**. The Confirmation message appears.

5 To start the data collection process immediately, click **Yes**. To go back and make any changes, click **No**.

The Data Collection Summary page indicates that the data collection has started. It also displays a summary of the specifications you entered for the data collection.

6 Click **Finish**. VRAdvisor continues to collect data for the specified duration, although the wizard window closes.

The data collection wizard displays an appropriate error message if it is unsuccessful in starting the data collection process. Select **Cancel**, fix the reported error and launch the data collection wizard again.

After the data collection completes, the file specified by **File Name** contains the sample of data in a format that can be used for analysis by VRAdvisor. You can proceed to analyze the collected data using VRAdvisor.

See "About analyzing the sample of data" on page 447.

Collecting the sample of data using the diskStats command

On Windows, use the `diskStats` command to collect the data required for analysis. This command can be used to collect data whether or not the Storage Foundation is installed on the system. The `diskStats` utility is installed in the following location:

```
\VERITAS\Volume Replicator Advisor\bin\diskStats.exe
```

To collect data using the diskStats command

1. Navigate to the specified path:

```
\VERITAS\Volume Replicator Advisor\bin\
```

2. At the prompt, enter the following command with exactly the parameters shown:

   The `diskStats` command can accept only drive letters of the volumes as inputs. Volume names or mount points are not supported. Volumes created by any application are supported.

```
diskStats  [-i interval [-c count]] \
 <drive 1> [[drive 2][drive 3]... ]
```

   The command will display the output on the console.

   To save the output to a file, you can redirect the output to a named file using the command:

```
diskStats [-i interval [-c count]] \
 <drive 1> [[drive 2][drive 3]... ] > <filename>
```

   After the data collection completes, the file *filename* contains the sample of data in the `diskStats` format, which can be used for analysis by VRAdvisor. You can proceed to analyze the collected data using VRAdvisor.

   See "About analyzing the sample of data" on page 447.

Chapter 18

Analyzing the sample of data

This chapter includes the following topics:

- About analyzing the sample of data
- Launching the VRAdvisor wizard
- Analyzing the collected data
- Understanding the results of the analysis

About analyzing the sample of data

You can use VRAdvisor to analyze the sample of data that you have collected. VRAdvisor analyzes the sample of data according to parameters that you specify such as available network bandwidth and network outage. In addition, VRAdvisor enables you to perform a What-If analysis by varying the values of the parameters. The output of the analysis gives the network bandwidth required to replicate in synchronous mode, and the SRL size required for a given bandwidth and for the given outages to replicate in asynchronous mode. The results of the analysis help you to set up an optimum configuration for VVR.

See “Sizing the SRL” on page 77.

VRAdvisor enables you to analyze data collected on any of the supported platforms. However, to analyze the data, you must install and use VRAdvisor on either a Solaris or a Windows system.

See “VRAdvisor System requirements” on page 461.

Prerequisites for analyzing the sample of data

Before analyzing the sample of data, observe the following prerequisites.

- All the files to be analyzed must be present in a single directory.
- The sample of data must be available in a format required by VRAdvisor.

  VRAdvisor accepts the following formats:

  - `vxstat` output
  - `diskStats` output
  - VRAdv CSV format (used by VRAdvisor wizard or the UNIX data collection script)

See "Analyzing the collected data" on page 448.

Launching the VRAdvisor wizard

You can launch the VRAdvisor wizard on Solaris or on Windows.

To launch the VRAdvisor wizard on Solaris

1. Set the locale to a supported locale. For example:

   ```
   # export LC_ALL=C
   ```

2. Change directory as follows:

   ```
   # cd /opt/VRTSvradv/bin
   ```

3. Launch the VRAdvisor wizard, using the following command:

   ```
   # ./vradvgui
   ```

To launch the VRAdvisor wizard on Windows

1. Set the locale to a supported locale. For example:

   ```
   # export LC_ALL=C
   ```

2. Choose **Start > Programs > VERITAS > Volume Replicator Advisor > VRAdvisor Wizard**.

Analyzing the collected data

This section describes how to analyze the collected data.

To analyze the collected data using VRAdvisor

1. Launch the VRAdvisor wizard.

   See "Launching the VRAdvisor wizard" on page 448.

2. In the Welcome page, select **Analysis**, and then click **Next**.

3. In the Directory Specification page, specify the directory, and then click **Next**.

   See "Specifying the location of the data files" on page 450.

4. In the File Selection page, select the files to be analyzed, and then click **Next**.

   See "Selecting the data files to be analyzed" on page 450.

5. In the Block Size and Collection Interval Specification page, specify the metadata, and then click **Next**.

   See "Specifying block size and collection interval for the data files" on page 451.

6. In the Volume or Disk Selection page, select the volumes or disks to be analyzed, and then click **Next**.

   See "Selecting volumes or disks for analysis" on page 452.

7. The RVG Summary page displays the disks or volumes that were selected for analysis. The disks or volumes for each analyzed file are grouped under an RVG name.

   Click **Back** to modify the selections, or click **Next** to continue.

   See "Specifying the data for analysis" on page 449.

8. In the Network Parameters for Analysis page, specify the parameters that apply to all defined RVGs, and then click **Next**.

   See "Specifying the parameters for analysis" on page 452.

9. Complete the RVG Specific Parameters page, and then click **Next**.

   See "Specify the RVG-specific parameters" on page 453.

10. In the Summary of Inputs page, click **Back** to modify the parameters, or select **Analyze** to start the analysis.

    See "Summary of inputs" on page 453.

    VRAdvisor displays the results of the analysis for the selected data files.

Specifying the data for analysis

When doing the analysis, you need to specify the following information to VRAdvisor:

- location of the data files

- which data files to analyze
- the metadata associated to the data files
- which volumes or disks to analyze

The RVG Summary page displays the disks or volumes that were selected for analysis. The disks or volumes for each analyzed file are grouped under an RVG name.

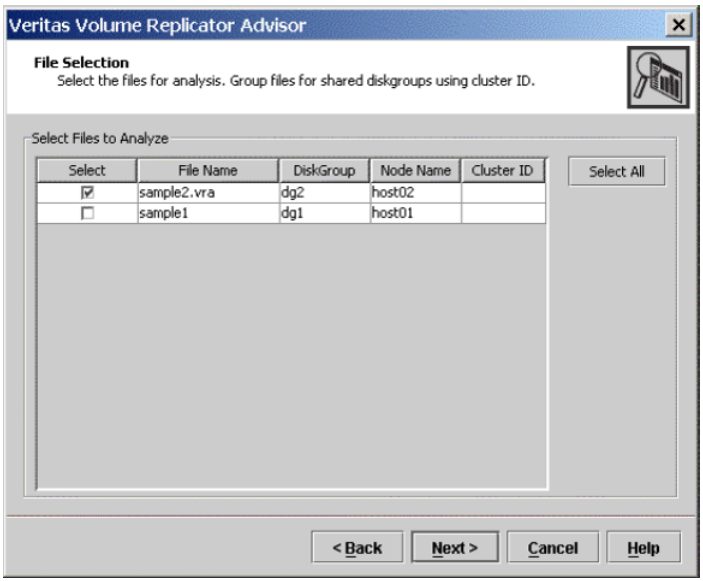
Specifying the location of the data files

In the Directory Specification page, enter the name of the directory containing the data files to be analyzed. All of the files to be analyzed must be in the same directory.

The specified directory must contain the data files and any metadata files associated with each data file. The associated metadata and data files must have the same name except for the extension. Metadata files must have the extension `.meta`.

Selecting the data files to be analyzed

In the File Selection page, VRAdvisor displays the list of files in a table.



Select the files to be analyzed as follows:

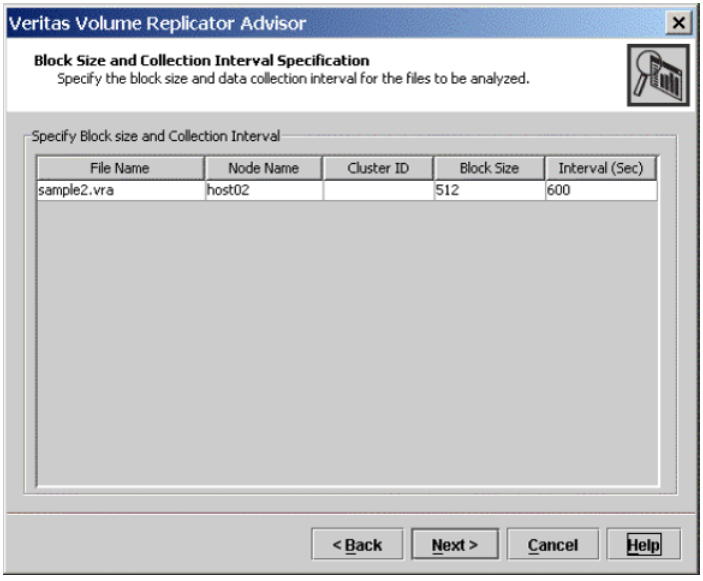
- Files containing information from nodes that will use the same network bandwidth for replication should be analyzed together. Otherwise, files should not be

selected together. In order for the files to be analyzed together, the data collection for each node must start at the same time.

- To analyze files for nodes within a shared disk group, specify a cluster ID for each shared disk group. The analysis wizard uses the cluster ID field to group the files. During the analysis, I/O statistics for files sharing the same cluster ID are considered together.
- Enter the DiskGroup name, Node Name and Cluster ID, if necessary.

Specifying block size and collection interval for the data files

Use the Block Size and Collection Interval Specification page of the VRAdvisor wizard to specify the metadata.



If the data was collected using the data collection script for UNIX platforms, the generated files contain metadata such as block size, and data collection interval.

If the files do not contain metadata, because the data was collected using operating system commands or the VRAdvisor Wizard, enter the appropriate metadata:

- Specify the block size, if required.
- If no timestamps are present in the file, or if VRAdvisor is unable to parse the timestamps, specify the interval used during the data collection.

Selecting volumes or disks for analysis

In the Volume or Disk Selection page, select the tab for each selected file. For each file, the wizard lists the disks or volumes for which data has been collected.

When selecting disks or volumes, make sure you do not select:

- RAID-5 volumes because these are not supported.
- Sub-level volumes (if the volumes are layered volumes). Select only the top-level volumes.
- The volume that you intend to use as the SRL.
- Drives or volumes containing high-activity data that will not be replicated.
- Using VRA to analyze data from drives or volumes containing high-activity data that will not be replicated leads to erroneous results.

Specifying the parameters for analysis

Specify the parameters to be used for analyzing the data.

The parameters are one of the following types:

- Network parameters
  See "Specifying the network parameters for analysis" on page 452.
- RVG-specific parameters
  See "Specify the RVG-specific parameters" on page 453.

The Summary of Inputs page displays the parameters you have specified.

See "Summary of inputs" on page 453.

Specifying the network parameters for analysis

In the Network Parameters for Analysis page, specify the parameters that apply to all defined RVGs.

- **Network Bandwidth Available for Replication** indicates the total bandwidth of the network across which you are replicating. Enter the network bandwidth that will be available for replication. Select the unit for the network bandwidth from the drop-down list. The default is 100 Mbps.

Note: Before specifying the network bandwidth you must also consider the loss of available bandwidth because of the TCP-IP/UDP headers, because VRAdvisor does not handle this.

- **Network Outage Duration** indicates the maximum expected outage times applicable for all defined RVGs, for example, the time during which the network link is unavailable for the network that is used by all of the RVGs for replication. Enter the duration of the network outage in days, hours, or minutes. The default is zero.

Specify the RVG-specific parameters

Complete the RVG Specific Parameters page, and then click **Next**.

For each RVG, select the tab and specify the following parameters:

- **Bandwidth Limit** indicates the bandwidth throttling for that RVG. The default is 0 (zero), which indicates that no bandwidth limit applies.
- **Secondary Outage Duration** indicates the maximum expected outage times specific to that RVG, for example, the time during which the Secondary host for the RVG is unavailable. Enter the outage duration in days, hours, or minutes. The default is one hour.
- **Apply to all RVG(s)** indicates that the same bandwidth limit and outage duration apply to all RVGs. Select this check box to enable the All tab and disable the RVG-specific tabs.

Summary of inputs

The Summary of Inputs page displays. The Total Outage Duration column shows the sum of the Network Outage Duration and the Secondary Outage for that RVG.

In the Summary of Inputs page, click **Back** to modify the parameters, or select **Analyze** to start the analysis.

Understanding the results of the analysis

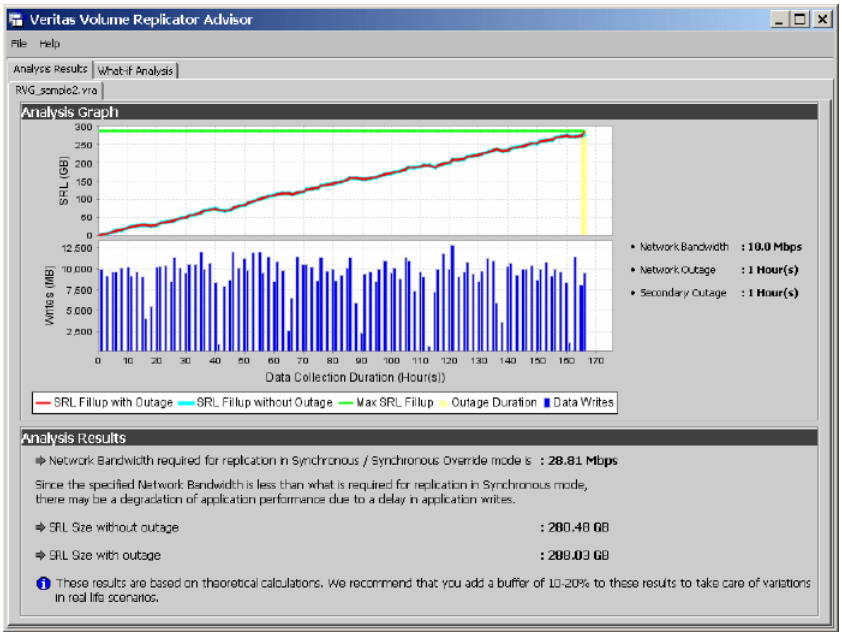
After the analysis completes, VRAdvisor displays the results of the analysis and also enables you to recalculate the results after changing some parameters.

The results are described in the following sections:

- Viewing the analysis results
- Recalculating the analysis results
- Recording and viewing the results

Viewing the analysis results

After the analysis completes, the Analysis Results page displays by default.



The Analysis Results page displays the results of the analysis for each RVG. Select the tab for the RVG to display the results for that RVG.

The results pages display the following information:

- Analysis graph
- Analysis results

Analysis graph

The Analysis Graph section shows the following information:

- The top graph shows the SRL fillup in megabytes (MB) on the y-axis. The fillup rate is shown both with the specified outages and with no outage. The x-axis shows the data write duration values. The peak SRL fillup size is shown against a max outage window, displayed in yellow, which indicates the worst case scenario.

Note: If the SRL fillup value in the graph is steadily increasing, with the maximum at the last data write duration, it indicates that you do not have sufficient network bandwidth for the amount of data writes in the sample of data.

- The bar graph shows the values of the Application Writes in bytes for the y-axis. The x-axis shows the data write duration values.
- To the right of the graphs, the page displays the values you specified for network bandwidth and the outage parameters.

Analysis results

The Analysis Results section displays the following information:

- Network bandwidth required for synchronous replication.
  If the required bandwidth is more than the bandwidth that you specified, then VRAdvisor displays a message to indicate that the performance of the application writing to the disk writes will be affected.
- The required SRL size with the specified outage.
- The required SRL size with no outage.

Note: We recommend that you add a 10-20 percent buffer to the values calculated by VRAdvisor when setting up your VVR configuration. VRAdvisor analyzes the data based on the specified values, which could be affected by factors that VRAdvisor does not consider, such as TCP/IP headers overhead, or network congestion.

Recalculating the analysis results

You can recalculate the analysis results in the following ways:

- Applying different parameters to the existing sample of data
- Performing What-if analysis

Applying different parameters to the existing sample of data

You can recalculate the analysis results by changing the values you specified for the network bandwidth and the outage durations.

To recalculate the analysis results

1. To change the values you specified, select **File > Change Inputs**.
2. On the Network Parameters for Analysis page, specify new values for any of the fields as required. Click **Next** to specify RVG-specific parameters, or click **Back** to change volume or disk selection.

3 Continue using the **Next** and **Back** buttons to navigate through the input pages and change values as required.

   See "Analyzing the collected data" on page 448.

4 When the values are correct, click **Next** to navigate to the Summary of Inputs page.

5 Select **Analyze** to start the analysis.

   VRAdvisor performs the analysis of the data using the changed values and displays the results.

Performing What-if analysis

After checking the analysis results, you can use the What-if Analysis page to do additional calculations, to plan for future requirements or alternative scenarios.

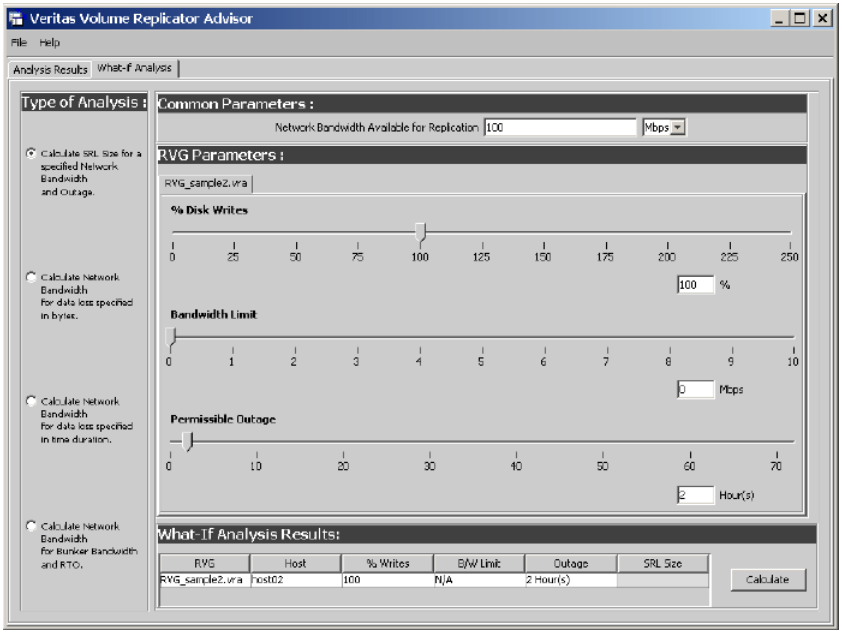
You can vary the parameters and recalculate the results according to different criteria. For example, you can vary the network bandwidth parameter to see what effect it would have on the SRL size. Or, you can specify a potential SRL size and see how much network bandwidth would be required for that SRL size.

Note: Before specifying the network bandwidth, you must also consider the loss of available bandwidth due to the TCP-IP/UDP headers, because VRAdvisor cannot manage this.

What-if Analysis also enables you to vary the percentage of disk writes as compared to the sample of data that was analyzed. For example, if you anticipate that your future needs will involve twenty percent more disk writes, set the percentage of disk writes to 120% and recalculate.

To recalculate results using the What-If Analysis

1 Select the What-If Analysis tab.



2 Select the appropriate option on the left side of the What-If Analysis page to recalculate results, as follows:

- **Calculate SRL Size for a specified Network Bandwidth and Outage.**
  See "Calculating the SRL Size for a specified Network Bandwidth and Outage" on page 458.
- **Calculate the Network Bandwidth for data loss specified in bytes.**
  See "Calculating the Network Bandwidth for data loss specified in bytes" on page 458.
- **Calculate Network Bandwidth for data loss specified in time duration.**
  See "Calculating the Network Bandwidth for data loss specified in time duration" on page 458.
- **Calculate Network Bandwidth for Bunker and RTO.**
  See "Calculating the Network Bandwidth for Bunker and RTO" on page 458.

The right side of the page displays the parameters you can specify for each option, and the corresponding slider bars.

3 In the Common Parameters section, change the bandwidth value shared by all RVGs.

4 In the RVG Parameters section, select the tab for the RVG that you want to change, and then use the slider bar to specify the value for each parameter. Each slider has a default range of values, which can be customized using the **Preferences** option that is available from the **File** menu.

   See "Changing the value ranges on the slider bar " on page 459.

5 Click **Calculate** at the lower region of the page. The What-if Analysis Results are displayed in this section.

Calculating the SRL Size for a specified Network Bandwidth and Outage

On the What-if Analysis page, use the **Calculate SRL Size for a specified Network Bandwidth and Outage** option to calculate the SRL size for a specified network bandwidth and outage duration.

Available parameters for this option are % Disk Writes and Permissible Outage.

Calculating the Network Bandwidth for data loss specified in bytes

On the What-if Analysis page, use the **Calculate the Network Bandwidth for data loss specified in bytes** option to calculate the network bandwidth that would be required to minimize the amount of data loss at the Primary host.

Available parameters for this option are % Disk Writes and Data loss in bytes.

Calculating the Network Bandwidth for data loss specified in time duration

On the What-if Analysis page, use the **Calculate Network Bandwidth for data loss specified in time duration** option to calculate the network bandwidth that would be required to minimize the amount of data loss at the Primary host.

Available parameters for this option are % Disk Writes and Data loss in time.

Calculating the Network Bandwidth for Bunker and RTO

In a Bunker replication setup, the available bandwidth determines the RPO and the RTO that can be achieved after a disaster. On the What-if Analysis page, use the **Calculate Network Bandwidth for Bunker and RTO** option to calculate the bandwidth required between the Primary and the Secondary, and between the Bunker and the Secondary, based on the desired RPO and RTO.

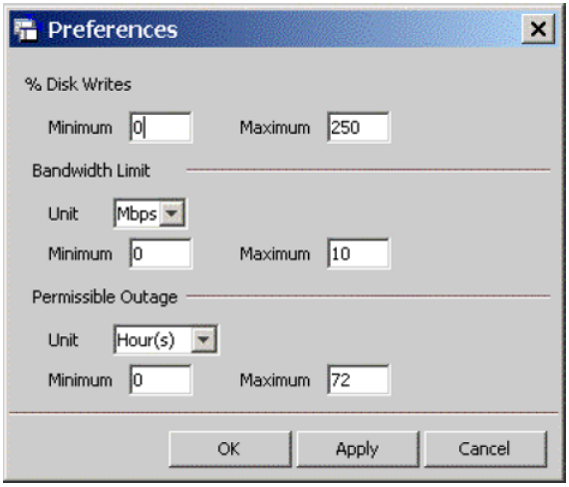
Available parameters for this option are % Disk Writes and RTO. The Have Bunker check box indicates whether the RVG has a bunker attached.

Changing the value ranges on the slider bar

Use the Preferences dialog box to change the value ranges for the slider bars.

To change the value ranges for the slider bars

1. Make sure the option for which you want to change the value ranges is selected in the left side of the What-if Analysis page.
2. Use the **File > Preferences** option to display the Preferences page.



   The Preferences dialog box displays parameters corresponding to the calculate option that you selected.

3. Change the values on the Preferences page as required:
   - Select the Unit for each option from the drop-down box.
   - Specify the appropriate values in the **Maximum** and **Minimum** fields. These values are used to indicate the range of values available on the slider bar.
4. Click **Ok**.

Recording and viewing the results

VRAdvisor records any values that you had specified in the analysis phase and the results of the What-if Analysis to a file, which uses the following naming convention:

`VRAdvResults_`*`Datestamp_and_Timestamp`*`.txt`

For Windows, the file is located in the `VERITAS/Volume Replicator Advisor/results` subdirectory. For Solaris, the file is located in the `/opt/VRTSvradv/results` subdirectory.

Every time you start the Analysis wizard, this file is automatically created and referenced later.

Chapter 19

Installing Volume Replicator Advisor (VRAdvisor)

This chapter includes the following topics:

- VRAdvisor System requirements
- Installing VRAdvisor on Solaris
- Installing VRAdvisor on Windows

VRAdvisor System requirements

VRAdvisor is supported on the following operating systems:

- Solaris 11
- Windows XP
- Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacenter Server

  Note: If Windows 2000 is installed on your system, you must have Service Pack (SP) 4 installed on it.

- Windows Server 2003 (32-bit): Standard Edition, Web Edition, Enterprise Edition, Datacenter Edition

Installing VRAdvisor on Solaris

To install VRAdvisor, run the following command:

```
# pkgadd -d storage_foundation/tools/vradvisor/pkgs VRTSvradv
```

where `VRTSvradv` is the package name.

Uninstalling VRAdvisor on Solaris

To uninstall `VRTSvradv`, run the following command:

```
# pkgrm VRTSvradv
```

Installing VRAdvisor on Windows

This section gives instructions on installing VRAdvisor on Windows. The procedure to install on Windows 2000 and Windows Server 2003 is the same, except that on a Windows 2000 system, a reboot is required after the installation completes.

Note: VRAdvisor is not installed as a part of the common installation process that uses the product installer. To install Volume Replicator Advisor, follow the procedure in this section.

Note: Although VRAdvisor is supported in a non-English locale, the wizards are still displayed in English.

To install VRAdvisor

1. If a previous version of VRAdvisor is installed, remove the existing VRAdvisor before installing VRAdvisor.
2. Navigate to the `windows` directory on the CD.
3. Run the `VRTSvradv.msi` from the `windows` directory.

   The installation wizard is launched. A message indicates that the `VRAdvisor` setup file is checking for the necessary parameters before starting the installation process.
4. In the Welcome page, click **Next**.
5. In the Customer Information page, enter your user name and organization, and click **Next**.

6. To install `VRAdvisor` in the default directory `c:\Program Files\VERITAS\Volume Replicator Advisor`, click **Next** and skip to 8.
7. To choose another location for installing `VRAdvisor`, click **Change** on the Destination Folder page.

   In the Change Current Destination Folder page, in the **Folder name** field, enter the complete path to the directory where you want the `VRAdvisor` package to be installed. You can also use the browse button to navigate to the required directory. Click **OK**.

   In the Destination Folder page, click **Next**.
8. In the Ready to Install the Program page, click **Install** to proceed with the installation.

   The Installing Volume Replicator Advisor page appears. This page displays a progress bar to indicate that the installation is in progress.

   After the installation completes, a message indicates that the installation was successful.
9. Click **Finish**.
10. If required, a message prompts you to restart the system. Click **Yes** to reboot the system now. Click **No** to reboot it later.

    On Windows 2000, installing VRAdvisor requires a reboot to activate disk performance counters for logical volumes. By default, Windows 2000 activates only the Physical Disk performance counters. To collect sample data, VRAdvisor requires the Logical Disk performance counters. The VRAdvisor installer runs the `diskperf -YV` command to activate the Logical Disk performance counters; however, the system must be restarted before the counters are enabled.

    On Windows 2003 or Windows XP, a reboot is not required to enable disk performance counters.

Uninstalling VRAdvisor on Windows

This section describes uninstalling VRAdvisor on Windows.

To uninstall VRAdvisor

1. Select **Settings > Control Panel** from the Windows **Start** menu.
2. Select **Add or Remove Programs**.
3. Select **Volume Replicator Advisor** from the list of programs.

4. Click **Remove**. Windows prompts you to confirm that you want to remove Volume Replicator Advisor.
5. Click **Yes**. The Volume Replicator Advisor dialog box appears.

   The progress bar on the Volume Replicator Advisor dialog box indicates that the removal is in progress.

   After the uninstallation procedure completes, the Add or Remove Programs dialog indicates that the Volume Replicator Advisor program has been removed successfully.

Section 5

VVR reference

- Appendix A. VVR command reference
- Appendix B. Using the In-band Control Messaging utility vxibc and the IBC programming API
- Appendix C. Volume Replicator object states
- Appendix D. Alternate methods for synchronizing the Secondary
- Appendix E. Migrating VVR from Internet Protocol version 4 to Internet Protocol version 6
- Appendix F. Sample main.cf files

Appendix A

VVR command reference

This appendix includes the following topics:

- VVR command reference

VVR command reference

Table A-1 lists the VVR commands and their descriptions.

The `vradmin` command can be issued from any host in the Replicated Data Set (RDS); the low-level VVR commands must be issued on the host on which the object resides.

Note: This reference lists command options for frequently used scenarios. For a complete list of options, refer to the respective manual page.

Table A-1 VVR command reference

VVR Command	Command Description
`vradmin -g` *`diskgroup`* `createpri` *`rvg_name dv01_name,dv02_name... srl_name`*	Creates Primary RVG of an RDS.
`vradmin -g` *`diskgroup`* `addsec` *`local_rvgname pri_hostname sec_hostname`*	Adds a Secondary RVG to an RDS.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vradmin -g diskgroup set local_rvgname sec_hostname synchronous=value`	Sets up mode of replication: `synchronous=off` sets asynchronous `synchronous=override` sets synchronous `vradmin` command does not allow you to set `synchronous=fail`. You can do this using the `vxedit` command. For more information on the `vxedit` command refer to the `vxedit` manual page.

`vradmin -g diskgroup set local_rvgname sec_hostname latencyprot=value`	Sets up Latency Protection:
	`latencyprot=fail`
	`latencyprot=override`
	`latencyprot=off`

`vradmin -g diskgroup set local_rvgname sec_hostname latency_high_mark=n`	Sets up `latency_high_mark:` `latency_high_mark=n`
`vradmin -g diskgroup set local_rvgname sec_hostname latency_low_mark=n`	Sets up `latency_low_mark:` `latency_low_mark=n`
`vradmin -g diskgroup set local_rvgname sec_hostname srlprot=value`	Sets up SRL Overflow Protection: `srlprot=autodcm (default) srlprot=dcm srlprot=override srlprot=fail srlprot=off`
`vradmin -g diskgroup set local_rvgname sec_hostname packet_size=n`	Sets up the packet size.

`vradmin -g diskgroup set local_rvgname sec_hostname protocol=value`	Sets the protocol.
	`protocol=TCP`
	`protocol=UDP`

`vradmin -g diskgroup set local_rvgname sec_hostname bandwidth_limit=value`	Sets the bandwidth limit for replication to the Secondary.
	`bandwidth_limit=value`
	`bandwidth_limit=none`

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description

`vradmin -g diskgroup set local_rvgname [sec_hostname] compression=value`	Enables or disables compression.
	`compression=on`
	`compression=off`

`vradmin -g diskgroup changeip local_rvgname [sec_hostname] attrs....`	Changes the host name or IP address of the Primary and Secondary RLINKs to the new values specified in the `newpri` and `newsec` attributes.
`vradmin -g diskgroup -l repstatus local_rvgname`	Displays consolidated replication-related information about an RDS.
`vradmin [-l] printrvg`	Displays information for all RDSs on local host.
`vradmin -g diskgroup [-l] printrvg local_rvgname`	Displays detailed information for a specific RDS.
`vradmin printvol`	Displays information about data volumes in all RDSs on the local host.
`vradmin -g diskgroup printvol local_rvgname`	Displays information about data volumes in an RDS.
`vradmin -g diskgroup pauserep local_rvgname sec_hostname`	Pauses replication to a Secondary.
`vradmin -g diskgroup resumerep local_rvgname sec_hostname`	Resumes replication to a Secondary.
`vradmin -g diskgroup -a startrep local_rvgname sec_hostname`	Starts replication and synchronizes the Secondary using autosync.
`vradmin -g diskgroup -c checkpt_name startrep local_rvgname sec_hostname`	Starts replication and synchronizes the Secondary using a Storage Checkpoint.
`vradmin -g diskgroup stoprep local_rvgname sec_hostname`	Stops replication to a Secondary.
`vradmin -g diskgroup -c checkpt_name syncrvg local_rvgname sec_hostname....`	Synchronizes the Secondary volumes with the corresponding Primary volumes based on differences when the application is active or inactive.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vradmin -g diskgroup -full -c checkpt_name syncrvg local_rvgname sec_hostname....`	Performs full synchronization with Storage Checkpoint to the Secondary when the application is active or inactive.
`vradmin -g diskgroup -full syncvol local_vols_list remote_hostname.... bandwidth_limit=value`	Synchronizes volumes on local host and remote hosts using full synchronization. The synchronization uses the specified bandwidth limit.
`vradmin -g diskgroup -verify syncrvg local_rvgname sec_hostname...`	Verifies and reports any data differences between Secondary volumes and the corresponding Primary volumes.
`vradmin -g diskgroup -verify syncvol local_vols_list remote_hostname... bandwidth_limit=value`	Verifies and reports any data differences between remote volumes and the corresponding local volumes. The operation uses the specified bandwidth limit.
`vradmin -g diskgroup [-k {cache|snap}] verifydata local_rvgname sec_hostname {cache=cacheobj | cachesize=size}`	Verifies that the data on the Secondary data volumes is identical to the Primary data volumes. The `-k` option cannot be used if you are using the `cachesize` option to create a cache object as these cache objects are destroyed automatically once the `vradmin verifydata` command executes successfully.
`vradmin -g diskgroup addvol local_rvgname volume_name`	Adds a volume to an RDS.
`vradmin -g diskgroup [-f] resizevol local_rvgname volume_name length [pridiskname=primary_disk_names] [secdiskname=secondary_disk_names]`	Resizes a data volume in an RDS. The `pridiskname` and `secdiskname` options enable you to specify a comma-separated list of disk names for the resize operation.
`vradmin -g diskgroup resizesrl local_rvgname length [pridiskname=primary_disk_names] [secdiskname=secondary_disk_names]`	Resizes the SRL in an RDS. The `pridiskname` and `secdiskname` options enable you to specify a comma-separated list of disk names for the resize operation.
`vradmin -g diskgroup delvol local_rvgname volume_name`	Removes a data volume from an RDS.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vradmin -g diskgroup ibc local_rvgname task_name [sec_host]......[-all]`	Performs the specified off-host processing task on the Secondary.
`vradmin -g diskgroup migrate local_rvgname newprimary_name`	Migrates the Primary role to Secondary *newprimary_name*.
`vradmin -g diskgroup takeover local_rvgname`	Takes over the Primary role from an original Primary with fast failback enabled.
`vradmin -g diskgroup -autofb takeover local_rvgname`	Takes over the Primary role from an original Primary with fast failback enabled and automatically synchronizes the original Primary when it becomes available.
`vradmin -g diskgroup -N takeover local_rvgname`	Changes the role from Secondary to Primary without enabling fast failback.
`vradmin -g diskgroup fbsync local_rvgname [cache=cache-object | cachesize=size]`	Converts the original Primary to a Secondary and starts resynchronization of the original Primary using fast-failback. Optionally, it also takes space-optimized snapshots of the original Primary's data volumes before starting the resynchronization.
`vradmin -g diskgroup -wait fbsync local_rvgname`	Converts the original Primary to a Secondary and starts resynchronization of the original Primary using fast failback. The command returns after resynchronization completes.
`vradmin -g diskgroup makesec local_rvgname newprimary_name`	Converts an original Primary to a Secondary when fast failback was not enabled.
`vradmin -g diskgroup resync local_rvgname [cache=cache-object | cachesize=size]`	Replays a DCM that is active due to an SRL overflow to incrementally synchronize the Secondary. Optionally, it also takes space-optimized snapshots of the original Primary's data volumes before starting the resynchronization.
`vradmin -g diskgroup -wait resync local_rvgname`	Replays a DCM that is active due to an SRL overflow to incrementally synchronize the Secondary. The command returns after resynchronization completes.
`vradmin -g diskgroup delsec local_rvgname sec_hostname`	Removes a Secondary from an RDS.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vradmin -g diskgroup delpri rvg_name`	Removes a Primary when the application is inactive.
`vradmin -g diskgroup -f delpri rvg_name`	Removes a Primary when the application is active.
`vradmin -g diskgroup activatebunker local_rvgname`	Activates a bunker. This command must be issued on the bunker host.
`vradmin -g diskgroup deactivatebunker local_rvgname`	Deactivates a bunker. This command must be issued on the bunker host.

`vradmin -g diskgroup -bdg bunkerdgname addbunker local_rvgname pri_hostname bunker_hostname protocol=value`	Creates a bunker RVG on the bunker host.
	`protocol=TCP`
	`protocol=UDP`
	`protocol=STORAGE`

`vradmin -g diskgroup delbunker local_rvgname bunker_hostname`	Deletes a bunker RVG from an RDS.
`vradmin -g diskgroup addvol local_rvgname volumeset_name`	Adds a volume set to an RDS.
`vradmin -g diskgroup -tovset volumeset_name addvol local_rvgname volume_name`	Adds a volume to a volume set that is associated with an RDS.
`vradmin -g diskgroup delvol local_rvgname volumeset_name`	Removes a volume set from an RDS.
`vradmin -g diskgroup -fromvset volumeset_name delvol local_rvgname volume_name`	Removes a volume from both the volume set and an RDS.
`vxrvg -g diskgroup [-l] getdatavols rvg_name`	Displays the names of all data volumes that are associated with the specified RVG.
`vxrvg -g diskgroup [-l] getrlinks rvg_name`	Displays the names of all RLINKs that are associated with the specified RVG.
`vxrvg -g diskgroup start rvg_name`	Enables I/O access to the data volumes associated with the RVG.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vxrvg -g diskgroup stop rvg_name`	Disables I/O access to the data volumes associated with the RVG.
`vxrvg -g diskgroup recover rvg_name`	Recovers the RVG after rebooting a node.
`vxrvg -g diskgroup -c checkpt_name checkstart rvg_name`	Marks the beginning of a Primary Storage Checkpoint by pointing to the current location of the SRL.
`vxrvg -g diskgroup checkend rvg_name`	Marks the end of a Primary Storage Checkpoint by pointing to the current location of the SRL.
`vxrvg -c checkpt_name checkdelete rvg_name`	Deletes the specified Primary Storage Checkpoint.
`vxrvg -g diskgroup cplist rvg_name`	Displays information about all existing Storage Checkpoints that are associated with the RVG.
`vxrvg -g diskgroup [-f] [-p] [-P prefix | -a] snapback rvg_name`	Reattaches snapshot volumes to their original volumes in the RVG. This operation is similar to the `vxassist snapback` command for traditional (third-mirror breakoff) snapshots, and the `vxsnap reattach` command for instant snapshots.
`vxrvg -g diskgroup snapprint rvg_name`	Displays information on the relationships between the data volumes of an RVG and any corresponding snapshots.
`vxrvg -g diskgroup [-P prefix] snaprefresh rvg_name`	Refreshes all existing snapshot volumes from the corresponding data volumes in the specified RVG. The `-P` option can be used to select the snapshot volume names that begin with the specified prefix for the refresh operation. The prefix "SNAP" is used if no prefix is specified. If a prefix is specified, this command refreshes only the snapshot volumes with that prefix. This creates a new point-in-time image for each snapshot, and the snapshot is immediately available for use with its new contents.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vxrvg -g diskgroup [-f] [-P prefix] snaprestore rvg_name`	Restores the contents of all of the data volumes in the specified RVG from the corresponding snapshot volumes. If a prefix is specified, this command restores the contents only from the snapshot volumes with that prefix. The volumes are immediately available with the restored contents.
`vxrvg -g diskgroup [-i interval] [-t timestamp_frequency] [-C count] stats rvg_name`	Displays the application statistics for the specified RVG. This is only valid on the Primary.
`vxrvg -g diskgroup [-P prefix] [-F|-S] snapshot rvg_name [instantfull=volume_list] [instantso=volume_list] [plexbreakoff=volume_list] [exclude=volume_list] [plexprefix=plex_prefix] [cache=cachename |cachesize=size] [syncing={yes|no}] [comment="comment"]`	Creates snapshots for all volumes in the specified RVG. This operation is similar to the `vxassist snapshot` command for traditional (third-mirror breakoff) snapshots, and the `vxsnap make` command for instant snapshots.
`vxrlink -g diskgroup assoc rvg_name rlink_name`	Associates an RLINK with an RVG.
`vxrlink -g diskgroup dis rlink_name`	Dissociates an RLINK from the RVG with which it is associated.
`vxrlink -g diskgroup [-a|-c checkpt_name]|-f] att rlink_name`	Enables an RLINK to connect to its remote RLINK by using auto attach, Storage Checkpoint attach or force attach.
`vxrlink -g diskgroup det rlink_name`	Detaches an RLINK.
`vxrlink -g diskgroup pause rlink_name`	Pauses updates to the Secondary RVG.
`vxrlink -g diskgroup resume rlink_name`	Resumes updates to the Secondary RVG that has been paused.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vxrlink -g diskgroup recover rlink_name`	Recovers the RLINK after rebooting a node.
`vxrlink -g diskgroup -c checkpt_name restore rlink_name`	Restores a failed Secondary RVG from a previously taken backup and Secondary Storage Checkpoint.
`vxrlink -c checkpt_name checkdelete rlink_name`	Deletes the specified Secondary Storage Checkpoint. **Note:** This command must be run only on the Primary.
`vxrlink -g diskgroup verify rlink_name`	Displays the configuration status of the given RLINK.
`vxrlink -g diskgroup [-i <interval>] [-e] stats rlink_name`	Gives details of the use of the network by VVR. The -e option displays extended statistics.
`vxrlink -g diskgroup [-i <interval>] [-T] status rlink_name`	Displays how much of the SRL is being used by the RLINK and how much the Secondary is behind. This incremental synchronization status is displayed after an interval of <*i*> seconds. This command output can also display the status with a timestamp, using the -T option.
`vxrlink -g diskgroup cplist rlink_name`	Displays information about the existing Secondary Storage Checkpoints associated with the RLINK, which includes the name of the Storage Checkpoint, its size, and the percentage of SRL used.
`vxrlink -g diskgroup [-T] updates rlink_name`	Valid on the Secondary only. Displays the update ID received by the Secondary, as well as the number of updates by which the Primary is ahead. This information can be used to determine the most up-to-date Secondary RVG. This command when used with the `-T` option, displays the exact time in hours by which the Secondary is behind.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vrstat -g diskgroup [-R] [-V] [-M] rvg_name`	Valid on the Primary as well as the Secondary. Displays detailed statistics. The output of this command is a combination of the outputs of the `vxrlink stats`, `vxrlink status`, `vxstat`, and `vxmemstat` commands.
`vxprint -V[l]`	Displays all RVGs.
`vxprint -P[l]`	Displays all RLINKS.
`vxmake -g diskgroup rlink rlink_name protocol=protocol_name remote_host=sec_hostname remote_rlink=rlink_name`	Creates an RLINK with the specified network transport protocol. The attribute `protocol_name` can have a value of `TCP` or `UDP`.

`vxmake -g diskgroup rlink rlink_name compression=value`	Enables or disables compression.
	`compression=on`
	`compression=off`

`vxmemstat [-i interval [-t count]] [-e]`	Display memory statistics for Veritas Volume Manager.
`vxtune [ -rH ] keyword arg ...`	Modify and display Volume Replicator and VxVM tunables.
`vrport [ -a | -r ] keyword arg ...`	Perform Volume Replicator port management operations.
`vrnotify -g diskgroup [-n number] [-t timeout] [rvg_name ...]`	Display Volume Replicator (VVR) events.
`vxedit -g diskgroup set protocol=protocol_name rlink_name`	Allows you to change the specified network transport protocol. The protocol can be set to either *TCP* or *UDP*. For more information refer to the `vxedit` manual page.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vxedit -g` *`diskgroup`* `set bunker_target=on` *`RLINK_from_secondary_to_bunker`*	Sets the `bunker_target` flag on the RLINK from the Secondary to the bunker node. You must run this command on the Secondary for global clustering to automatically replay the bunker to the Secondary. You only need to run this command when you are upgrading from Release 5.1 to 5.1 SP1.

`vxedit -g` *`diskgroup`* `set compression=`*`value rlink_name`*	Enables or disables compression.
	`compression=on`
	`compression=off`

In a shared disk group environment, commands must be issued from the CVM master. However, the RLINK informational and administrative commands, `vxrlink pause`, `vxrlink resume`, `vxrlink status`, and `vxrlink stats` can be issued on any node in the cluster.

Appendix B

Using the In-band Control Messaging utility vxibc and the IBC programming API

This appendix includes the following topics:

- About the IBC messaging utility vxibc
- In-band Control Messaging overview
- Using the IBC messaging command-line utility
- Examples—Off-host processing
- In-band Control Messaging API

About the IBC messaging utility vxibc

This appendix explains how to use the IBC Messaging command-line utility `vxibc` and the API for off-host processing. You can use the In-Band Control (IBC) Messaging feature with the FastResync (FMR) feature of Veritas Volume Manager (VxVM) and its integration with VVR to take application-consistent snapshots at the replicated volume group (RVG) level. This lets you perform off-host processing on the Secondary host. Typically, to perform off-host processing, you would use the `vradmin ibc` command to sequence and automate the operations.

See "Performing off-host processing tasks" on page 288.

However, if you want to customize the process beyond what can be achieved by using the `vradmin ibc` scripts, or if you want to program and integrate off-host

processing in your control facility, you need to use the `vxibc` command or the IBC API.

Off-host processing consists of performing operations on application data on a host other than the one where the application is running. Typical applications include Decision Support Systems (DSS), backup, and trial failover in VVR. In a VVR environment, off-host processing reduces the load on the application server, the Primary, and uses the potentially under-utilized Secondary.

You can use any of the snapshots (mirror break-off, full instant snapshot, or the space saving instant snapshot) to create a point in time (PIT) image of the RVG volumes on the Secondary site for off-host processing. You can use the IBC mechanism to make sure that the snapshots taken on the Secondary site are application-consistent. You can then refresh the snapshots or reattach them using fast resynchronization.

In-band Control Messaging overview

When you take a snapshot on the Secondary, it contains a point-in-time copy of the data on the Primary. Because the Secondary may be behind the Primary, it is not known exactly what time on the Primary this point-in-time copy represents.

VVR maintains a block-level consistency between the Primary and Secondary data volumes. But applications, for example, a file system, that use the data volumes have a higher-level consistency requirement. To support this, VVR provides the IBC facility.

IBC messaging enables you to send a message in the replication stream to notify the Secondary that an event has occurred on the Primary. In the case of a file system, you can use the `sync` command on the Primary, and then send an IBC message. When this message arrives on the Secondary, the data on the Secondary is consistent at the file system level and replication stops. Therefore, further updates are not applied to Secondary data volumes but are stored in the Secondary SRL. You then split off a mirror, which now contains a consistent image of the file system, and unfreeze replication. After the unfreeze operation all the pending updates that are stored in the secondary SRL are applied to the secondary data volumes.

The model with IBC Messaging is that a process on the Secondary waits for the IBC Message and a process on the Primary sends the message when the desired event has occurred.

Note: If you choose not to use IBC Messaging, the data on the Secondary is consistent and can be recovered by the application but it might be out-of-date and potentially out of sync.

Using the IBC messaging command-line utility

The `vxibc` command-line utility enables you to perform the following IBC Messaging tasks:

- See "Registering an application name" on page 479.
- See "Displaying the registered application name for an RVG" on page 479.
- See "Receiving an IBC message" on page 480.
- See "Sending an IBC message" on page 480.
- See "Unfreezing the Secondary RVG" on page 481.
- See "Unregistering an application name" on page 481.
- See "Receiving and processing an IBC message using a single command" on page 482.
- See "Sending and processing an IBC message using a single command" on page 482.

For details on using the `vxibc` command and the available options, see the online manual pages.

Registering an application name

Before being able to perform IBC operations on an RVG, you must register an application name for the RVG. The sender and the receivers of the IBC message must register the same application name. Multiple application names (up to a maximum of 32) can be registered for an RVG. Registration is not persistent through host reboots. Applications on rebooted hosts must be reregistered.

To register an application name for an RVG:

```
# vxibc [-g diskgroup] [-D deliver_timeout] \
  register application_name rvg_name
```

Displaying the registered application name for an RVG

You can use the `vxibc status` command to display the currently registered application names for a Replicated Volume Group (RVG).

To display the registered application names for an RVG: *# vxibc [-g diskgroup] status rvg_name*

Receiving an IBC message

You can use the `vxibc receive` command to receive the IBC message sent from the Primary to a Secondary.

To receive an IBC Message:

```
# vxibc [-g diskgroup] [-n | -R receive_timeout] [-f filename] \
   [-l buf_length] receive application_name rvg_name
```

Note that the *application_name* for the Secondary RVG must have been previously registered.

When the Secondary receives the IBC message, the state of the data volumes on the Secondary is the same as the state of the data volumes on the Primary at the time the IBC message was inserted in the replication stream. Subsequent writes are delivered to the Secondary and stored in the SRL, that is, replication is frozen. Secondary replication remains frozen until an `unfreeze` operation is performed, or the specified *freeze_timeout* expires. The default behavior for the receive operation is to block until an IBC message is received. The option `-n` makes the receive operation non-blocking, and returns if there is no message to receive. If the operation succeeds, the received message is displayed; if a file name is specified the message is written to that file.

Sending an IBC message

You can use the `vxibc send` command to send an IBC message from the Primary to a Secondary.

To send an IBC message:

```
# vxibc [-g diskgroup] [-N | -F freeze_timeout] \
  [-f filename | -m message] send application_name rvg_name \
  [rlink_name....]
```

Note that the *application_name* for the Primary RVG must be previously registered.

The IBC message is inserted into the update stream of the specified RLINKs. If an RLINK is not specified, the message is sent to all RLINKs currently attached to the Primary RVG.

IBC messages are always sent to the Secondary RVG irrespective of whether or not the `application_name` is registered on the Secondary.

If the application is registered on the Secondary, then the IBC message is discarded on the Secondary if a receive operation is not performed within the deliver-timeout period.

In the case the application is not registered at the Secondary, then the IBC message is held for the number of seconds specified in `deliver_timeout`. The default is 600 (10 minutes). If the `application_name` is registered within this time, then the IBC message is discarded if a receive operation is not performed within the deliver-timeout period. On the Secondary, the RVG remains frozen until an `unfreeze` operation is performed or the specified `freeze_timeout` expires.

Unfreezing the Secondary RVG

The `vxibc unfreeze` command unfreezes the Secondary RVG. This operation must be performed after receiving the IBC message using the `receive` operation.

To unfreeze an IBC message:

```
# vxibc [-g diskgroup] unfreeze application_name rvg_name
```

The `vxibc unfreeze` command permits replication to continue by allowing updates that were performed on the Primary data volumes after the `send` operation was executed on the Primary RLINK, to be applied to the Secondary RVG.

Unregistering an application name

The `vxibc unregister` command unregisters an application name for the RVG.

To unregister an application name:

```
# vxibc [-g diskgroup] unregister application_name rvg_name
```

The application name must have been previously registered for the RVG. Further `send` operations against the application name are not possible after unregistering on the Primary RVG.

You can unregister the application on the Secondary if the following conditions are met:

- If the IBC message has arrived on the Secondary and has been received by the user.
- If the IBC message has arrived on the Secondary and not received by the user, but the delivery timeout period has expired.

If you used the `vxibc regrecv` command, you do not have to unregister the application.

Receiving and processing an IBC message using a single command

The `vxibc regrecv` command enables you to specify a command to be performed on the arrival of the IBC together with the command arguments. The `vxibc regrecv` command performs the following operations in a single step:

- Registers the application name
- Receives the IBC message
- Runs the specified command with the provided arguments
- Unfreezes the Secondary RVG
- Unregisters the application name.

To receive and process an IBC message in a single step:

```
# vxibc [-g diskgroup] [-R receive_timeout] [-f filename] \
        [-l buf_length] regrecv application_name rvg_name command \
        [argument]
```

Sending and processing an IBC message using a single command

The `vxibc regsend` command performs the following operations in a single step:

- Registers the application name
- Sends the IBC message
- Unregisters the application name

The `vxibc regrecv` command must be started on the Secondary host before the IBC message sent from the Primary host gets invalidated due to a delivery timeout on the Secondary. This can also be done by first executing the `vxibc regsend` command on the Primary, followed by the `vxibc regrecv` command on the Secondary within the delivery time-out period which is by default 600 seconds. Otherwise, the IBC message is discarded on the Secondary because there is no corresponding registered application name.

To send and process an IBC message in a single step:

```
# vxibc [-g diskgroup] [-D deliver_timeout] \
      [-N | -F freeze_timeout] [-f filename | -m message] \
      regsend application_name rvg_name [rlink_name...]
```

The `vxibc regrecv` command must be issued before the IBC message delivery times out at the Secondary. Typically, this command is issued before the IBC is sent from the Primary.

Examples—Off-host processing

The examples in this chapter assume that the following VVR configuration has been set up on the Primary and Secondary hosts:

Name of the Primary host: `seattle`

`hrdg`	Disk group
`hr_rvg`	Primary RVG
`rlk_london_hr_rvg`	Primary RLINK for Secondary `london`
`hr_dv01`	Primary data volume #1
`hr_dv02`	Primary data volume #2
`hr_srl`	Primary SRL volume

Name of the Secondary host: `london`

`hrdg`	Disk group
`hr_rvg`	Secondary RVG
`rlk_seattle_hr_rvg`	Secondary RLINK for Primary `seattle`
`hr_dv01`	Secondary data volume #1
`hr_dv02`	Secondary data volume #2
`hr_srl`	Secondary SRL volume

The examples use the application name `dss_app` for sending and receiving IBC messages.

For example 1, example 2, and example 3, perform the following steps before you begin

1. Create a snapshot plex on the Secondary for each data volume using the command:

```
# vxassist -g hrdg snapstart hr_dv01
# vxassist -g hrdg snapstart hr_dv02
```

   You can use the `-b` option with the `vxassist snapstart` command to run the command in the background. Note that if you use the `-b` option of the `vxassist snapstart` command, you must wait for the snapshot plexes for all the data volumes in the RVG to be created and synchronized completely. When the plex synchronization completes, the output of the `vxprint` command displays the state of the new snapshot plex as `SNAPDONE`.

2. If you have bought a license for FastResync (FR) use the command:

```
# vxvol -g hrdg set fmr=on hr_dv01
# vxvol -g hrdg set fmr=on hr_dv02
```

Example 1—Decision support using the traditional snapshot feature and the vxibc utility

This example shows implementing decision support using the traditional snapshot feature and the `vxibc` utility.

To use the traditional snapshot feature and the vxibc utility for decision support

1. On the Secondary, register the application name `dss_app` and get ready to receive the IBC message. The command to break-off the snapshot plex when the IBC is received is specified with the `vxibc regrecv` command, as follows:

```
# vxibc -g hrdg regrecv dss_app hr_rvg vxrvg snapshot hr_rvg
```

2. On the Primary, put the application that is using the Primary data volumes `hr_dv01` and `hr_dv02` into a consistent state.

Note: Putting the application in a consistent state involves flushing all the buffers and pending transactions. For example, the file system can be brought into a consistent state by synchronizing the volumes using the VxFS specific sync command. In Oracle, the application can be brought into a consistent state by putting it in a hot-backup mode.

---

3 On the Primary, register the application name `dss_app` and send the IBC message using the following command:

```
# vxibc -g hrdg regsend dss_app hr_rvg
```

When the Secondary receives the IBC message, replication is frozen and no more data is written to the secondary data volumes.

The `vxrvg snapshot` command specified in step 1 then breaks-off the snapshot plexes from the Secondary data volumes, and replication starts again.

When the `vxibc` commands complete on both the hosts, the application name is unregistered.

4 On the Primary host, resume the application if it was suspended in step 2.

If the RLINK is asynchronous and behind there may be a delay between the `vxibc regsend` and `vxrvg snapshot` command. During this delay, the application is running.

5 On the Secondary, use the snapshot data volumes `SNAP-hr_dv01 and SNAP-hr_dv02` for running the DSS application, that is, for off-host processing.

6 When the application completes, reattach the snapshot plexes to the data volumes using the following command:

```
# vxrvg -g hrdg snapback hr_rvg
```

The reattach destroys the SNAP volumes and reattaches the snapshot plexes to their original volumes. If you have enabled FR on these volumes, only the blocks that were changed by the off-host processing application are resynchronized.

Example 2—Backing up using the snapshot feature and the vxibc utility

This example shows backing up using the snapshot feature and the `vxibc` utility.

To back up using the snapshot feature and the vxibc utility

1 Perform step 1 to step 4 of Example 1—Decision support using the traditional snapshot feature and the vxibc utility.

2 On the Secondary, copy the snapshot to tapes using a backup utility or the UNIX command `dd`. Use the `dd` command as follows:

```
# dd if=/dev/vx/rdsk/hrdg/SNAP-hr_dv01 of=/dev/rmt/0
# dd if=/dev/vx/rdsk/hrdg/SNAP-hr_dv02 of=/dev/rmt/0
```

3 Reattach the snapshot plexes to the original volumes:

```
# vxrvg -g hrdg snapback hr_rvg
```

Example 3—Trial failover using the snapshot feature

Because the goal is to simulate a crash on the Primary, do not use IBC Messaging for trial failover.

1 Pause the RLINK either on the Primary or the Secondary to maintain consistency.

To pause the RLINK on the Primary, type:

```
# vxrlink -g hrdg pause rlk_london_hr_rvg
```

To pause the RLINK on the Secondary, type:

```
# vxrlink -g hrdg pause rlk_seattle_hr_rvg
```

2 When the RLINK is paused, take snapshots of all the data volumes in the RVG:

```
# vxrvg -g hrdg -P trial snapshot hr_rvg
```

where `trial` is the prefix for the snapshot plexes for all data volumes. Snapshot data volumes with names `trial-hr_dv01` and `trial-hr_dv02` are created.

3 When the snapshots are complete, resume the RLINK by typing:

```
# vxrlink -g hrdg resume rlink_name
```

where *rlink_name* is the name of the paused RLINK.

4 Start the application using the data volumes `trial-hr_dv01` and `trial-hr_dv02` that you have snapped off.

5 Use the recovery function of the application to recover it, and then run the application. For example to recover a `vxfs` file system, use `fsck`.

```
# fsck -t vxfs /dev/vx/rdsk/hrdg/trial-hr_dv01
# fsck -t vxfs /dev/vx/rdsk/hrdg/trial-hr_dv02
```

6 When the test is complete, shut down the application. For a file system, unmount the file system.

7 Reattach the snapshot plexes to the original data volumes.

```
# vxrvg -g hrdg -P trial snapback hr_rvg
```

The `-P` option to the `vxrvg snapback` command reattaches to the original volume the plexes with the prefix specified when taking the snapshot.

Example 4—Decision support using the instant full snapshot feature and the vxibc utility

This example shows implementing decision support using the instant full snapshot feature and the `vxibc` utility.

To use the instant full snapshot feature and the vxibc utility for decision support

1 On the Secondary, prepare the volumes for which you want to create instant snapshots using the command:

```
# vxsnap -g hrdg prepare hr_dv01
# vxsnap -g hrdg prepare hr_dv02
```

This operation needs to be performed only for the first time you perform the snapshot operation.

2 On the Secondary, create snapshot volumes of the same size as the original volumes and with an appropriate prefix:

```
# vxassist -g hrdg make dss-hr_dv01 volume_length
# vxassist -g hrdg make dss-hr_dv02 volume_length
```

where *volume_length* is the length of the original volumes.

3 On the Secondary, prepare the snapshot volumes for which you want to create instant snapshots using the command:

```
# vxsnap -g hrdg prepare dss-hr_dv01
# vxsnap -g hrdg prepare dss-hr_dv02
```

4 On the Secondary, issue the following command:

```
# vxibc -g hrdg regrecv dss_app hr_rvg [vxrvg -g hrdg -F -P dss \
  snapshot hr_rvg]
```

The command `vxrvg -g hrdg -F -P dss snapshot hr_rvg` is run when the IBC message arrives on the Secondary and the command creates an instant full snapshot.

5 On the Primary, put the application that is using the Primary data volumes `hr_dv01` and `hr_dv02` into consistent state using the application specific method.

For information on the consistent state, see step 2.

6 On the Primary, register the application name `dss_app` and send the IBC message using the following command:

```
# vxibc -g hrdg regsend dss_app hr_rvg
```

7 On the Primary host, resume the application if it was suspended in step 5.

If the RLINK is asynchronous and behind there may be a delay between the `vxibc regsend` and `vxrvg snapshot` command. During this delay, the application is running.

8 On the Secondary, use the snapshot data volumes `dss-hr_dv01 and dss-hr_dv02` for running the DSS application, that is, for off-host processing.

9 When the application completes, reattach the snapshot plexes to the data volumes using the following command:

```
# vxrvg -g hrdg snapback hr_rvg
```

The reattach destroys the `dss` volumes and reattaches the snapshot plexes to their original volumes.

In-band Control Messaging API

This section explains how to use the In-Band Control (IBC) Messaging Application Programming Interface (API). VVR supports a special set of ioctls for accessing the IBC messaging facility. These ioctl commands allow an application to register with the facility, send and receive IBC messages, and unregister from the facility.

The IBC facility enables applications to insert application-defined control messages inband with the Primary RVG update stream being replicated to a Secondary RVG. When an IBC message arrives at the Secondary RVG, replication is frozen until directed to unfreeze by a companion application residing on the Secondary host. In this way, an application can signal a Secondary RVG that some user-defined

event has occurred relative to the update stream, such as a point of application-level consistency, and enable the Secondary RVG to take some action while replication is frozen.

VVR provides the following ioctl commands:

- RV_IBC_REGISTER
- RV_IBC_SEND
- RV_IBC_RECEIVE
- RV_IBC_UNFREEZE
- RV_IBC_UNREGISTER
- RV_IBC_STATUS

IOCTL commands

This section describes the IOCTL commands supported.

Note: The RVG must be started for the IOCTLs to succeed.

RVG devices support five special ioctls: `RV_IBC_REGISTER`, `RV_IBC_UNREGISTER`, `RV_IBC_SEND`, `RV_IBC_RECEIVE`, and `RV_IBC_UNFREEZE`. The format for calling each ioctl command is:

```
#include <fcntl.h>
 #include <errno.h>
 #include <sys/types.h>
 #include <vxvm/voldefs.h>
 #include <vxvm/volioctl.h>
 #include <vxvm/volibc.h>
 #include <vxvm/volmachdep.h>

int ioctl(int fd, int cmd, void *arg);
```

Include the path `/usr/include` to build your program.

The argument `fd` is the file descriptor obtained by opening the RVG device using the `open` (2) system call.

The value of `cmd` is the ioctl command code, and `arg` is a pointer to a structure containing the arguments to be passed to the kernel. Definitions of the argument structures for each ioctl are described below.

The return value for all ioctls is `0` if the command was successful, and `-1` if it was rejected. If the return value is `-1`, then `errno` is set to indicate the cause of the error.

RV_IBC_REGISTER

This ioctl registers an application name for the RVG and returns a key. Only registered application names, using the key, may use the IBC messaging facility on a particular RVG. Multiple application names can be registered for any RVG, up to a maximum of 32.

The ioctl argument structure for the `RV_IBC_REGISTER` command is:

```
    struct ibc_register_args {
        char            application_name[NAME_SZ];
        int             deliver_timeout;
        ibc_appid_t     application_id;
};
```

Argument `deliver_timeout` specifies a time-out value in seconds for delivery of an IBC message after it has arrived at the Secondary RVG. When the time-out expires, the Secondary RVG discards the IBC message and continues replication. See `RV_IBC_SEND` and `RV_IBC_RECEIVE` for definition of message delivery. A `deliver_timeout` of `0` is used to specify no time-out.

Argument `application_id` is returned by the ioctl. It must be supplied as input argument to all other IBC ioctls. The `NAME_SZ` value is 32.

Use of IBC messages is inherently distributed. A copy or agent of the application is expected to be resident on each participating host, and each participating application must register on its own host. Those resident on the Secondary host must register using an application name identical to the name registered on the Primary host. The returned *application_id* has a local scope; it can be distributed to any cooperating applications on the same host, but it cannot be used successfully by an application on a remote host.

An IBC message received on the Secondary for an application name that is not registered is discarded after delivery timeout. Registration is not persistent across system reboots. Applications must be registered again after the host reboots. After the Secondary is rebooted, the application must be registered within ten minutes after `vxnetd` is started if an IBC message has already arrived.

The `vxnetd` command is started from the system startup script:

```
/etc/init.d/vxnm-vxnetd
```

On failure, the `errno` value can be one of many possible error codes, including generic operating system error codes. However, the only IBC-specific error codes are the following:

`EIBC_NOMEM`	Maximum number of applications (32) already registered.

EIBC_DUP_APPLICATION *application_name* is already registered.

RV_IBC_SEND

This ioctl can only be issued against the Primary RVG with a valid key obtained from the RV_IBC_REGISTER ioctl. The ioctl inserts an IBC message into the data update stream of one or all RLINKs attached to the RVG.

If it is desired that an IBC message be inserted at an exact location in the update stream, such as a point of application-level consistency, then there must be no concurrent write activity to the RVG when the RV_IBC_SEND ioctl is issued. Note that writes made to the block device interface of a data volume may be cached, so a disk sync must be done before issuing the ioctl. If there are active writes to the RVG when the ioctl is issued, the insertion point of the IBC message in the RLINK update data stream is arbitrary in relation to that activity.

The ioctl returns using the same semantics as a data write to the RVG; it returns when the IBC message has been committed to the SRL and has also been transferred to all synchronous-mode replicas attached to the RVG.

The ioctl argument structure for the RV_IBC_SEND command is:

```
    struct ibc_send_args {          /* IOCTL_STRUCT */
        vx_u32_t        ibc_magic;
        vx_u32_t        ibc_version;
        ibc_appid_t     application_id;
        char            replica[NAME_SZ];
        int             flags;
        int             freeze_timeout;
        caddr_t         msg_buf;
        int             msg_len;
};
```

Argument *ibc_magic* is used to verify whether the ioctl structure is a valid 4.0 structure. It should be set to NM_IBC_MAGIC.

Argument *ibc_version* specifies the current IBC version. It should be set to NM_IBC_VERSION.

Argument *application_id* is the key returned by the RV_IBC_REGISTER ioctl. A registration must be done before the RV_IBC_SEND ioctl can be used.

Argument *replica* specifies the name of the RLINK to which the IBC message is to be send. The null string specifies a broadcast to all RLINKs currently attached to the Primary RVG.

Argument *flags* set to IBC_FREEZE causes the secondary replication to freeze for the time-out period specified in freeze_timeout. If replication is not desired to be frozen, then *flags* should be set to 0.

Argument *freeze_timeout* specifies a time-out value in seconds between delivery of an IBC message on the Secondary and execution of an RV_IBC_UNFREEZE ioctl against the Secondary RVG. When the time-out expires, replication at the Secondary continues. A time-out value of zero is used to specify no time-out.

Argument *msg_buf* is a pointer to a buffer containing an IBC message. The content of an IBC message is user-defined and has no restriction except size.

Argument *msg_len* is the length, in bytes, of the IBC message and can be no greater than 128k bytes.

On failure, possible values returned in errno are:

EIBC_NO_RLINK	No *RLINK* or specified *RLINK* exists
EIO I/O	I/O error while logging the IBC message
EIBC_MSG_LENGTH	Message is greater than maximum allowable length (128K)

RV_IBC_RECEIVE

This ioctl can only be issued against a Secondary RVG with a valid key obtained from the RV_IBC_REGISTER ioctl. The ioctl receives an IBC message sent from the Primary RVG. At the time of receipt, Secondary replication is frozen. The state of the data volumes on the Secondary is the same as that on the Primary at the time the IBC message was sent. Secondary replication remains frozen until an RV_IBC_UNFREEZE ioctl is issued against the Secondary RVG, or the *freeze_timeout* specified when the IBC message was sent expires, or the *deliver_timeout* specified when the application name was registered for the Primary RVG expires and the receive operation has not been performed.

The ioctl argument structure for the RV_IBC_RECEIVE command is:

```
    struct ibc_receive_args {
        ibc_appid_t     application_id;
        int             flags;
        ibc_timeout_t   timeout;
        int             drop_count;
        caddr_t         msg_buf;
        size_t          buf_len;
        size_t          msg_len;
};
```

Argument *application_id* is the key returned by the RV_IBC_REGISTER ioctl. A registration must be done before the RV_IBC_RECEIVE ioctl can be used.

Argument *flags* may specify IBC_BLOCK. If this flag is set, the ioctl will block until an IBC message is available to receive. If IBC_BLOCK is not set, the ioctl returns with an error if no IBC message is available.

Argument *timeout* specifies a time-out value in seconds to block waiting for an IBC message if flag IBC_BLOCK is set. When the time-out has expired, the ioctl returns with an error. A time-out of zero is used to specify no time-out. If you set IBC_FREEZE flag, indicates that a freeze timeout is enforced.

Value *drop_count* is returned by the ioctl. This value contains the number of messages that have been dropped due to delivery time-outs. If *drop_count* is non-zero, no message is returned and the ioctl returns an error.

Argument *msg_buf* is a pointer to a buffer to receive the IBC message.

Argument *buf_len* is the length, in bytes, of the *msg_buf*.

Value *msg_len* is returned by the ioctl and specifies the length in bytes of the IBC message. The maximum IBC message length is 128K bytes. If *msg_len* is greater than *buf_len*, the IBC message is truncated to *buf_len* bytes and no error is indicated.

On failure, possible values returned in errno are:

EIBC_NO_APPLICATION	Argument *application_id* is not valid.
ENOMSG	IBC messages have been dropped due to delivery time-out, or if no IBC message was available.

RV_IBC_UNFREEZE

This ioctl can only be issued against a Secondary RVG with a valid key obtained from the RV_IBC_REGISTER ioctl. The ioctl unfreezes replication of the Secondary RVG; that is, it resumes updates of the Secondary volumes.

The ioctl argument structure for the RV_IBC_UNFREEZE command is:

```
    struct ibc_unfreeze_args {
        ibc_appid_t     application_id;
};
```

Argument application_id is the key returned by the RV_IBC_REGISTER ioctl. A registration must be done before the RV_IBC_UNFREEZE ioctl can be used.

On failure, the possible values returned in errno are:

`EIBC_NO_APPLICATION`	Argument *application_id* is not valid.
`EBUSY`	There is currently active use of the IBC messaging facility by one or more ioctls using this *application_id*

RV_IBC_UNREGISTER

This ioctl unregisters an application name. This ioctl returns an error if any ioctl is active for the RVG using the registered key.

For a Primary RVG, no `RV_IBC_SEND` ioctls will be accepted with the registered key after unregistering. Those IBC messages already introduced into the update stream are not affected by a subsequent unregister, even if they have not yet been sent to the Secondary RVG.

For a Secondary RVG, `RV_IBC_RECEIVE` or `RV_IBC_UNFREEZE` ioctls using the registered key cannot be successfully executed after the unregister, and any IBC messages arriving for the registered name are discarded.

The ioctl argument structure for the `RV_IBC_UNREGISTER` command is:

```
    struct ibc_unregister_args {
        ibc_appid_t     application_id;
};
```

Argument *`application_id`* is the key returned by the `RV_IBC_REGISTER` ioctl. A registration must be done before the `RV_IBC_UNREGISTER` ioctl can be used.

On failure, possible values returned in `errno` are:

`EIBC_NO_APPLICATION`	Application is not registered.
`EIBC_IBC_PENDING`	IBC deliver or unfreeze pending.

RV_IBC_STATUS

This ioctl returns all the currently registered application names for the RVG. You can register up to 32 application names can be registered in RVG.

The ioctl argument structure for the `RV_IBC_STATUS` command is:

```
struct ibc_status_args {
        int             napps;
        caddr_t         stat_buf;
        int             buf_len;
}
```

Argument *napps* is the number of applications registered.

Argument *stat_buf* returns the application name and the application ID. This is a preallocated buffer with the following structure:

```
struct ibc_status       {
        char            application_name[NAME_SZ];
        ibc_appid_t     application_id;
};
```

Argument *buf_len* specifies the times of the *ibc_status* buffer, which determines the number of application names it returns. You can register to 32 application names per RVG.

If no application names are registered in the RVG, no error is returned. In this case, the argument *napps* be 0.

Using the IBC API

The ioctl command set is intended to be used by a set of daemons, one on the RVG Primary host and one on each Secondary host that is to participate in IBC message retrieval. Each must register under an identical application name and be registered before IBC message generation begins. Because registration does not survive host crashes, but IBC messages once sent do persist beyond host crashes, it is suggested that the Secondary daemons be spawned as a part of system startup.

IBC messages use at-least-once delivery semantics. Retrieval daemons must be tolerant of receiving the same IBC message more than once. It is however guaranteed any duplicate copies of a messages will be delivered before the next new message is delivered.

Appendix C

Volume Replicator object states

This appendix includes the following topics:

- Volume Replicator Kernel State
- Volume Replicator utility states

Volume Replicator Kernel State

The Kernel State (KSTATE) indicates the accessibility of the RVG and RLINK objects. When you issue the `vxprint` command, the KSTATE is listed under the KSTATE heading of the output.

Tip: In most cases, if the KSTATE/STATE is enabled/active, the object is available.

RVG KSTATEs

Table C-1 lists the RVG KSTATEs

Table C-1 RVG KSTATEs

RVG KSTATE	Description
ENABLED	You can do I/O to the volumes that belong to the RVG.
DISABLED	You cannot do I/O to the volumes that belong to the RVG. It must be enabled before you can use the RVG. Issue the `vxrvg start` command.

Table C-1 RVG KSTATEs *(continued)*

RVG KSTATE	Description
RECOVER	You cannot do I/O to the volumes that belong to the RVG. This state is triggered after a disk group import or if the RVG object was not recovered properly after a reboot or a crash. Issue the `vxrvg recover` command.

RLINK KSTATEs

Table C-2 lists the RLINK KSTATEs

Table C-2 RLINK KSTATEs

RLINK KSTATE	Description
CONNECT	Replication is taking place.
ENABLED	The RLINK is not communicating with its peer; therefore, replication is not taking place. When the RLINK does communicates with its peer, it automatically reverts to the CONNECT state.
DETACHED	The RLINK is not replicating and is not attempting to connect. Issue the `vxrlink att` command.
RECOVER	The RLINK is out of operation. This state is triggered after a disk group import or if the RLINK object was not recovered properly after a reboot or a crash. Issue the `vxrlink recover` command.

Volume Replicator utility states

This section describes the utility state (STATE) of the RVG and RLINK objects.

Tip: In most cases, if the KSTATE/STATE is ENABLED/ACTIVE, the object is available.

RVG utility states

This section lists the RVG states and their descriptions.

- EMPTY—State of a newly created RVG. Issue the `vxrvg start` command to start the RVG.
- CLEAN—The RVG is stopped. This state is seen after issuing the `vxrvg stop` command. Issue the `vxrvg start` command to start the RVG.

- ACTIVE—This determines if I/O can be performed to the data volumes:
  - If the KSTATE is ENABLED, I/O can be performed.
  - If the KSTATE is RECOVER, I/O cannot be performed (this state usually occurs after a system crash).
  - If the KSTATE is DISABLED, I/O cannot be performed.
- FAIL—A data volume error occurred.

RLINK utility states

This section lists the RLINK states and their descriptions.

- UNASSOC—Not associated to an RVG.
- STALE—Associated to an RVG but needs complete synchronization between the Primary and Secondary.
- ACTIVE—Replicating or ready to replicate.
- PAUSE—Replication is not active because of an administrative action or a configuration error.
- FAIL—Data volume error occurred on the Secondary or the `vxrlink -w` pause command was issued on the Secondary.
  See "Inconsistent RLINKs" on page 500.
- PAUSING—Temporary state while `vxrlink pause` is executing.
- RESUMING—Temporary state while `vxrlink resume` is executing.
- restoring—Temporary state while `vxrlink restore` is executing.

Inactive RLINKs

An RLINK is considered inactive whenever the Primary cannot send data to the Secondary for any reason, such as:

- A temporary failure of the network connection between the Primary and the Secondary.
- A failure of the Secondary node.
- The execution of a `vxrlink pause` command by an administrator.

The data to be sent on an inactive RLINK is buffered in the SRL. If the RLINK remains inactive for an extended period, the SRL may not be able to buffer new writes; even if it can, the Secondary becomes increasingly out-of-date. So it is important that the SRL is large enough to accommodate most of the inactive periods.

To help control the behavior of an inactive RLINK, SRL overflow protection may be used.

See "Setting the SRL overflow protection for a Secondary" on page 148.

STALE RLINK state

An RLINK is STALE when the Secondary data volumes do not contain the Primary's data and cannot be brought up-to-date using the SRL. When an RLINK is first created, its initial state is STALE.

RLINKs can enter the `STALE` state when they are detached either manually (via `vxrlink det`) or by the kernel (on the Primary only, if an SRL media error occurs). An RLINK can also become STALE if the log overflows. You can prevent the log from overflowing by setting SRL protection.

See "About the srlprot attribute" on page 91.

You can change the state of an RLINK from stale to active, using the automatic synchronization feature, using full synchronization with Storage Checkpoint, or synchronizing the Secondary using block-level backup.

See "Using the automatic synchronization feature" on page 154.

See "Using the full synchronization feature" on page 502.

See "Example—Synchronizing the Secondary using block-level backup" on page 506.

FAIL RLINK state

A Primary RLINK enters the fail state when the corresponding Secondary RLINK enters the `FAIL` state for any reason. This happens if there is an unrecoverable I/O error on one of the Secondary data volumes.

There are two ways for a Secondary RLINK to fail. One is if it encounters an I/O error that cannot be corrected. This is less likely to happen if the data volumes have been configured in a redundant fashion.

The second way for a Secondary RLINK to fail is if you enter the `vxrlink -w pause` command on the Secondary. Use this command carefully because the data volumes on the Secondary become writable. The command must be used if a data volume must be restored from backup.

When the restore operation is complete, execute the following command:

```
# vxrlink -g diskgroup -c checkpoint_name restore rlink_name
```

This will return both the Primary and Secondary RLINKs to the ACTIVE state.

Secondaries can also be restored from a Primary Storage Checkpoint if a Secondary Storage Checkpoint is not available, but the Primary Storage Checkpoint and corresponding backup are available.

If the Secondary RLINK cannot be restored, or if it is no longer needed, then `vxrlink det` can be used on either the Primary or the Secondary to detach the Primary RLINK and make it STALE.

Note: In some cases after an RLINK has been moved from the FAIL state back to the ACTIVE state, the RVG may remain in the FAIL state. This can be corrected by entering: `# vxrvg -g diskgroup start rvg_name`

Inconsistent RLINKs

When an RLINK is inconsistent, the Secondary cannot be used for a failover because the data volumes do not reflect the data on the Primary node.

Note that `inconsistent` is not a state. Whether an RLINK is consistent, or not, is shown in the flags field of the RLINK.

To see whether the `consistent` or `inconsistent` flag is set, use the following command:

```
# vxprint -g diskgroup -l rlink_name
```

An RLINK that is `inconsistent` and in the FAIL state must be restored before it can be used again. It can be restored using a Primary or a Secondary Storage Checkpoint. An RLINK becomes `inconsistent` and gets in the FAIL state in situations such as:

- When you enter the `vxrlink -w pause` command
  Used to put an RLINK in the fail state. Not normally used.
- If there is an unrecoverable I/O error on a data volume on the Secondary
  If the data volume can be restored from backup, it is possible to recover. Loss of volumes due to I/O errors is usually preventable by mirroring.

If an RLINK is `inconsistent`, but not in the fail state, it could be a temporary situation and the inconsistent flag will clear when the operation completes. This happens in situations, such as:

- During atomic update
  An atomic update operation would happen automatically, for example, to catch up after a network outage. If a machine crashed during such an update, the user would see the inconsistent flag set while not in the FAIL state. This is unlikely, however, and as long as the Primary node has not been lost, VVR will

automatically make the RLINK consistent again once the Primary-Secondary network connection is reestablished.

- During DCM resynchronization
  When you execute the `vxrvg resync` command after the SRL has overflowed, the RLINK becomes inconsistent during resynchronization until the DCM replay is complete.

When the `inconsistent` flag is set, a flag is displayed indicating whether the RLINK can be resynchronized. If the RLINK has the `cant_sync` flag set, it is inconsistent, and this Secondary needs to be resynchronized before it can take part in replication again. If the `inconsistent` and `can_sync` flags are set, there is enough information to make it consistent again. This will occur automatically.

Pausing, resuming, and restoring RLINK states

PAUSING, RESUMING, and RESTORING are temporary states through which the RLINK transitions when doing a `Pause`, `Resume`, or `Restore`, respectively. If these states persist, it means that the command failed halfway through the execution. Recovery from these states is simple.

If the state is PAUSING, it means that some error prevented the pause operation from completing. The error is displayed during execution of the `vxrlink pause` command. When the error is corrected, the next `vxrlink pause` command will succeed.

If the state is RESUMING, it means that some error prevented the resume operation from completing. The error is displayed during execution of the `vxrlink resume` command. When the error is corrected, the next `vxrlink resume` command will succeed.

If the state is restoring, a `vxrlink restore` command failed. You must execute either a `vxrlink -w pause` command to put the RLINK back into the FAIL state, or a `vxrlink -c` *`checkpoint`* `restore` command to put it into the ACTIVE state.

Two other Volume Replicator commands also use two-phase transactions. If these commands fail after executing partially, they can be safely repeated. The commands are:

- `vxrlink recover`
- `vxrvg recover`

If the `vxrlink recover` or `vxrvg recover` command fails, the state of the object will still be RECOVER. Volume Replicator commands that do two-phase transactions report an error message and have a nonzero exit code if they fail.

Appendix D

Alternate methods for synchronizing the Secondary

This appendix includes the following topics:

- Using the full synchronization feature
- Using block-level backup and Storage Checkpoint
- Using the Disk Group Split and Join feature
- Using difference-based synchronization
- Examples for setting up a simple Volume Replicator configuration

Using the full synchronization feature

This section explains how to use the Full Synchronization feature of VVR to synchronize the Secondary completely and start replication. Full synchronization compresses zeroes while processing the data and hence proves beneficial when a large part of the Primary data volumes contain zeroes. However, we recommend that you use Automatic Synchronization to synchronize the Secondary because its performance is better than Full Synchronization. Automatic Synchronization also handles network outages efficiently and continues even after the system reboots.

Full synchronization synchronizes the Secondary over the network when the Primary data volumes contain data and when the application is active or inactive. After the Primary and Secondary are synchronized, replication must be started.

By default, the `vradmin syncrvg` command synchronizes Secondary data volumes using difference-based synchronization. To perform a full synchronization, specify the `-full` option.

We recommend that you always use the `-c` option with the `vradmin syncrvg` command to synchronize the Secondary using full synchronization. The `-c` *`checkpoint`* option starts a Storage Checkpoint, synchronizes the data volumes, and ends the Storage Checkpoint after the synchronization completes. After the `vradmin syncrvg` command completes, use this Storage Checkpoint with the `vradmin startrep` command to start replication. To delete the Primary Storage Checkpoints, use the `vxrvg checkdelete` command.

The SRL must be large enough to hold the incoming updates to the Primary data volumes while the synchronization is in progress. The SRL might fill up and the Storage Checkpoint might overflow if the number of writes to the Primary data volumes is high when the Secondary is being synchronized.

A Storage Checkpoint that has overflowed becomes invalid and cannot be used to start replication. If the Storage Checkpoint overflows while synchronization is in progress, the `vradmin syncrvg` command must be issued again.

The `vradmin syncrvg` command can be used to synchronize multiple Secondaries at the same time. The `vradmin syncrvg` command displays the progress of the synchronization.

The `vradmin syncrvg` command synchronizes the volumes in an RVG. If a volume set is associated to an RVG, synchronizing the RVG only affects the component volumes of the volume set that are associated with the RVG. If the volume set includes component volumes that are not associated to the RVG, those volumes are not synchronized.

See “About SmartMove for VVR” on page 157.

To synchronize the Secondary RVG with the Primary RVG using full synchronization with Storage Checkpoint

1. Verify that the RLINKs are detached to ensure that replication is stopped.
2. To synchronize the Secondary RVG, issue the following command:

```
# vradmin -g diskgroup -full -c checkpt_name syncrvg \
    local_rvgname sec_hostname....
```

   Note that you can use the `-c` option with the `vradmin syncrvg` command when performing full synchronization, to automatically start a Storage Checkpoint with the specified name. After the data volumes are synchronized, the Storage Checkpoint is ended. This Storage Checkpoint can then be used to start replication using the `vradmin startrep` command.

   The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

   The argument *`sec_hostname...`* is a space-separated list of the names of the Secondary hosts as displayed in the output of the `vradmin printrvg` command.

   The argument *`checkpt_name`* specifies the name of the Primary Storage Checkpoint of your choice.

3. After the synchronization completes, start replication to the Secondary with the Storage Checkpoint:

```
# vradmin -g diskgroup -c checkpt_name startrep \
 local_rvgname sec_hostname
```

   After the RLINKs are attached, the Secondary remains inconsistent until it has received all of the accumulated updates up to the checkend. While the Secondary is inconsistent, the `inconsistent` flag is set on the Secondary RLINK. After all updates up to the checkend have been received and applied at the Secondary, the `inconsistent` flag is cleared.

   Use `vxrlink status` to view the status of synchronization.

   See "Displaying the status of a Secondary" on page 174.

Example—Synchronizing the Secondary using full synchronization with Storage Checkpoint

This example explains how to synchronize the Secondary RVG `hr_rvg` on the Secondary host `london` with the Primary RVG on host `seattle`.

To synchronize the Secondary RVG `hr_rvg` on `london` with its Primary RVG on `seattle` using full synchronization

1. Verify that the RLINKs are detached to ensure that replication is stopped.
2. Issue the following command from any host in the RDS:

   ```
   # vradmin -g hrdg -full -c checkpt_presync syncrvg hr_rvg \
       london
   ```

   Note that you can use the `-c` option with the `vradmin syncrvg` command when performing full synchronization, to automatically start a Storage Checkpoint with the specified name. After the data volumes are synchronized, the Storage Checkpoint is ended. This Storage Checkpoint can then be used to start replication using the `vradmin startrep` command.

   The name *`checkpt_presync`* is the Primary Storage Checkpoint that you will create.
3. After the synchronization is complete, issue the following command to start replication using the Storage Checkpoint:

   ```
   # vradmin -g hrdg -c checkpt_presync startrep hr_rvg london
   ```

Using block-level backup and Storage Checkpoint

This method is useful for low bandwidth networks or very large data sets. You can use the block-level backup and Storage Checkpoint method to synchronize the Secondary when a backup of the data is available and a Storage Checkpoint has been started on the Primary. You do not have to use the network to transfer the data. This method does have a risk of SRL overflow.

Make sure that the SRL is large enough to contain all the writes made by the application while synchronization is in progress. If necessary, you can resize the SRL.

See "Changing the size of the SRL on the Primary and the Secondary" on page 223.

Caution: During the process the Storage Checkpoint will overflow if the SRL fills up. To determine if the Storage Checkpoint has overflowed, issue the *`vxrvg cplist rvg_name`* command on the Primary to display the list of valid Storage Checkpoints.

See "Example—Synchronizing the Secondary using block-level backup" on page 506.

To synchronize the Secondary using backup and Primary Storage Checkpoint

1. Start a Primary Storage Checkpoint using the `vxrvg checkstart` command:

```
# vxrvg -g diskgroup -c checkpt_name checkstart  \
     local_rvgname
```

2. Perform a block-level backup of the data volumes in the Primary RVG.
3. End the Storage Checkpoint in the SRL when the backup is complete by using the `vxrvg checkend` command:

```
# vxrvg -g diskgroup checkend local_rvgname
```

4. Restore the backup to the Secondary data volumes.
5. Start replication using the Storage Checkpoint after the restore on the Secondary completes:

```
# vradmin -g diskgroup -c checkpt_name startrep \
 local_rvgname sec_hostname
```

After the RLINKs are attached, the Secondary remains inconsistent until it has received all of the accumulated updates up to the checkend. While the Secondary is inconsistent, the `inconsistent` flag is set on the Secondary RLINK. After all updates up to the checkend have been received and applied at the Secondary, the `inconsistent` flag is cleared.

Example—Synchronizing the Secondary using block-level backup

This example explains how to synchronize the Secondary RVG `hr_rvg` on the Secondary host `london` with the Primary RVG on host `seattle` using block-level backup and Storage Checkpoint.

To synchronize the Secondary using block-level backup and Storage Checkpoint

1. Start a Primary Storage Checkpoint on `seattle`:

```
# vxrvg -g hrdg -c checkpt_presync checkstart hr_rvg
```

2. Perform a block-level backup of the data volumes in the Primary RVG.
3. End the Primary Storage Checkpoint when the backup is complete:

```
# vxrvg -g hrdg checkend hr_rvg
```

4 Restore the backup to the Secondary data volumes.

5 Start replication using the Storage Checkpoint after the restore is complete:

```
# vradmin -g hrdg -c checkpt_presync startrep hr_rvg london
```

Using the Disk Group Split and Join feature

The Disk Group Split and Join feature of Veritas Volume Manager enables you to synchronize the Secondary. For more information on the Disk Group Split and Join feature, refer to the *Storage Foundation Administrator's Guide*. To set up replication using this method, ensure that you have a valid Disk Group Split and Join license on your system.

Linked break-off snapshots are another preferred method for off-host processing. Linked break-off snapshots are a variant of third-mirror break-off snapshots, which use the `vxsnap addmir` command to link a specially prepared volume with the data volume. You prepare the volume that is used for the snapshot in the same way you do for full-sized instant snapshots. However, unlike full-sized instant snapshots, you can set up this volume in a different disk group from the data volume. This makes linked break-off snapshots especially suitable for recurring off-host processing applications because it avoids the disk group split/join administrative step.

See "Example for setting up replication using Disk Group Split and Join" on page 515.

To synchronize the Secondary using Disk Group Split and Join

1 Create a snapshot plex for each data volume in the Primary RVG by issuing the following command on the Primary:

```
# vxassist -g diskgroup snapstart dv_name
```

You can use the `-b` option with the `vxassist snapstart` command to run the command in the background. Note that if you use the `-b` option of the `vxassist snapstart` command, you must wait for the snapshot plexes for all the data volumes in the RVG to be created and synchronized completely before you proceed to the next step. When the plex synchronization completes, the output of the `vxprint` command displays the state of the new snapshot plex as `SNAPDONE`.

2 Start a Primary Storage Checkpoint by issuing the following command on the Primary:

```
# vxrvg -g diskgroup -c checkpt_name checkstart \
local_rvgname
```

3 Take a snapshot of each data volume in the Primary RVG by issuing the following command on the Primary:

```
# vxrvg -g diskgroup snapshot local_rvgname
```

4 End the Storage Checkpoint by issuing the following command on the Primary:

```
# vxrvg -g diskgroup checkend local_rvgname
```

5 Split the snapshot volumes into a new disk group by issuing the following command on the Primary:

```
# vxdg split diskgroup new_diskgroup SNAP-dv_name ...
```

6 Rename each snapshot volume in the new disk group with the same name as the corresponding data volume in the Primary RVG by issuing the following command on the Primary:

```
# vxedit -g new_diskgroup rename SNAP-dv_name dv_name
```

7 Deport the split-off disk group, rename it to the same name as the disk group of the Primary RVG, and change the ownership of the split-off disk group to the Secondary host so that it may be automatically imported on the Secondary on reboot.

```
# vxdg -n diskgroup -h sec_hostname deport new_diskgroup
```

The argument sec_hostname is the name of the Secondary host displayed in the output of the `uname -a` command.

8 Physically remove the disks contained in the deported disk group by following the procedures recommended by the disk manufacturer; then attach the disks to the Secondary host.

9 On the Secondary, import the disks that were moved over from the Primary if not already imported:

```
# vxdg import diskgroup
```

10 Add the Secondary to the RDS by issuing the following command on the Primary:

```
# vradmin -g diskgroup addsec local_rvgname pri_hostname   \
     sec_hostname
```

11 Start replication by issuing the following command from any host in the RDS:

```
# vradmin -g diskgroup -c checkpt_name startrep \
    local_rvgname sec_hostname
```

The argument *sec_hostname* is the name of the Secondary host displayed in the output of the `vradmin printrvg` command. If the RDS contains only one Secondary, the `sec_hostname` is optional.

Using difference-based synchronization

You can synchronize the Secondary using difference-based synchronization when there is little difference between the Primary and Secondary data volumes in an RDS. Difference-based synchronization can be used to transfer data over the network when the application is active or inactive.

In difference-based synchronization, the `syncrvg` command generates MD5 checksums for the data blocks on the Primary data volume and the corresponding Secondary data volume and compares these checksums. The `syncrvg` command then transfers over the network only those blocks for which checksums do not match. These steps are repeated for the entire Primary data volume and Secondary data volume.

MD5 checksum is generated using the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

Difference-based synchronization is useful in situations such as:

- Storage Replicator Log (SRL) Overflow— To synchronize the Secondary when the SRL protection has not been set to use the Data Change Map (DCM).
- Failing Back to the Original Primary—To synchronize the original Primary data volumes with the new Primary data volumes.

The `vradmin syncrvg` command enables you to synchronize the Secondary RVG with the Primary RVG based on differences. You can issue the `vradmin syncrvg` command from any host in the RDS. The `vradmin syncrvg` command synchronizes the data volumes associated with the Secondary RVG in an RDS with the corresponding data volumes associated with its Primary RVG. The `vradmin syncrvg` command can be used to synchronize multiple Secondaries at the same time.

If a volume set is associated to an RVG, synchronizing the RVG only affects the component volumes of the volume set that are associated with the RVG. If the volume set includes component volumes that are not associated to the RVG, those volumes are not synchronized.

To synchronize Secondary RVG with Primary RVG based on differences

1. Verify that the RLINKs are detached.
2. Use the `-c checkpoint` option with the `vradmin syncrvg` command as follows:

```
# vradmin -g diskgroup -c checkpt_name syncrvg local_rvgname \
     sec_hostname....
```

Use the `-c` option with the `vradmin syncrvg` command when performing difference-based synchronization, to automatically start a Storage Checkpoint with the specified name. After the data volumes are synchronized the Storage Checkpoint is ended. This Storage Checkpoint can then be used to start replication using the `vradmin startrep` command.

The argument *`local_rvgname`* is the name of the RVG on the local host and represents its RDS.

The argument *`sec_hostname`* is a space-separated list of the names of the Secondary hosts as displayed in the output of the `vradmin printrvg` command.

The argument *`checkpt_name`* specifies the name of the Primary Storage Checkpoint of your choice.

Example—synchronizing the Secondary based on differences

This example explains how to synchronize the Secondary RVG `hr_rvg` on the Secondary host `london` with the Primary RVG on host `seattle`.

To synchronize the Secondary RVG, hr_rvg, on london with its Primary RVG on seattle based on differences

Before issuing this command, make sure that the RLINKs are detached.

```
# vradmin -g hrdg -c checkpt_presync syncrvg hr_rvg london
```

Note that you can use the `-c` option with the `vradmin syncrvg` command when performing difference-based synchronization to automatically start a Storage Checkpoint with the specified name. After the data volumes are synchronized the Storage Checkpoint is ended. This Storage Checkpoint can then be used to start replication using the `vradmin startrep` command.

The name checkpt_presync is the Primary Storage Checkpoint that you will create.

Examples for setting up a simple Volume Replicator configuration

The examples in this section explain how to use Volume Replicator (VVR) to set up a simple VVR configuration under different situations. The examples explain how to set up a VVR configuration with one Secondary and hence one RLINK; however, VVR enables you to configure and set up configurations with multiple Secondaries. The examples give the steps to replicate from the Primary host `seattle` to the Secondary host `london`.

Unless otherwise noted, the following assumptions apply to all the examples in this section:

- All VVR processes are running.
- The network connection between the Primary site `london` and the Secondary site `seattle` is active.
- The Primary and the Secondary have a disk group named `hrdg` with enough free space to create the VVR objects mentioned in the examples.
- Examples 1-5 assume that the Primary data volumes have been set up and contain data.
- The `/etc/vx/vras/.rdg` file on the Secondary host contains the Primary disk group ID. Ensure that each disk group ID entry in the `.rdg` file appears on a separate line. A Secondary can be added to an RDS only if the `/etc/vx/vras/.rdg` file on the Secondary host contains the Primary disk group ID. You can enter the following command to ensure that all disk groups are automatically added to the `/etc/vx/vras/.rdg` file:

  ```
  echo "+" >> /etc/vx/vras/.rdg
  ```

  Use the `vxprint -l diskgroup` command to display the disk group ID of the disk group `hrdg`, which is being used.

Configuration Considerations:

Consider the following in each example:

- The data volumes on the Secondary must have the same names and sizes as the data volumes on the Primary.
- The name of the Storage Replicator Log (SRL) on the Secondary must be the same as the name of the SRL on the Primary.
- The SRL must be created on disks that do not have other volumes on them.
- The data volumes and SRL must be mirrored.

In the examples, each data volume is 4 GB; the Primary and Secondary SRL are 4 GB each.

The examples in this chapter use the following names:

Primary Hostname: `seattle`

`hrdg`	Disk group
`hr_rvg`	Primary RVG
`rlk_london_hr_rvg`	Primary RLINK to Secondary `london`
`hr_dv01`	Primary data volume #1
`hr_dv02`	Primary data volume #2
`hr_srl`	Primary SRL volume

Secondary Hostname: `london`

`hrdg`	Disk group
`hr_rvg`	Secondary RVG
`rlk_seattle_hr_rvg`	Secondary RLINK to Primary `seattle`
`hr_dv01`	Secondary data volume #1
`hr_dv02`	Secondary data volume #2
`hr_srl`	Secondary SRL volume

Creating a Replicated Data Set for the examples

This procedure describes how to create an example Replicated Data Set (RDS).

Note: This example assumes that the disks available on the system have labels such as disk01, disk02, disk03 and so on.

To create the example RDS

1. Create the data volumes on the Secondary host `london`. Use different disks for the data volumes and SRL.

```
# vxassist -g hrdg make hr_dv01 4G \
     layout=mirror logtype=dcm mirror=2 disk01 disk02
 # vxassist -g hrdg make hr_dv02 4G \
     layout=mirror logtype=dcm mirror=2 disk03 disk04
```

2. Create the SRL on disks that do not have other volumes on them by typing the following command on the Primary `seattle` and the Secondary `london`:

```
# vxassist -g hrdg make hr_srl 4G mirror=2 disk05 disk06
```

Note: You must create the SRL on disks that do not contain any other volume.

3. Create the Primary RVG of the RDS by typing the following command on the Primary `seattle`:

```
# vradmin -g hrdg createpri hr_rvg hr_dv01,hr_dv02 hr_srl
```

4. Make sure the `/etc/vx/vras/.rdg` file on the Secondary host `london` contains the Primary disk group ID of `hrdg`; then add the Secondary `london` to the RDS by typing the following command on the Primary `seattle`:

```
# vradmin -g hrdg addsec hr_rvg seattle london
```

Example for setting up replication using full synchronization

This example assumes that the RDS has been created using the example procedure.

See "Creating a Replicated Data Set for the examples" on page 512.

You can use full synchronization with Storage Checkpoint when the application is active or inactive.

To synchronize the Secondary using full synchronization with Storage Checkpoint

1. Synchronize the Secondary RVG `hr_rvg` on `london` with its Primary RVG on `seattle` using full synchronization with Storage Checkpoint:

```
# vradmin -g hrdg -full -c chkpt_presync syncrvg hr_rvg \
     london
```

2. Start replication with Storage Checkpoint by issuing the following command on any host in the RDS:

```
# vradmin -g hrdg -c chkpt_presync startrep hr_rvg london
```

Example for setting up replication using block-level backup and checkpointing

This example assumes that the RDS has been created using the example procedure.

See "Creating a Replicated Data Set for the examples" on page 512.

You can synchronize the Secondary using block-level backup and checkpointing when the application is active or inactive.

To synchronize the Secondary using block-level backup and checkpointing

1. Start a Storage Checkpoint on the Primary:

```
# vxrvg -g hrdg -c checkpt_presync checkstart hr_rvg
```

   Note down the Storage Checkpoint name you use, that is, `checkpt_presync`.

2. Perform a block-level backup of the data volumes in the Primary RVG.
3. End the Primary Storage Checkpoint when the backup is complete:

```
# vxrvg -g hrdg checkend hr_rvg
```

4. Restore the backup to the Secondary data volumes.
5. Use the `vxrvg cplist` command on the Primary to check whether the Storage Checkpoint you created is still valid.

   If the Storage Checkpoint has overflowed, repeat 1 to 4.

   The output resembles:

```
Name              MBytes       % Log   Started/Completed
checkpt_presync   10               9          Completed
```

6 Start replication using the Storage Checkpoint:

```
# vradmin -g hrdg -c checkpt_presync startrep hr_rvg london
```

7 On the Primary, check whether the `consistent` flag is set on the Primary RLINK using the `vxprint` command. The RLINK becomes `consistent` only after the data contained in the Storage Checkpoint is sent to the Secondary. Wait and then issue the following command on the Primary:

```
# vxprint -g hrdg -l rlk_london_hr_rvg
```

If the Secondary is consistent, the synchronization was successful.

If the Storage Checkpoint overflows before the Secondary becomes consistent, the synchronization process has failed. Increase the size of the SRL.

See "Changing the size of the SRL on the Primary and the Secondary" on page 223.

Then restart the procedure beginning at step 1.

It is likely that there might be writes beyond the Storage Checkpoint that are yet to be sent to the Secondary after `consistent` flag is set on the RLINK. Use the `vxrlink status` command to check whether the RLINK is up-to-date:

```
# vxrlink -g hrdg status rlk_london_hr_rvg
```

The same backup and the corresponding Storage Checkpoint can be used to set up additional Secondary hosts while the Storage Checkpoint is still valid. If a Storage Checkpoint has overflowed, its corresponding backup cannot be used to resynchronize the Secondary. Eventually, any Storage Checkpoint that becomes STALE is unusable. There is no warning to indicate that this has occurred. However, the `vxrvg cplist` command indicates that the Storage Checkpoint has overflowed and hence is unusable.

See "Displaying a list of Storage Checkpoints" on page 176.

Example for setting up replication using Disk Group Split and Join

This procedure assumes you have not already created a sample Replicated Data Set.

To set up replication using Disk Group Split and Join

1. Create the SRL on disks that do not have other volumes on them by typing the following command on the Primary `seattle`:

   ```
   # vxassist -g hrdg make hr_srl 4G mirror=2
   ```

   Note: You must create the SRL on disks that do not contain any other volume.

2. Create the Primary RVG of the RDS by typing the following command on the Primary `seattle`:

   ```
   # vradmin -g hrdg createpri hr_rvg hr_dv01,hr_dv02 hr_srl
   ```

3. Create a snapshot plex for each data volume in the Primary RVG by issuing the following commands on the Primary `seattle`:

   ```
   # vxassist -g hrdg snapstart hr_dv01
   # vxassist -g hrdg snapstart hr_dv02
   ```

   You can use the `-b` option with the `vxassist snapstart` command to run the command in the background. Note that if you use the `-b` option of the `vxassist snapstart` command, you must wait for the snapshot plexes for all the data volumes in the RVG to be created and synchronized completely before you proceed to the next step. When the plex synchronization completes, the output of the `vxprint` command displays the state of the new snapshot plex as `SNAPDONE`.

4. Start a Primary Storage Checkpoint by issuing the following command on the Primary `seattle`:

   ```
   # vxrvg -g hrdg -c checkpt_presync checkstart hr_rvg
   ```

5. Take snapshot of each data volume in the Primary RVG by issuing the following command on the Primary `seattle`:

   ```
   # vxrvg -g hrdg snapshot hr_rvg
   ```

6. End the Storage Checkpoint by issuing the following command on the Primary `seattle`:

   ```
   # vxrvg -g hrdg checkend hr_rvg
   ```

7 Split the snapshot volumes into a new disk group by issuing the following command on the Primary `seattle`:

```
# vxdg split hrdg new_hrdg SNAP-hr_dv01 SNAP-hr_dv02
```

8 Rename each snapshot volume in the new disk group with the same name as the corresponding data volume in the Primary RVG by issuing the following commands on the Primary `seattle`:

```
# vxedit -g new_hrdg rename SNAP-hr_dv01 hr_dv01
# vxedit -g new_hrdg rename SNAP-hr_dv02 hr_dv02
```

9 Deport the split-off disk group, rename it to the same name as the disk group of the Primary RVG and change the ownership of the split-off disk group to be the Secondary host so that it may be automatically imported on the Secondary on reboot. To do this, issue the following command on the Primary `seattle`:

```
# vxdg -n hrdg -h london deport new_hrdg
```

10 Physically remove the disks contained in the deported disk group by following the procedures recommended by the disk manufacturer; then attach the disks to the Secondary host.

11 On the Secondary `london`, import the disks that were moved over from the Primary if not already imported.

```
# vxdg import hrdg
```

12 Create the SRL on disks that do not have other volumes on them by typing the following command on the Secondary `london`:

```
# vxassist -g hrdg make hr_srl 4G mirror=2
```

Note: You must create the SRL on disks that do not contain any other volume.

---

13 Make sure the `/etc/vx/vras/.rdg` file on the Secondary host `london` contains the Primary disk group ID of `hrdg`; then add the Secondary to the RDS by issuing the following command on the Primary `seattle`:

```
# vradmin -g hrdg addsec hr_rvg seattle london
```

14 Start replication by issuing the following command on any host in the RDS:

```
# vradmin -g hrdg -c checkpt_presync startrep hr_rvg  london
```

Example for setting up replication using differences-based synchronization

This example assumes that the RDS has been created using the example procedure.

See "Creating a Replicated Data Set for the examples" on page 512.

You can synchronize the Secondary using difference-based synchronization with Storage Checkpoint when the application is active or inactive.

To synchronize the Secondary using difference-based synchronization with Storage Checkpoint

1. Synchronize the Secondary RVG `hr_rvg` on `london` with its Primary RVG on `seattle` using difference-based synchronization with Storage Checkpoint:

```
# vradmin -g hrdg -c chkpt_presync syncrvg hr_rvg london
```

2. Start replication with Storage Checkpoint by issuing the following command on any host in the RDS:

```
# vradmin -g hrdg -c chkpt_presync startrep hr_rvg london
```

Example for setting up replication when data volumes are initialized with zeroes

Because the Primary data volumes are initialized with zeroes, the data on the Secondary node need not be synchronized with the Primary. However, we recommend that you zero initialize the Secondary data volumes.

To set up replication when data volumes are initialized with zeroes

1. Create the data volumes by typing the following commands on the Primary `seattle` and Secondary `london`. Use different disks for the data volumes and SRL.

```
# vxassist -g hrdg make hr_dv01 4G layout=mirror \
   logtype=dcm mirror=2 init=zero disk01 disk02
# vxassist -g hrdg make hr_dv02 4G layout=mirror \
   logtype=dcm mirror=2 init=zero disk03 disk04
```

2. Create the SRL on disks that do not have other volumes on them by typing the following command on the Primary `seattle` and Secondary `london`:

```
# vxassist -g hrdg make hr_srl 4G mirror=2 disk05 disk06
```

3 Create the Primary RVG of the RDS by typing the following command on the Primary `seattle`:

```
# vradmin -g hrdg createpri hr_rvg hr_dv01,hr_dv02 hr_srl
```

4 Make sure the `/etc/vx/vras/.rdg` file on the Secondary host `london` contains the Primary disk group ID of `hrdg`; then add the Secondary `london` to the RDS by typing the following command on the Primary `seattle`:

```
# vradmin -g hrdg addsec hr_rvg seattle london
```

Note: Do not start the application or mount the file system before completing step 5.

---

5 Start replication using the option `-f` with the `vradmin startrep` command by typing the following command on any host in the RDS:

```
# vradmin -g hrdg -f startrep hr_rvg london
```

After completing this step, start the application.

Appendix E

Migrating VVR from Internet Protocol version 4 to Internet Protocol version 6

This appendix includes the following topics:

- Overview of VVR migration from IPv4 to IPv6
- About migrating to IPv6 when VCS global clustering and VVR agents are not configured
- About migrating to IPv6 when VCS global clustering and VVR agents are configured
- About migrating to IPv6 when VCS global clustering and VVR agents are configured in the presence of a bunker

Overview of VVR migration from IPv4 to IPv6

Traditionally, replication between the Primary and the Secondary sites within VVR takes place through the IPv4 network. However, you can migrate VVR from IPv4 to the IPv6 network.

To migrate from IPv4 to IPv6, do the following in the order presented:

- Configure IPv6 addresses on all the hosts in all the VVR sites.
- Add new IP and NIC resources for the IP addresses in the VCS configuration. Creating new resources does not affect VVR replication in IPv4.

- Use the `vradmin changeip` command to migrate VVR from IPv4 to IPv6. The command has the following syntax.

  ```
  vradmin -g diskgroup changeip newpri=virtual_IPv6_1_address \
  newsec=virtual_IPv6_2_address
  ```

- Remove the IPv4 network from all the sites.

The migration is supported in the following environments:

- VCS global clustering and VVR agents are not configured.
- VCS global clustering and VVR agents are configured.
- VCS global clustering and VVR agents are configured in the presence of a bunker.

About migrating to IPv6 when VCS global clustering and VVR agents are not configured

To migrate to IPv6, do the following:

- Understand the current configuration
- Meet the prerequisites for migration
- Perform the actual migration

Note: The following steps assume that a separate network interface is used for IPv6 connectivity.

Understanding the current IPv4 configuration when VCS global clustering and VVR agents are not configured

Before you migrate VVR from IPv4 to IPv6, review your current configuration. The sample configuration is as follows:

- There are two nodes each on the Primary and the Secondary site of replication.
- RVG objects are created on both the sites. Replication between the sites uses the IPv4 address.
- A failover service of the Virtual IPv4 address is created for high availability.
- The failover service group consists of one IP resource and one NIC resource for the virtual IPv4 address.

- On every system, the state of the resources in the failover service group is similar to the following:

```
# hares -state | grep -i res
#Resource     Attribute        System     Value
ipres         State            swlx25     ONLINE
ipres         State            swlx27     OFFLINE
nicres        State            swlx25     ONLINE
nicres        State            swlx27     ONLINE
```

- On both the systems, the state of the failover service group is similar to the following:

```
# hastatus -summ | grep -i VVRGRP
B  VVRGRP          swlx25               Y          N      ONLINE
B  VVRGRP          swlx27               Y          N      OFFLINE
```

- The contents of the `main.cf` file in the failover service group are displayed as follows:

```
group VVRGRP (
        SystemList = { swlx25 = 0, swlx27 = 1 }
        AutoStartList = { swlx25 }
        )

        IP ipres (
                Device = eth0
                Address = "10.209.87.186"
                NetMask = "255.255.252.0"
        )

        NIC nicres (
                Enabled = 1
                Device = eth0
        )

        ipres requires nicres

        // resource dependency tree
        //
        //      group VVRGRP
        //      {
        //      IP ipres
        //          {
```

```
//          NIC nicres
//          }
//      }
```

- After you create the service group VVRGRP, the output of the `ifconfig` command is as follows:

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:30:6E:2B:4F:B6
          inet addr:10.209.85.35 Bcast:10.209.87.255 Mask:255.255.252.0
          inet6 addr: fe80::230:6eff:fe2b:4fb6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:172660 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17172 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16012095 (15.2 Mb)  TX bytes:3357637 (3.2 Mb)
          Interrupt:96

eth0:0    Link encap:Ethernet  HWaddr 00:30:6E:2B:4F:B6
          inet addr:10.209.87.186  Bcast:0.0.0.0  Mask:255.255.252.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:96

eth1      Link encap:Ethernet  HWaddr 00:11:43:37:DA:68
          inet6 addr: fe80::211:43ff:fe37:da68/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:33161 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18635 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:3315207 (3.1 Mb)  TX bytes:3288512 (3.1 Mb)

eth2      Link encap:Ethernet  HWaddr 00:11:43:37:DA:69
          inet6 addr: fd4b:454e:205a:111:211:43ff:fe37:da69/64
          Scope:Global
          inet6 addr: fe80::211:43ff:fe37:da69/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:71504 errors:0 dropped:0 overruns:0 frame:0
          TX packets:856 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:7700084 (7.3 Mb)  TX bytes:285676 (278.9 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

```
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:38 errors:0 dropped:0 overruns:0 frame:0
TX packets:38 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:4876 (4.7 Kb)  TX bytes:4876 (4.7 Kb)
```

Migration prerequisites when VCS global clustering and VVR agents are not configured

Before you start the migration, meet the following prerequisites:

- IPv6 must be configured on all the systems. The systems must be dual-node; that is, they have both IPv4 and IPv6 addresses configured.
- Modify the `/etc/resolv.conf` file so that it includes entries for both the IPv4 and IPv6 DNS servers and domain names.
- If the DNS servers are not available, add the IPv6 hostname and corresponding IP address entries in the `/etc/hosts` file.

Migrating to IPv6 when VCS global clustering and VVR agents are not configured

To migrate to IPv6, do the following:

- Modify the failover service group used for replication to IPv6.
- Migrate VVR RLINKs from IPv4 to IPv6.
- Remove the IPv4 resources from the VCS configuration.

Modifying the failover service group used for replication to IPv6

On the Primary site, in the existing failover service group created for the IPv4 virtual IP, create an IP resource and/or NIC resource for the virtual IPv6 address. In the following example, the service group is VVRGRP.

To modify the failover service group used for replication to IPv6

1. Enable write operations on the VCS configuration. Enter the following:

```
# haconf -makerw
```

2. Add the NIC resource for the IPv6 address and configure the related attributes:

```
# hares -add  nicres_v6  NIC  VVRGRP
# hares -modify nicres_v6 Device  eth1
# hares -modify nicres_v6 Enabled 1
# hares -probe nicres_v6 -sys node1
# hares -probe nicres_v6 -sys node2
```

3. Add the IP resource for the IPv6 address and configure the necessary attributes.

```
# hares -add  ipres_v6  IP  VVRGRP
# hares -modify ipres_v6 Device  eth1
# hares -modify ipres_v6 \
  Address fd4b:454e:205a:111:211:43ff:feaa:af71
# hares -modify ipres_v6 Enabled 1
# hares -modify ipres_v6 PrefixLen 64
# hares -probe ipres_v6 -sys node1
# hares -probe ipres_v6 -sys node2
# hares -online ipres_v6 -sys node1
```

In the example above, the new IP resource `ipres_v6` is configured for the virtual IPv6 address.

Note: If you configure the IPv6 address on the same NIC that is used for IPv4, you do not need to create a new NIC resource. Instead, use the existing NIC resource and link the new IP resource to it. If you configure the IPv6 address on a separate interface, create a new NIC resource (`nicres_v6`) for the IPv6 interface.

---

4. Link the IP resource to the NIC resource, Enter the following:

```
# hares -link ipres_v6 nicres_v6
```

5 Save your configuration changes. Enter the following:

```
# haconf -dump -makero
```

6 Repeat step 1 to 5 on the Secondary site.

The virtual IP will be different on each site.

Note: Do not stop VCS; do not modify the existing service group in the `main.cf` file. These changes affect replication between the sites because the service group will go offline.

After you save the configuration, the service group (VVRGRP) has the following settings:

- The state of resources after configuring the virtual IPv6 address:

```
# hares -state | grep -i res
Resource     Attribute        System     Value
ipres        State            swlx25     ONLINE
ipres        State            swlx27     OFFLINE
ipres_v6     State            swlx25     ONLINE
ipres_v6     State            swlx27     OFFLINE
nicres       State            swlx25     ONLINE
nicres       State            swlx27     ONLINE
nicres_v6    State            swlx25     ONLINE
nicres_v6    State            swlx27     ONLINE
```

- The state of the failover service group (VVRGRP):

```
# hastatus -summ | grep -i vvr
B  VVRGRP          swlx25               Y          N      ONLINE
B  VVRGRP          swlx27               Y          N      OFFLINE
```

- The updated contents of `main.cf` file after adding new resources:

```
group VVRGRP (
        SystemList = { swlx25 = 0, swlx27 = 1 }
        AutoStartList = { swlx25 }
        )

        IP ipres (
                Device = eth0
                Address = "10.209.87.186"
```

```
        NetMask = "255.255.252.0"
        )

IP ipres_v6 (
        Device = eth1
        Address = "fd4b:454e:205a:111:211:43ff:feaa:af71"
        PrefixLen = 64
        )

NIC nicres (
        Enabled = 1
        Device = eth0
        )

NIC nicres_v6 (
        Enabled = 1
        Device = eth1
        )

ipres requires nicres
ipres_v6 requires nicres_v6

// resource dependency tree
//
//      group VVRGRP
//      {
//      IP ipres
//          {
//          NIC nicres
//          }
//      IP ipres_v6
//          {
//          NIC nicres_v6
//          }
//      }
```

- After you modify the service group VVRGRP, the output of the `ifconfig` command is as follows:

```
# ifconfig -a
eth0    Link encap:Ethernet  HWaddr 00:30:6E:2B:4F:B6
        inet addr:10.209.85.35  Bcast:10.209.87.255 Mask:255.255.252.0
        inet6 addr: fe80::230:6eff:fe2b:4fb6/64 Scope:Link
```

```
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:224685 errors:0 dropped:0 overruns:0 frame:0
        TX packets:21258 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:20728359 (19.7 Mb)  TX bytes:3942243 (3.7 Mb)
        Interrupt:96

eth0:0  Link encap:Ethernet  HWaddr 00:30:6E:2B:4F:B6
        inet addr:10.209.87.186 Bcast:0.0.0.0  Mask:255.255.252.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        Interrupt:96

eth1    Link encap:Ethernet  HWaddr 00:11:43:37:DA:68
        inet6 addr: fe80::211:43ff:fe37:da68/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:43279 errors:0 dropped:0 overruns:0 frame:0
        TX packets:24072 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100
        RX bytes:4281957 (4.0 Mb)  TX bytes:3960499 (3.7 Mb)

eth2    Link encap:Ethernet  HWaddr 00:11:43:37:DA:69
        inet6 addr: fd4b:454e:205a:111:211:43ff:feaa:af71/64
        Scope:Global
        inet6 addr: fd4b:454e:205a:111:211:43ff:fe37:da69/64
        Scope:Global
        inet6 addr: fe80::211:43ff:fe37:da69/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:92654 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1213 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100
        RX bytes:9995434 (9.5 Mb)  TX bytes:425815 (415.8 Kb)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:44 errors:0 dropped:0 overruns:0 frame:0
        TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:5668 (5.5 Kb)  TX bytes:5668 (5.5 Kb)
```

In this example, the virtual IPv4 address is `eth0: 10.209.87.186` and the virtual IPv6 address is `eth2: fd4b:454e:205a:111:211:43ff:feaa:af71`.

Note: Replication continues to use the virtual IPv4 address. The virtual IPv6 address is successfully configured without affecting the existing VVR replication.

Because both the virtual IPv4 and IPv6 addresses are online, you can migrate the Primary and Secondary VVR sites from the IPv4 network to the IPv6 network without disrupting the replication between the sites.

Migrating the VVR links from IPv4 to IPv6 when VCS global clustering and VVR agents are not configured

Before you perform the steps in this procedure, make sure all sites have both IPv4 and IPv6 addresses online.

In the following example, the hostnames of the systems configured with IPv6 match the ones in the IPv4 network. The hostnames of the virtual IPs in the IPv4 and IPv6 networks also match.

To migrate the VVR links from IPv4 to IPv6

1 Check the replication status. Enter the following:

```
# vradmin -g hrdg repstatus hr_rvg

Replicated Data Set: hr_rvg
Primary:
  Host name:                  10.209.87.170
  RVG name:                   hr_rvg
  DG name:                    hrdg
  RVG state:                  enabled for I/O
  Data volumes:               1
  VSets:                      0
  SRL name:                   hr_srl
  SRL size:                   800.00 M
  Total secondaries:          1

Secondary:
  Host name:                  10.209.87.171
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Data status:                consistent, up-to-date
  Replication status:         replicating (connected)
  Current mode:               synchronous
  Logging to:                 SRL
  Timestamp Information:      behind by 0h 0m 0s
```

2 On the Primary site, migrate the VVR links from the IPv4 to IPv6 network using the `vradmin changeip` command:

```
# vradmin -g dg changeip rvg newpri=Virtual_IPv6_Addr \
  newsec=Virtual_IPv6_Addr
```

For example:

```
# vradmin -g hrdg changeip hr_rvg newpri=fd4b:454e:205a:111:211:43ff:feaa:af70 \
  newsec=fd4b:454e:205a:111:211:43ff:feaa:af71

Message from Primary:
VxVM VVR vxrlink INFO V-5-1-3614 Secondary data volumes detected with rvg hr_rvg as parent:
VxVM VVR vxrlink INFO V-5-1-6183 hr_dv01:   len=10485760     primary_datavol=hr_dv01
```

3 Check the replication status. Enter the following:

```
# vradmin -g hrdg repstatus hr_rvg
Replicated Data Set: hr_rvg
Primary:
  Host name:                  fd4b:454e:205a:111:211:43ff:feaa:af70
  RVG name:                   hr_rvg
  DG name:                    hrdg
  RVG state:                  enabled for I/O
  Data volumes:               1
  VSets:                      0
  SRL name:                   hr_srl
  SRL size:                   800.00 M
  Total secondaries:          1

Secondary:
  Host name:                  fd4b:454e:205a:111:211:43ff:feaa:af71
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Data status:                consistent, up-to-date
  Replication status:         replicating (connected)
  Current mode:               synchronous
  Logging to:                 SRL
  Timestamp Information:      behind by 0h 0m 0s
```

4 Verify that the RLINKs now use the IPv6 network for replication. Enter the following:

```
# vxprint -Pl
```

The IPv4 network is still present. You can remove the IPv4 network to complete the migration process.

See "Removing the IPv4 resources from the VCS configuration when VCS global clustering and VVR agents are not configured" on page 531.

Removing the IPv4 resources from the VCS configuration when VCS global clustering and VVR agents are not configured

To remove the IPv4 resources from the VCS configuration do the following:

- Remove the IPv4 resources from the Primary and Secondary sites.

- Remove the IPv4 network on the Primary and Secondary sites. This task is optional. Only perform this task if you want to have an IPv6-only environment.

To remove the IPv4 resources from the Primary and Secondary sites

1. Make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

2. Take the system offline. Enter the following:

```
# hares -offline ipres -sys swlx25
```

3. Unlink the IP and NIC resources. Enter the following:

```
# hares -unlink ipres nicres
```

4. Delete the IP and NIC resources. Enter the following:

```
# hares -delete ipres
# hares -delete nicres
```

5. Save your configuration changes. Enter the following:

```
# haconf -dump -makero
```

6. Verify that only IPv6 addresses are defined.

```
# hares -state | grep -i res
#Resource    Attribute        System     Value
ipres_v6     State            swlx25     ONLINE
ipres_v6     State            swlx27     OFFLINE
nicres_v6    State            swlx25     ONLINE
nicres_v6    State            swlx27     ONLINE
```

To remove the IPv4 network on the Primary and Secondary sites (optional)

1. Down the interface on which IPv4 was configured.

   ```
   # ifdown eth0
   ```

2. Remove the IPv4 network cable. This does not impact replication because the VVR configuration has been migrated to use IPv6 on all sites.
3. Remove the IPv4 entries from `/etc/hosts` and `/etc/resolv.conf` files on all the hosts on both the Primary and Secondary sites.

   Note: Do not remove the IPv4 loopback entry from the `/etc/hosts` file. If you remove the loopback entry, and replication is affected, stop and restart the VVR service using the `vxstart_vvr stop` and `vxstart_vvr start` commands.

You have migrated VVR to an IPv6-only network.

About migrating to IPv6 when VCS global clustering and VVR agents are configured

To migrate to IPv6 when VCS global clustering and VVR agents are configured, do the following:

- Understand the current configuration
- Meet the migration prerequisites
- Perform the actual migration

Note: The following examples assume that a separate network interface is used for IPv6 connectivity.

Understanding the current IPv4 configuration when VCS global clustering and VVR agents are configured

The following procedures use the following sample configuration:

- VCS global clustering and VVR agent setup uses the IPv4 addresses.
- There are two sites, Primary and Secondary. The Primary site has two systems, `vvrias05` and `vvrias06`. The Secondary also has two systems, `vvrias07` and `vvrias08`. Replication is running between these two sites using the IPv4 addresses.

- VCS global clustering and VVR agents are used to control the VVR functionality.
- The VCS global clustering and VVR agent service group is similar to the following:

```
cluster sfcfs_gco1 (
        ClusterAddress = "10.209.87.162"
        SecureClus = 1
        HacliUserLevel = COMMANDROOT
        )

remotecluster sfcfs_gco2 (
        ClusterAddress = "10.209.87.163"
        )

heartbeat Icmp (
        ClusterList = { sfcfs_gco2 }
        Arguments @sfcfs_gco2 = { "10.209.87.163" }
        )

system swlx20 (
        )

system swlx21 (
        )

group ClusterService (
        SystemList = { swlx20 = 0, swlx21 = 1 }
        AutoStartList = { swlx20, swlx21 }
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
        )

        Application wac (
                StartProgram = "/opt/VRTSvcs/bin/wacstart"
                StopProgram = "/opt/VRTSvcs/bin/wacstop"
                MonitorProcesses = { "/opt/VRTSvcs/bin/wac" }
                RestartLimit = 3
                )

        IP gcoip (
                Device = eth0
                Address = "10.209.87.162"
                NetMask = "255.255.252.0"
```

```
        )

NIC gconic (
        Device = eth0
        )

gcoip requires gconic
wac requires gcoip
```

- The service group which monitors or controls the logowner is similar to the following:

```
group rlogowner (
        SystemList = { swlx20 = 0, swlx21 = 1 }
        AutoStartList = { swlx20, swlx21 }
        )

        IP logowner_ip (
                Device = eth0
                Address = "10.209.87.164"
                )

        NIC nic (
                Device = eth0
                )

        RVGLogowner logowner (
                RVG = rac1_rvg
                DiskGroup = shared_dg
                )

        requires group RVGgroup online local firm
        logowner requires logowner_ip
        logowner_ip requires nic
```

Migration prerequisites when VCS global clustering and VVR agents are configured

Before you start the migration, meet the following prerequisites:

- IPv6 must be configured on all the systems. The systems must be dual-node; that is, they have both IPv4 and IPv6 addresses configured.

- Modify the `/etc/resolv.conf` file to include the entries for both the IPv4 and IPv6 DNS servers and domain names.
- If the DNS servers are not available, add the IPv6 hostname and corresponding IP address entries in the `/etc/hosts` file.

Migrating to IPv6 when VCS global clustering and VVR agents are configured

To migrate the network to IPv6, do the following:

- Migrate the VCS global clustering service group to IPv6.
- Add the IP and NIC resources for IPv6 addresses in the RVG agent group.
- Migrate the VVR RLINKs from IPv4 to IPv6.
- Remove IPv4 resources from the VCS configuration.

Migrating the VCS global clustering service group to IPv6 when VCS global clustering and VVR agents are configured

An online migration of the VCS global clustering service group is not supported. Put the service group offline and then modify the required parameters.

To migrate the VCS global clustering service to IPv6, do the following:

- Take the ClusterService Group offline on the Primary and Secondary sites.
- Edit the VCS configuration for IPv6 addresses.
- Bring the ClusterService Group online on the Primary and Secondary sites.

To take the ClusterService Group offline on the Primary and Secondary sites

◆ On the Primary and Secondary sites, enter the following command:

```
# hagrp -offline -force ClusterService -sys node_name
```

For example, on the Primary site, enter the following:

```
# hagrp -offline -force ClusterService -sys swlx25
```

On the Secondary site, enter the following:

```
# hagrp -offline -force ClusterService -sys swlx27
```

Note: Putting the service group offline does not impact the replication between the sites.

To edit the VCS configuration for IPv6 addresses

1. On the Primary site, make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

2. Modify the VCS global clustering related attribute values to support the IPv6 environment. Enter the following:

```
# haclus -modify ClusterAddress \
  fd4b:454e:205a:111:213:72ff:fe5b:2f67 -clus sfcfs_gco1
# haclus -modify ClusterAddress \
  fd4b:454e:205a:111:211:43ff:fede:1e11 -clus sfcfs_gco2
# hahb -modify Icmp Arguments \
  fd4b:454e:205a:111:211:43ff:fede:1e11 -clus sfcfs_gco2
# hares -modify gcoip Address \
  fd4b:454e:205a:111:213:72ff:fe5b:2f67
# hares -modify gcoip Enabled 1
# hares -modify gcoip PrefixLen 64
# hares -modify gcoip NetMask ""
# hares -modify gcoip Device eth1
# hares -modify gconic Device eth1
# hares -modify gconic Enabled 1
```

3. Save your configuration changes. Enter the following:

```
# haconf -dump -makero
```

4. Modify the VCS global clustering related attributes on the Secondary site using steps 1 through 3.

To bring the ClusterService Group online on the Primary and Secondary sites

- On the Primary and Secondary sites, enter the following:

```
# hagrp -online ClusterService -sys node_name
```

For example, on the Primary site, enter the following:

```
# hagrp -online -force ClusterService -sys swlx25
```

On the Secondary site, enter the following:

```
# hagrp -online -force ClusterService -sys swlx27
```

The global clustering ClusterService Group in the VCS `main.cf` configuration file now looks similar to the following:

```
cluster sfcfs_gco1 (
        ClusterAddress = "fd4b:454e:205a:111:213:72ff:fe5b:2f67"
        SecureClus = 1
        HacliUserLevel = COMMANDROOT
        )

remotecluster sfcfs_gco2 (
        ClusterAddress = "fd4b:454e:205a:111:211:43ff:fede:1e11"
        )

heartbeat Icmp (
        ClusterList = { sfcfs_gco2 }
        Arguments @sfcfs_gco2 = { "fd4b:454e:205a:111:211:43ff:fede:1e11" }
        )

system swlx20 (
        )

system swlx21 (
        )

group ClusterService (
        SystemList = { swlx20 = 0, swlx21 = 1 }
        AutoStartList = { swlx20, swlx21 }
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
        )

        Application wac (
                StartProgram = "/opt/VRTSvcs/bin/wacstart"
                StopProgram = "/opt/VRTSvcs/bin/wacstop"
                MonitorProcesses = { "/opt/VRTSvcs/bin/wac" }
                RestartLimit = 3
                )

        IP gcoip (
                Device = eth1
                Address = "fd4b:454e:205a:111:213:72ff:fe5b:2f67"
                PrefixLen = 64
                )

        NIC gconic (
                Device = eth1
```

```
        )

gcoip requires gconic
wac requires gcoip
```

Adding IP and NIC resources for IPv6 addresses in the RVG agent group when VCS global clustering and VVR agents are configured

Currently, replication between the Primary and Secondary sites uses an IPv4 network. Before you can migrate the network to IPv6, you must add the IP and NIC resources to VCS control. You can do this in one of the following ways:

- Add new IP and NIC resources for IPv6 in the RVG group.
- Modify the current IP and NIC resources in the RVG group.

The first method, adding new IP and NIC resources for IPv6, ensures high availability. For example, suppose you add resources for IP and NIC resources for IPv6 addresses in the `main.cf` on the node in the Primary site from where replication is active. If the Primary site crashes before VVR migration to IPv6 from IPv4, the application service group fails over to the other node in the Primary site and replication continues using the IPv4 address. The existing IP and NIC resources are still in the `main.cf` file.

The second method, modifying the current IP and NIC resources, hampers replication. If a node in the Primary site crashes as you modify the existing IP resource, the original IPv4 virtual IP is lost. The application service group fails over to the other node, but the IPv4 IP address does not come online on the other node because you changed the IP resource from an IPv4 address to an IPv6 address. Instead, the IPv6 IP comes online. However, VVR is not configured use the IPv6 IP for replication, and replication pauses or stops. The IPv4 IP is not available on the other node.

To add the IP and NIC resources on the Primary and Secondary sites

1. Make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

2. Modify the IP and NIC attributes of the service group which has the RVG logowner agent resource for controlling the logowner. On the Primary site, enter the following:

```
# haconf -makerw
# hares -modify logowner_ip Device eth1
# hares -modify logowner_ip \
  Address fd4b:454e:205a:111:213:72ff:fe59:4a23
# hares -modify logowner_ip Enabled 1
# hares -modify nic Device eth1
# hares -modify nic Enabled 1
# hares -modify logowner_ip PrefixLen 64
# haconf -dump -makero
```

Note: VVR replication between sites is not impacted by these modifications.

3. Repeat step 2 on the Secondary site, using the appropriate system names and IP addresses.

4. Save your configuration changes. Enter the following:

```
# haconf -dump -makero
```

Migrating VVR RLINKs from IPv4 to IPv6 when VCS global clustering and VVR agents are configured

To migrate VVR RLINKs from IPv4 to IPv6

1. Migrate VVR replication from the IPv4 network to the IPv6 network. The IPs specified for the `newpri` and `newsec` attributes must be the same as the ones specified for the Primary and the Secondary logowner IP.

   For example:

   ```
   # vradmin -g hrdg changeip hr_rvg \
     newpri=fd4b:454e:205a:111:213:72ff:fe59:4a23 \
     newsec=fd4b:454e:205a:111:213:72ff:fe58:3d8b
   ```

2. Verify that the replication RLINKs on both the sites now use the IPv6 addresses. Enter the following:

   ```
   # vxprint -Pl
   ```

   The service group which monitors or controls the logowner on the Primary site is now similar to the following:

   ```
   group rlogowner (
           SystemList = { swlx20 = 0, swlx21 = 1 }
           AutoStartList = { swlx20, swlx21 }
           )

           IP logowner_ip (
                   Device = eth1
                   Address = "fd4b:454e:205a:111:213:72ff:fe59:4a23"
                   PrefixLen = 64
                   )

           NIC nic (
                   Device = eth1
                   )

           RVGLogowner logowner (
                   RVG = rac1_rvg
                   DiskGroup = shared_dg
                   )
   ```

The entire VCS global clustering and VVR setup now uses the IPv6 network connectivity on both the Primary and Secondary sites. However, both the IPv4 and IPv6 network connections are online.

If you want an IPv6-only configuration, you must remove the IPv4 network.

Removing the IPv4 resources from the VCS configuration when VCS global clustering and VVR agents are configured

See "Removing the IPv4 resources from the VCS configuration when VCS global clustering and VVR agents are not configured" on page 531.

About migrating to IPv6 when VCS global clustering and VVR agents are configured in the presence of a bunker

To migrate to IPv6, do the following:

- Understand the current configuration.
- Meet the prerequisites for migration.
- Perform the actual migration.

Understanding the current IPv4 configuration when VCS global clustering and VVR agents are configured in the presence of a bunker

VVR is configured between the Primary site and the Secondary site and between the Primary site and the bunker site using the IPv4 network. VCS global clustering and VVR agents are configured on the Primary site and the Secondary site.

- The VCS global clustering and VVR agents are configured using IPV4 addresses.
- There are three sites: the Primary site, the Secondary site, and the bunker site.
- Replication is in progress from the Primary to the Secondary and from the Primary to the bunker using IPv4 addresses.

When the VVR global clustering agents are configured, the `main.cf` file on the Primary site looks similar to the following:

```
cluster sfha_site1 (
        ClusterAddress = "10.209.87.170"
        SecureClus = 1
        )
```

```
remotecluster sfha_site2 (
        ClusterAddress = "10.209.87.171"
        )

heartbeat Icmp (
        ClusterList = { sfha_site2 }
        Arguments @sfha_site2 = { "10.209.87.171" }
        )

system swlx25 (
        )

system swlx26 (
        )

group App_Group (
        SystemList = { swlx25 = 0, swlx26 = 1 }
        ClusterList = { sfha_site1 = 1, sfha_site2 = 2 }
        Authority = 1
        AutoStartList = { swlx25, swlx26 }
        ClusterFailOverPolicy = Manual
        )

        Mount res_mnt (
                MountPoint = "/hr_mnt"
                BlockDevice = "/dev/vx/dsk/hrdg/hr_dv01"
                FSType = vxfs
                FsckOpt = "-n"
                )

        RVGPrimary Rvg_Pri (
                RvgResourceName = hr_rvg
                )

        requires group VVRGRP online local hard
        res_mnt requires Rvg_Pri


        // resource dependency tree
        //
        //      group App_Group
        //      {
```

```
        //      Mount res_mnt
        //          {
        //          RVGPrimary Rvg_Pri
        //          }
        //      }


group ClusterService (
        SystemList = { swlx25 = 0, swlx26 = 1 }
        AutoStartList = { swlx25, swlx26 }
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
        )

        Application wac (
                StartProgram = "/opt/VRTSvcs/bin/wacstart"
                StopProgram = "/opt/VRTSvcs/bin/wacstop"
                MonitorProcesses = { "/opt/VRTSvcs/bin/wac" }
                RestartLimit = 3
                )

        IP gcoip (
                Device = eth0
                Address = "10.209.87.206"
                NetMask = "255.255.252.0"
                )

        NIC gconic (
                Device = eth0
                NetworkHosts = { "10.209.85.3" }
                )

        gcoip requires gconic
        wac requires gcoip


        // resource dependency tree
        //
        //      group ClusterService
        //      {
        //      Application wac
        //          {
        //          IP gcoip
```

```
        //              {
        //              NIC gconic
        //              }
        //          }
        //      }


group VVRGRP (
        SystemList = { swlx25 = 0, swlx26 = 1 }
        AutoStartList = { swlx25, swlx26 }
        )

        DiskGroup res_dg (
                DiskGroup = hrdg
                )

        IP ipres (
                Device = eth0
                Address = "10.209.87.170"
                NetMask = "255.255.252.0"
                )

        NIC nicres (
                Device = eth0
                NetworkHosts = { "10.209.84.1" }
                )

        RVG res_rvg (
                RVG = hr_rvg
                DiskGroup = hrdg
                )

        ipres requires nicres
        res_rvg requires res_dg
        res_rvg requires ipres


        // resource dependency tree
        //
        //      group VVRGRP
        //      {
        //      RVG res_rvg
        //          {
```

```
//          DiskGroup res_dg
//          IP ipres
//              {
//              NIC nicres
//              }
//          }
//      }
```

Migration prerequisites when VCS global clustering and VVR agents are configured in the presence of a bunker

Before you start the migration, meet the following prerequisites:

- Configure IPv6 addresses on the host systems. The systems would be in dual mode; that is, they are using both IPv4 and IPv6 addresses.
- Modify the `/etc/resolv.conf` file so that it includes both IPv4 and IPv6 DNS servers and domain names.
- If DNS servers are not available, then add the IPv6 entries in the `/etc/hosts` file.
- The hostnames of virtual IPs used for both IPv4 and IPv6 for VCS global clustering and RVG service groups should also be the same.

Migrating to IPv6 when VCS global clustering and VVR agents are configured in the presence of a bunker

To migrate the network to IPv6, do the following:

- Migrate the VCS global clustering service group to IPv6.
- Add the IP and NIC resources for IPv6 addresses in the RVG agent group.
- Migrate the VVR RLINKs from IPv4 to IPv6.
- Remove IPv4 resources from the VCS configuration.

Migrating the VCS global clustering service group to IPv6 when VCS global clustering and VVR agents are configured in the presence of a bunker

To migrate the VCS global clustering service to IPv6, do the following:

- Take the ClusterService Group offline on the Primary and Secondary sites.
- Edit the VCS configuration for IPv6 addresses.
- Bring the ClusterService Group online on the Primary and Secondary sites.

To take the ClusterService Group offline on the Primary and Secondary sites

1. On the Primary site, enter the following:

```
# hagrp -offline -force ClusterService -sys swlx25
```

2. On the Secondary site, enter the following:

```
# hagrp -offline -force ClusterService -sys swlx27
```

To edit the VCS configuration for IPv6 addresses

1. Make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

2. Modify the following parameters:

```
haclus -modify ClusterAddress \
  fd4b:454e:205a:111:210:20ff:fe98:0206 -clus sfha_site1
# haclus -modify ClusterAddress \
  fd4b:454e:205a:111:210:20ff:fe98:0205 -clus sfha_site2
# hahb -modify Icmp Arguments \
  fd4b:454e:205a:111:210:20ff:fe98:0205 -clus sfha_site2
# hares -modify gcoip Device eth2
# hares -modify gcoip Address \
  fd4b:454e:205a:111:210:20ff:fe98:0206
# hares -modify gcoip NetMask ""
# hares -modify gcoip PrefixLen 64
# hares -modify gconic Device eth2
# hares -modify gconic NetworkHosts \
  fd4b:454e:205a:111:211:43ff:fed3:f327
# hares -modify gconic Enabled 1
```

3. Save your configuration changes. Enter the following:

```
# haconf -dump -makero
```

4. On the Secondary site, modify the global clustering ClusterServer Group attributes.

To bring the ClusterService Group online on the Primary and Secondary sites

1. On the Primary site, enter the following:

```
# hagrp -online -force ClusterService -sys swlx25
```

2 On the Secondary site, enter the following:

```
# hagrp -online -force ClusterService -sys swlx27
```

The global clustering ClusterService Group in the VCS `main.cf` configuration file now looks similar to the following:

```
cluster sfha_site1 (
        ClusterAddress = "fd4b:454e:205a:111:210:20ff:fe98:0206"
        SecureClus = 1
        )

remotecluster sfha_site2 (
        ClusterAddress = "fd4b:454e:205a:111:210:20ff:fe98:0205"
        )

heartbeat Icmp (
        ClusterList = { sfha_site2 }
        Arguments @sfha_site2 = { "fd4b:454e:205a:111:210:20ff:fe98:0205" }
        )

system swlx25 (
        )

system swlx26 (
        )

group ClusterService (
        SystemList = { swlx25 = 0, swlx26 = 1 }
        AutoStartList = { swlx25, swlx26 }
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
        )

        Application wac (
                StartProgram = "/opt/VRTSvcs/bin/wacstart"
                StopProgram = "/opt/VRTSvcs/bin/wacstop"
                MonitorProcesses = { "/opt/VRTSvcs/bin/wac" }
                RestartLimit = 3
                )

        IP gcoip (
                Device = eth2
                Address = "fd4b:454e:205a:111:210:20ff:fe98:0206"
```

```
        PrefixLen = 64
        )

NIC gconic (
        Device = eth2
        NetworkHosts = { "fd4b:454e:205a:111:211:43ff:fed3:f327" }

        )

gcoip requires gconic
wac requires gcoip


// resource dependency tree
//
//      group ClusterService
//      {
//      Application wac
//          {
//          IP gcoip
//              {
//              NIC gconic
//              }
//          }
//      }
```

Adding the IP and NIC resources for IPv6 addresses in the RVG agent group when VCS global clustering and VVR agents are configured in the presence of a bunker

Currently, replication between the Primary and Secondary sites uses an IPv4 network. Before you can migrate the network to IPv6, you must add the IP and NIC resources to VCS control. You can do this in one of the following ways:

- Add new IP and NIC resources for IPv6 in the RVG group.
- Modify the current IP and NIC resources in the RVG group.

The first method, adding new IP and NIC resources for IPv6, ensures high availability. For example, suppose you add resources for IP and NIC resources for IPv6 addresses in the `main.cf` on the node in the Primary site from where replication is active. If the Primary site crashes before VVR migration to IPv6 from IPv4, the application service group fails over to the other node in the Primary site and replication continues using the IPv4 address. The existing IP and NIC resources are still in the `main.cf` file.

The second method, modifying the current IP and NIC resources, hampers replication. If a node in the Primary site crashes as you modify the existing IP resource, the original IPv4 virtual IP is lost. The application service group fails over to the other node, but the IPv4 IP address does not come online on the other node because you changed the IP resource from an IPv4 address to an IPv6 address. Instead, the IPv6 IP comes online. However, VVR is not configured use the IPv6 IP for replication, and replication pauses or stops. The IPv4 IP is not available on the other node.

To add the IP and NIC resources on the Primary and Secondary sites

1. Make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

2. Add the NIC resource for the IPv6 address and configure the related attributes:

```
# hares -add nicres_v6 NIC VVRGRP
VCS NOTICE V-16-1-10242 Resource added.
Enabled attribute must be set before agent monitors
# hares -modify nicres_v6 Enabled 1
# hares -modify nicres_v6 Device eth2
# hares -modify nicres_v6 NetworkHosts \
  fd4b:454e:205a:111:211:43ff:feaa:af71
# hares -probe nicres_v6 -sys swlx25
# hares -probe nicres_v6 -sys swlx26
```

3. Add the IP resource for the IPv6 address and configure the related attributes:

```
# hares -add ipres_v6 IP VVRGRP
VCS NOTICE V-16-1-10242 Resource added.
Enabled attribute must be set before agent monitors
# hares -modify ipres_v6 Enabled 1
# hares -modify ipres_v6 Device eth2
# hares -modify ipres_v6 Address \
  fd4b:454e:205a:111:211:43ff:feaa:af70
# hares -modify ipres_v6 PrefixLen 64
# hares -probe ipres_v6 -sys swlx25
# hares -probe ipres_v6 -sys swlx26
# hares -online ipres_v6 -sys swlx25
# hares -link ipres_v6 nicres_v6
# hares -link res_rvg ipres_v6
```

4 Save your configuration changes. Enter the following:

```
# haconf -dump -makero
```

5 After you add the IP and NIC resources to the Primary and Secondary sites, display the RVG agent group in the `main.cf` file to verify your changes.

```
group VVRGRP (
        SystemList = { swlx25 = 0, swlx26 = 1 }
        AutoStartList = { swlx25, swlx26 }
        )

        DiskGroup res_dg (
                DiskGroup = hrdg
                )

        IP ipres_v6 (
                Device = eth1
                Address = "fd4b:454e:205a:111:211:43ff:feaa:af70"
                PrefixLen = 64
                )

        IP ipres (
                Device = eth0
                Address = "10.209.87.170"
                NetMask = "255.255.252.0"
                )

        NIC nicres_v6 (
                Device = eth1
                NetworkHosts = { "fd4b:454e:205a:111:211:43ff:feaa:af71" }

                )

        NIC nicres (
                Device = eth0
                NetworkHosts = { "10.209.121.1" }
                )

        RVG res_rvg (
                RVG = hr_rvg
                DiskGroup = hrdg
                )

        ipres_v6 requires nicres_v6
        ipres requires nicres
        res_rvg requires ipres_v6
```

```
res_rvg requires res_dg
res_rvg requires ipres
```

6 Verify the new network configuration. Enter the following:

```
# ifconfig -a
```

7 Display a summary of all the HA groups. Enter the following:

```
# hastatus -sum
```

8 Add the IP and NIC resources in the VCS configuration on the Secondary site. Use a different IP address on the Secondary site for IPv6 resources. For example, the IPv4 virtual IP address could be `10.209.87.171` and the IPv6 virtual IP address could be `fd4b:454e:205a:111:211:43ff:feaa:af71`.

To add the IP and NIC resources for IPv6 in the VCS configuration on the bunker site

1 Optionally, display information on your current configuration.

- To display a summary of the HA groups on the bunker site, enter the following:

```
# hastatus -sum
```

- To display the current network configuration on the bunker site, enter the following:

```
# ifconfig -a
```

- Display the Service Group of the virtual IP on the bunker site. The following is a sample configuration:

```
group VVRGRP (
        SystemList = { swlx29 = 0 }
        AutoStartList = { swlx29 }
        )

        IP ipres (
                Device = eth0
                Address = "10.209.87.202"
                NetMask = "255.255.252.0"
                )

        NIC nicres (
```

```
        Device = eth0
        NetworkHosts = { "10.209.84.1" }
        )

ipres requires nicres


// resource dependency tree
//
//      group VVRGRP
//      {
//      IP ipres
//          {
//          NIC nicres
//          }
//      }
```

2 Make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

3 Add the NIC resource for the IPv6 address and configure the related attributes. Enter the following:

```
# hares -add nicres_v6 NIC VVRGRP
VCS NOTICE V-16-1-10242 Resource added.
Enabled attribute must be set before agent monitors
# hares -modify nicres_v6 Enabled 1
# hares -modify nicres_v6 Device eth1
# hares -modify nicres_v6 NetworkHosts \
  fd4b:454e:205a:111:211:43ff:feaa:af28
# hares -probe nicres_v6 -sys swlx29
# hares -link ipres_v6 nicres_v6
# hares -online ipres_v6 -sys swlx29
```

4 Add the IP resource for the IPv6 address and configure the related attributes. Enter the following:

```
# hares -add ipres_v6 IP VVRGRP
VCS NOTICE V-16-1-10242 Resource added.
Enabled attribute must be set before agent monitors
# hares -modify ipres_v6 Enabled 1
# hares -modify ipres_v6 Device eth1
# hares -modify ipres_v6 Address \
  fd4b:454e:205a:111:211:43ff:feaa:af72
# hares -probe ipres_v6 -sys swlx29
```

5 Save your configuration changes. Enter the following:

```
# haconf -dump -makero
```

6 Verify your changes on the bunker site. To display the updated network configuration, enter the following:

```
# ifconfig -a
```

Migrating VVR RLINKs from IPv4 to IPv6 when VCS global clustering and VVR agents are configured in the presence of a bunker

Before you perform the steps in this section, make sure all sites are in dual mode; that it, all sites have both IPv4 and IPv6 addresses defined.

To migrate VVR RLINKs from IPv4 to IPv6, do the following:

- Migrate the VVR RLINKs of the Primary and Secondary sites to IPv6.
- Migrate the VVR RLINKs of the Primary and bunker sites to IPv6.
- Manually edit the VVR RLINKs at the bunker and Secondary sites. You must edit the RLINKs manually because they are inactive and the `vradmin changeip` command cannot change the IP for these rlinks

To migrate the RLINKs of the Primary and Secondary sites to IPv6

1 On the Primary site node, run the `vradmin changeip` command.

```
# vradmin -g hrdg changeip hr_rvg 10.209.87.171 \
  newpri=fd4b:454e:205a:111:211:43ff:feaa:af70 \
  newsec=fd4b:454e:205a:111:211:43ff:feaa:af71
Message from Primary:
VxVM VVR vxrlink INFO V-5-1-3614 Secondary data volumes detected with rvg hr_rvg as parent:
VxVM VVR vxrlink INFO V-5-1-6183 datavol:      len=10485760          primary_datavol=datavol
```

2 Check the replication status. Enter the following:

```
# vradmin -g hrdg repstatus hr_rvg
Replicated Data Set: hr_rvg
Primary:
  Host name:                  10.209.87.170
  RVG name:                   hr_rvg
  DG name:                    hrdg
  RVG state:                  enabled for I/O
  Data volumes:               1
  VSets:                      0
  SRL name:                   hr_srl
  SRL size:                   1.00 G
  Total secondaries:          2

Bunker (secondary):
  Host name:                  10.209.87.202
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Data status:                consistent, behind
  Replication status:         replicating (connected)
  Current mode:               synchronous
  Logging to:                 SRL (48 Kbytes behind, 0% full)
  Timestamp Information:      behind by 0h 0m 0s

Secondary:
  Host name:                  fd4b:454e:205a:111:211:43ff:feaa:af71
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Data status:                consistent, behind
  Replication status:         replicating (connected)
  Current mode:               asynchronous
  Logging to:                 SRL (341972 Kbytes behind, 32% full)
  Timestamp Information:      behind by 0h 1m 10s
```

To migrate the RLINKs of the Primary and bunker sites to IPv6

1. On the Primary site node, run the `vradmin changeip` command.

```
# vradmin -g hrdg changeip hr_rvg 10.209.87.202 \
  newpri=fd4b:454e:205a:111:211:43ff:feaa:af70 \
  newsec=fd4b:454e:205a:111:210:20ff:fe98:af72
Message from Primary:
VxVM VVR vxrlink INFO V-5-1-12348 Secondary srl detected with rvg hr_rvg as parent:
```

2 Check the replication status. Enter the following:

```
# vradmin -g hrdg repstatus hr_rvg

Replicated Data Set: hr_rvg
Primary:
  Host name:                  fd4b:454e:205a:111:211:43ff:feaa:af70
  RVG name:                   hr_rvg
  DG name:                    hrdg
  RVG state:                  enabled for I/O
  Data volumes:               1
  VSets:                      0
  SRL name:                   hr_srl
  SRL size:                   1.00 G
  Total secondaries:          2

Bunker (secondary):
  Host name:                  fd4b:454e:205a:111:210:20ff:fe98:af72
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Data status:                consistent, up-to-date
  Replication status:         replicating (connected)
  Current mode:               synchronous
  Logging to:                 SRL
  Timestamp Information:      behind by 0h 0m 0s

Secondary:
  Host name:                  fd4b:454e:205a:111:211:43ff:feaa:af71
  RVG name:                   hr_rvg
  DG name:                    hrdg
  Data status:                consistent, behind
  Replication status:         replicating (connected)
  Current mode:               asynchronous
  Logging to:                 SRL (752444 Kbytes behind, 71% full)
  Timestamp Information:      behind by 0h 5m 4s
```

3 After you migrate to IPv6, check the RLINKs on the Primary site. Enter the following:

```
# vxprint -Pl
Disk group: hrdg

Rlink:    rlk_10.209.87.202_hr_rvg
info:     timeout=500 rid=0.1077
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=none
state:    state=ACTIVE
          synchronous=override latencyprot=off srlprot=off
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:211:43ff:feaa:af72
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af72 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276521227.64.swlx29
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.203_hr_rvg
          remote_rlink_rid=0.1048
          local_host=fd4b:454e:205a:111:210:20ff:fe98:af70
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af70 port=4145
protocol: TCP/IP
flags:    write enabled attached consistent connected bunker synchronous

Rlink:    rlk_10.209.87.204_hr_rvg
info:     timeout=500 rid=0.1073
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=10240kbps
state:    state=ACTIVE
          synchronous=off latencyprot=off srlprot=autodcm
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:211:43ff:feaa:af71
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af71 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276621272.66.swlx27
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.203_hr_rvg
          remote_rlink_rid=0.1068
          local_host=fd4b:454e:205a:111:211:43ff:feaa:af70
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:203 port=4145
protocol: TCP/IP
flags:    write enabled attached consistent connected asynchronous
```

To change the RLINKs on the bunker and Secondary sites to use IPv6 addresses

1 Check the RLINKs on the bunker site. Enter the following:

```
# vxprint -Pl
Disk group: hrdg

Rlink:    rlk_10.209.87.203_hr_rvg
info:     timeout=500 rid=0.1048
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=none
state:    state=ACTIVE
          synchronous=override latencyprot=off srlprot=off
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:211:43ff:feaa:af70
          IP_addr=fd4b:454e:205a:111:211:43ff:feaa:af70 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276518249.76.swlx25
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.202_hr_rvg
          remote_rlink_rid=0.1077
          local_host=fd4b:454e:205a:111:210:20ff:fe98:af72
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af72 port=4145
protocol: TCP/IP
flags:    write enabled attached consistent connected bunker

Rlink:    rlk_10.209.87.204_hr_rvg
info:     timeout=500 rid=0.1052
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=none
state:    state=STALE
          synchronous=override latencyprot=off srlprot=off
assoc:    rvg=hr_rvg
          remote_host=10.209.87.171 IP_addr=10.209.87.171 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276621272.66.swlx27
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.202_hr_rvg
          remote_rlink_rid=0.1075
          local_host=10.209.87.202 IP_addr=10.209.87.202 port=4145
protocol: TCP/IP
flags:    write enabled detached consistent disconnected bunker
```

2 Check the RLINKs on the Secondary site. Enter the following:

```
# vxprint -Pl
Disk group: hrdg

Rlink:    rlk_10.209.87.202_hr_rvg
info:     timeout=500 rid=0.1075
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=none
state:    state=STALE
          synchronous=off latencyprot=off srlprot=autodcm
assoc:    rvg=hr_rvg
          remote_host=10.209.87.202 IP_addr=10.209.87.202 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276521227.64.swlx29
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.204_hr_rvg
          remote_rlink_rid=0.1052
          local_host=10.209.87.204 IP_addr=10.209.87.204 port=
protocol: TCP/IP
flags:    write enabled detached consistent disconnected bunker_target

Rlink:    rlk_10.209.87.203_hr_rvg
info:     timeout=500 rid=0.1068
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=10240kbps
state:    state=ACTIVE
          synchronous=off latencyprot=off srlprot=autodcm
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:210:20ff:fe98:af70
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af70 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276518249.76.swlx25
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.204_hr_rvg
          remote_rlink_rid=0.1073
          local_host=fd4b:454e:205a:111:210:20ff:fe98:af71
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af71 port=4145
protocol: TCP/IP
flags:    write enabled attached consistent connected
```

3 On the bunker and Secondary sites, set the IPv6 addresses manually.

On the bunker site, enter the following:

```
# vxedit -g hrdg \
  set local_host=fd4b:454e:205a:111:210:20ff:fe98:af72 \
  remote_host=fd4b:454e:205a:111:210:20ff:fe98:af71 \
  rlk_10.209.87.204_hr_rvg
```

On the Secondary site, enter the following:

```
# vxedit -g hrdg \
  set local_host=fd4b:454e:205a:111:210:20ff:fe98:af71 \
  remote_host=fd4b:454e:205a:111:210:20ff:fe98:af72 \
  rlk_10.209.87.202_hr_rvg
```

4 Check the RLINKs on the bunker site to verify your changes:

```
# vxprint -Pl
Disk group: hrdg

Rlink:    rlk_10.209.87.203_hr_rvg
info:     timeout=500 rid=0.1048
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=none
state:    state=ACTIVE
          synchronous=override latencyprot=off srlprot=off
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:210:20ff:fe98:af70
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af70 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276518249.76.swlx25
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.202_hr_rvg
          remote_rlink_rid=0.1077
          local_host=fd4b:454e:205a:111:210:20ff:fe98:af72
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af72 port=4145
protocol: TCP/IP
flags:    write enabled attached consistent connected bunker

Rlink:    rlk_10.209.87.204_hr_rvg
info:     timeout=500 rid=0.1052
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=none
state:    state=STALE
          synchronous=override latencyprot=off srlprot=off
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:210:20ff:fe98:af71
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af71 port=
          remote_dg=hrdg
          remote_dg_dgid=1276621272.66.swlx27
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.202_hr_rvg
          remote_rlink_rid=0.0
          local_host=fd4b:454e:205a:111:210:20ff:fe98:af72
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af72 port=4145
protocol: TCP/IP
flags:    write enabled detached consistent disconnected bunker
```

5 Check the RLINKs on the Secondary site. Enter the following:

```
# vxprint -Pl
Disk group: hrdg

Rlink:    rlk_10.209.87.202_hr_rvg
info:     timeout=500 rid=0.1075
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=none
state:    state=STALE
          synchronous=off latencyprot=off srlprot=autodcm
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:210:20ff:fe98:af72
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af72 port=
          remote_dg=hrdg
          remote_dg_dgid=1276521227.64.swlx29
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.204_hr_rvg
          remote_rlink_rid=0.0
          local_host=fd4b:454e:205a:111:210:20ff:fe98:af71
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af71 port=4145
protocol: TCP/IP
flags:    write enabled detached consistent disconnected bunker_target

Rlink:    rlk_10.209.87.203_hr_rvg
info:     timeout=500 rid=0.1068
          latency_high_mark=10000 latency_low_mark=9950
          bandwidth_limit=10240kbps
state:    state=ACTIVE
          synchronous=off latencyprot=off srlprot=autodcm
assoc:    rvg=hr_rvg
          remote_host=fd4b:454e:205a:111:210:20ff:fe98:af70
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af70 port=4145
          remote_dg=hrdg
          remote_dg_dgid=1276518249.76.swlx25
          remote_rvg_version=30
          remote_rlink=rlk_10.209.87.204_hr_rvg
          remote_rlink_rid=0.1073
          local_host=fd4b:454e:205a:111:210:20ff:fe98:af71
          IP_addr=fd4b:454e:205a:111:210:20ff:fe98:af71 port=4145
protocol: TCP/IP
flags:    write enabled attached consistent connected
```

The replication IPs have been changed to IPv6 on all the Sites. Now the Sites can be moved to pure IPv6 by removing the IPv4 network.

Removing the IPv4 resources from the VCS configuration when VCS global clustering and VVR agents are configured in the presence of a bunker

To remove the IPv4 resources from the VCS configuration, do the following:

- Remove the IPv4 resources from the Primary and Secondary sites.
- Remove the IPv4 resources from the VCS configuration on the bunker site.
- Remove the IPv4 network. This task is optional. Only perform it if you want to have an IPv6-only environment.

To remove the IPv4 resources from the Primary and Secondary sites

1. On the Primary site, make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

2. Take the system offline. Enter the following:

```
# hares -offline ipres -sys swlx25
```

3. Unlink the RVG and IP resources. Enter the following:

```
# hares -unlink res_rvg ipres
```

4. Unlink the IP and NIC resources. Enter the following:

```
# hares -unlink ipres nicres
```

5. Delete the IP and NIC resources. Enter the following:

```
# hares -delete ipres
# hares -delete nicres
```

6. Save the changes made to the configuration. Enter the following:

```
# haconf -dump -makero
```

7 Verify that only IPv6 addresses are defined.

```
# hares -state | grep -i res
#Resource    Attribute        System     Value
ipres_v6     State            swlx25     ONLINE
ipres_v6     State            swlx26     OFFLINE
nicres_v6    State            swlx25     ONLINE
nicres_v6    State            swlx26     ONLINE
```

8 Repeat this procedure on the Secondary site to remove its IPv4 resources.

To remove the IPv4 resources from the VCS configuration on the bunker site

1 Enter the following command:

```
# hares -state
#Resource    Attribute             System     Value
ipres        State                 swlx29     ONLINE
ipres_v6     State                 swlx29     ONLINE
nicres       State                 swlx29     ONLINE
nicres_v6    State                 swlx29     ONLINE
phantom_vxss State                 swlx29     ONLINE
vxatd        State                 swlx29     ONLINE
```

2 Make the VCS configuration writeable. Enter the following:

```
# haconf -makerw
```

3 Unlink the IP and NIC resources. Enter the following:

```
# hares -unlink ipres nicres
```

4 Delete the IP and NIC resources. Enter the following:

```
# hares -delete ipres
# hares -delete nicres
```

5 Save the changes made to the configuration. Enter the following:

```
# haconf -dump -makero
```

6 Verify that only IPv6 addresses are defined.

```
# hares -state
#Resource    Attribute             System     Value
ipres_v6     State                 swlx29     ONLINE
nicres_v6    State                 swlx29     ONLINE
phantom_vxss State                 swlx29     ONLINE
vxatd        State                 swlx29     ONLINE
```

7 Display the current network status. Enter the following:

```
# ifconfig -a
```

To remove the IPv4 network on the Primary and Secondary sites (optional)

1 Down the interface on which IPv4 was configured.

```
# ifdown eth0
```

2 Remove the IPv4 network cable. This does not impact replication because the VVR configuration has been migrated to use IPv6 on all sites.

3 Verify that there are no IPv4 addresses defined.

```
# ifconfig -a
```

4 Remove the IPv4 entries from `/etc/hosts` and `/etc/resolv.conf` files on all the hosts on both the Primary and Secondary sites.

Note: Do not remove the IPv4 loopback entry from the `/etc/hosts` file. If you remove the loopback entry, and replication is affected, stop and restart the VVR service using the `vxstart_vvr stop` and `vxstart_vvr start` commands.

You have now migrated VVR to an IPv6-only network.

Appendix F

Sample main.cf files

This appendix includes the following topics:

- Globally clustered VCS and VVR main.cf

Globally clustered VCS and VVR main.cf

The following main.cf file applies to a globally clustered VCS and VVR configuration. It is common for the elect-primary feature and for use with bunker with the global cluster options with VVR.

```
include "OracleASMTypes.cf"
include "types.cf"
include "Db2udbTypes.cf"
include "OracleTypes.cf"
include "SybaseTypes.cf"

cluster gco2 (
        UserNames = { admin = dqrJqlQnrMrrPzrLqo }
        ClusterAddress = "10.182.44.221"
        Administrators = { admin }
        )

remotecluster gco1 (
        ClusterAddress = "10.182.71.20"
        )

heartbeat Icmp (
        ClusterList = { gco1 }
        StopTimeout @gco1 = 60
        AYATimeout @gco1 = 300
        AYARetryLimit = 1
```

```
        Arguments @gco1 = { "10.182.71.20" }
        )

system msdn15 (
        )

system msdn16 (
        )

group ClusterService (
        SystemList = { msdn15 = 0, msdn16 = 1 }
        AutoStartList = { msdn15, msdn16 }
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
        )

        Application wac (
                StartProgram = "/opt/VRTSvcs/bin/wacstart"
                StopProgram = "/opt/VRTSvcs/bin/wacstop"
                MonitorProcesses = { "/opt/VRTSvcs/bin/wac" }
                RestartLimit = 3
                )

        IP webip (
                Device = eth0
                Address = "10.182.44.221"
                NetMask = "255.255.240.0"
                )

        NIC csgnic (
                Device = eth0
                )

        wac requires webip
        webip requires csgnic


        // resource dependency tree
        //
        //        group ClusterService
        //        {
        //        Application wac
        //            {
```

```
        //            IP webip
        //                {
        //                NIC csgnic
        //                }
        //            }
        //        }


group VVRGrp (
        SystemList = { msdn15 = 0, msdn16 = 1 }
        AutoStartList = { msdn15, msdn16 }
        )

        DiskGroup Hr_Dg (
                DiskGroup = dg1
                )

        IP vvrip (
                Device = eth0
                Address = "10.182.44.220"
                NetMask = "255.255.240.0"
                )

        NIC vvrnic (
                Device = eth0
                )

        RVG Hr_Rvg (
                RVG = rvg1
                DiskGroup = dg1
                )

        Hr_Rvg requires Hr_Dg
        Hr_Rvg requires vvrip
        vvrip requires vvrnic


        // resource dependency tree
        //
        //        group VVRGrp
        //        {
        //        RVG Hr_Rvg
        //            {
```

```
        //            DiskGroup Hr_Dg
        //            IP vvrip
        //                {
        //                NIC vvrnic
        //                }
        //            }
        //        }


group app_rep (
        SystemList = { msdn16 = 0, msdn15 = 1 }
        ClusterList = { gco1 = 1, gco2 = 0 }
        AutoStartList = { msdn16, msdn15 }
        ClusterFailOverPolicy = Auto
        )

        FileOnOff test-res (
                PathName = "/var/tmp/test"
                )

        RVGPrimary rvg_prim_res (
                RvgResourceName = Hr_Rvg
                AutoResync = 1
                )

        requires group VVRGrp online local hard
        test-res requires rvg_prim_res


        // resource dependency tree
        //
        //        group app_rep
        //        {
        //        FileOnOff test-res
        //            {
        //            RVGPrimary rvg_prim_res
        //            }
        //        }


group app_rep_fd (
        SystemList = { msdn16 = 0, msdn15 = 1 }
        UserStrGlobal = "FD:app_rep"
```

```
AutoFailOver = 0
TypeDependencies = { IP, Mount, RVGSnapshot }
)

IP webip_fd (
        Device = eth0
        Address = "10.182.44.221"
        NetMask = "255.255.240.0"
        )

RVGSnapshot Hr_Rvg-sos (
        RvgResourceName = Hr_Rvg
        CacheObj = cacheobj
        Prefix = snap
        )

requires group VVRGrp online local firm


// resource dependency tree
//
//        group app_rep_fd
//        {
//        RVGSnapshot Hr_Rvg-sos
//        IP webip_fd
//        }
```

Index

A

acting Secondary 318
activating a bunker
    using command line 347
Add or Remove Programs option 463
adding a data volume
    using command line 196
adding a Secondary
    best practices 141
    using command line 139
agents. *See* individual agents
    best practices for setting up 108
    configuring 118
    configuring when VCS is stopped 124
    list of VVR 28
    setting up
        best practices 108
analysis results
    recalculating 455
    saving 459
    viewing from file 459
analyzing data
    overview 447
    performing what-if analysis 456
application writes
    displaying 455
application-defined control messages 489
applications
    characteristics 65
    defined 63
    on Secondary 299
    write rates 65
associating a data volume
    using command line 196
associating DCM to a data volume
    using command line 205
associating SRL to a Primary RVG
    using command line 136
asynchronous mode
    considerations 66
asynchronous replication
    data flow 34
    explained 28, 88
    replication using a bunker site 334
    setting up 145
asynchronous replication with secondary logging 35
    bulk transfer 36
    data flow 37
asynchronous writes in VVR 33
autodcm
    srlprot setting 91, 148
autofb option
    takeover command 312
automated failover
    using command line 352
automatic fast failback
    using command line 312
Automatic Synchronization
    synchronizing Secondary example 156
automating firedrill 272
autosync flag, definition 172

B

backup
    block-level tape 153, 505
    primary Storage Checkpoint 375
    using snapshot and vradmin ibc
        example 296
    using snapshot and vxibc
        example 485
backup constraints
    Secondary 80
backup interval
    see Secondary outage 458
balancing RPO with RTO
    replication using a bunker site 337
bandwidth
    network 72
    required for synchronous replication 455

bandwidth limit
controlling
used for replication. *See* using command line
used to synchronize volumes. *See* using command line
Bandwidth Throttling 95, 150
See also bandwidth limit
about 95, 150
disabling
using command line 151
bandwidth used by VVR
determining 182
best practices
setting up VVR agents 108
block size
DCM replay 390
block-level tape backup 153, 505
breaking off mirrors 221
buffer space
Primary 381
Secondary 382, 388
tunable parameters 382
VVR 381
bunker commands
list 354
bunker configuration
using command line 340
bunker replay
using command line 349
bunker Secondary
activating
using command line 347
administering
using command line 347
deactivating
using command line 347
using command line 344
bunker setup
using command line 351
bunker site
see Primary SRL 335
bunker SRL
using synchronous mode 335
business needs 64

C

cache object 219, 321
can_sync 501
cant_sync 501
changing IP addresses 229
changing network ports for replication 236
changing replication settings 227
checkdelete
vxrlink keyword 245
vxrvg keyword 245
checkend
vxrvg keyword 472
checkstart
vxrvg keyword 472
chunk size
memory tunable 390
clusters
cluster-shareable disk groups 41
collecting data
on UNIX 436
on Windows 444
commands 466
See also individual command names
reference 466
comparing
snapshot methods 60
component volumes of a volume set, associating to RDSs 201
configuration
sample 338
setting up the VVR 115
using command line 344
configuration errors
recovery from 359
configuring RVG agent
when VCS is started 119
when VCS is stopped 124
configuring VVR
in a firewall environment 75
introduction 63
configuring VVR in VCS environment
overview 100
requirements 108
consistency group
associating 407
creating 407
delete 421
disassociating 416
modifying 416
consistent data
concept 22
consistent RLINK 282

constraints
peak usage 78
Secondary backup 80
Secondary downtime 81
synchronization period 79
converting original Primary to Secondary
using command line 302
cplist
vxrlink keyword 176
vxrvg keyword 176
creating
cache object 254
instant plex-breakoff snapshot 258
instant snapshot 246
snapshot plexes 257
snapshot volumes 248
snapshots
overview 245
space-optimized snapshot 252
creating a Primary RVG
using command line 136
creating Replicated Data Set
using command line 135
creating RVGs
using vradmin vs. vxmake 85
current data
concept 22

D

DAS Direct Attached Storage 335
data
consistent vs. up-to-date 22
data analysis results
recalculating 455
saving 459
viewing from file 459
viewing in VRAdvisor 454
data analyzing
overview 447
performing what-if analysis 456
Data Change Map
see DCM 25
data collection
on UNIX 436
on Windows 444
data differences, see data verification 273
data flow
VVR asynchronous mode 34
data flow *(continued)*
VVR asynchronous mode with secondary logging 37
VVR synchronous mode 38
data flow in VVR 32
data loss
and takeover 311
calculating bandwidth based on 458
data on Secondary
using for off-host processing 284
data verification
concept 273
performing 273
data volume errors
missing 363
name mismatch 365
resize 364, 366
data volumes
administering
using command line 195
associating
using command line 196
dissociating or removing
using command line 211–212
mapping Secondary name 214
renaming
using command line 211
resizing
using command line 208
verifying data
about 273
before adding volumes to RDS 198
using offline verification 275
using online verification 273
date formats
supported 437
DCM
and fast failback 312
associating to a data volume 205
concept 25
required for takeover 311
dcm
srlprot setting
about 91
changing 148
DCM playback
after SRL overflow 218
during failback 312, 318
DCM replay block size 390

DCM size 205
dcm_logging flag, definition 219
deactivating a bunker
    using command line 347
Decision Support
    example 294, 483
deport bunker
    using command line 343
detach bunker
    using command line 346
difference-based synchronization
    about 509
    failing back using 323
disabling Bandwidth Throttling 151
disaster recovery
    failed Primary 308
    using a bunker 336, 347
    verifying readiness 54, 271
Disk Group Split and Join
    synchronizing Secondary using 507
disk groups
    displaying disk group ID 142
    shared
        replicating in 41
disks
    choosing for the SRL 380
diskStats command
    collecting data 445
displaying
    data volumes in RDS 162
    RVG and RDS information 161
    Storage Checkpoints, list 176
dissociating data volumes
    using command line 211
downtime constraint
    Secondary 81
DR readiness
    verifying 271
DSS. *See* Decision Support
Dual-node / Dual-stack 22

E

errors
    configuration 359
    data volume size mismatch 364, 366
    during an initial RLINK attach 359
    during modification of an RVG 363
    missing data volume 363
errors *(continued)*
    name mismatch
        data volume 365–366
event notification
    using command line 192
event types 192
examples
    setting up VVR in a VCS environment 114
    using command line
        creating an RDS 512
        migrating Primary 304
        off-host processing 294, 483
        setting up replication 156
        synchronizing the Secondary 506
        taking over from failed Primary 313–314

F

fail
    latencyprot setting 94
    mode of replication 89
    srlprot setting 148
FAIL state 499, 501
failback logging
    about 312
    example 319
failed Primary
    example takeover 313–314
failing back
    using command line
        about 316, 350
        fast failback versus difference-based synchronization 317
        using difference-based synchronization 323
        using fast failback or failback logging 317
fast failback feature
    about 312
    enabling with vradmin takeover 312
    versus difference-based synchronization 317
features of VVR 20
file replication job
    creating 405
    deleting 420
    displaying information 409
    managing 408
    modifying 413
files
    analysis results 460
firedrill
    performing 272

firewalls
configuring VVR in 75
using VVR in 141
VVR and Network Address Translation 393
VVR ports in 237
flag settings
RLINK 172
RVG 162
flags and definitions
autosync 172
can_sync 501
cant_sync 501
Data Change Map 219
inconsistent 501
resync_active 219
resync_paused 173, 219
resync_started 173
SRL Overflow Protection With DCM 219
Flexible Storage Sharing 93
freezing replication
about 284
before taking snapshot 249
off-host processing 286, 288
using ibc messaging 478
full synchronization
about 502
example 504

G

generic VVR setup in a VCS environment 109
global mapping 214
graphs
displaying analysis results 454
GUIs, VVR
about 135

H

heartbeat timeout
defined 390
heartbeats
ports used for 238
high mark
latency
setting. *See* using command line
host names
changing 229
hybrid group
about 85

I

IBC Messaging
API 488
explanation 285
function 488
off-host processing
using for 285
overview 284, 478
vradmin ibc
using for off-host processing 289
vxibc
individual commands 479
In-Band Control Messaging
see IBC Messaging 285
inactive RLINKs 499
inconsistency
inconsistent flag 501
inconsistent RLINKs 500
RLINK during restores 282
initial VVR setup
using command line 511
installation wizard 462
installing VRAdvisor
on Solaris 462
on Windows 462
interfaces of VVR
about 135
IP addresses
changing 229
IPv4-only node 22
IPv6-enabled node 22
IPv6-only node 22

J

Japanese locales
supported 437

K

kernel buffers 32

L

latency low and high mark
setting
using command line 146
latency protection
about 93
choosing 70

latency protection *(continued)*
disabling
using command line 147
enabling
using command line 147
using command line 345
layered volumes
collecting data on 452
layout of SRL
effect on performance 379
list of agents for VVR 28
local host
definition 27
local RVG
represents RDS 28
locales
supported 436
log overflow
STALE 499
low mark
latency
setting. *See* using command line

M

make Secondary
using command line 323
master node
about 41
maximum bandwidth 72
maximum outage window 454
maximum transmission unit
network 77
memory chunk size 390
memory statistics
displaying 475
migrate
vradmin keyword 302
migrating Primary
example 304
overview 300
using command line 302
minimum time for collecting data 435
mirroring the SRL 380
mirrors
breaking off before synchronization 221
missing data volume error 363
modes of replication
and network performance 72
asynchronous 66
modes of replication *(continued)*
setting 145
synchronous 67
most up-to-date Secondary
identifying 189
Mount resource
volume sets 109
MTU
see maximum transmission unit 77
multiple Secondaries in RDS
migrating Primary with 303

N

network
planning the 72
Network Address Translation firewall 393
network bandwidth
choosing 78
peak usage 72
required for synchronous replication 455
network bandwidth used by VVR
determining 182
network maximum transmission unit 77
network performance and mode of replication 72
network performance data 178, 181–182, 186
network ports
changing 236
network ports used by VVR 74
network protocol 73
network requirements for replication using a bunker site
list 339
network transport protocol
attribute
STORAGE 149
TCP 149
UDP 149
planning 141
setting 149
notify
events 192

O

object states
VVR 496
off
latencyprot setting 94
mode of replication 89, 145

off *(continued)*
srlprot setting 148
off-host processing
backup example 485
Decision Support example 294, 483
examples 294, 483
IBC Messaging
using 285
tasks to perform 288
trial failover example 486
offline data verification
concept 273
using command line 275
onfreeze script
about 289
creating 291
onfreeze task
off-host processing 288
online backup 281
online data validation 273
operating systems
supported 461
original Primary
recovering using command line 349
restoring using command line 349
outage
calculating SRL size from 458
outage window
maximum 454
overflow protection
SRL 148
override
latencyprot setting 94
mode of replication 89
srlprot setting 91, 148
synchronous 345
overview
configuring VVR in a VCS environment 100

P

packet size
choosing 76
setting 149
parameters
VVR tunables 382
pause
vxrlink keyword 499
pause/resume
temporary states 501
pauserep
vradmin keyword 227
pausing replication
using command line 227
Pausing state 501
peak usage constraint 78
performance
and mode of replication 72
and SRL 379
performing
offline data validation 273
online data validation 273
performing firedrill 272
period of synchronization 79
permissible outage
calculating SRL size from 458
pkgadd command 462
pkgrm command 462
platforms
supported 461
ports
in a firewall 76
used by VVR 74
ports used by VVR 74
changing 236
preferences
modifying 458
Primary buffer space 381
Primary fails
disaster recovery 347
Primary host
concept 25
failure recovery 368
Primary role takeover
example 313–314
overview 308
Primary RVG
creating
using command line 136
removing
using command line 242
Primary SRL
error
cleanup after 371
header error 373
overflow recovery 372
Primary Storage Checkpoint
cleanup using 375
deleting 245

Primary Storage Checkpoint *(continued)*
restoring from 281
restoring Secondary from 500
primary_datavol field 214
printrvg, vradmin keyword 161
printvol, vradmin keyword 163
protocol
network 73, 141

R

RAID-5 volumes and VVR 137
RAID-5 volumes not supported 452
RDC
about 85
SystemZones attribute of RVG and RVG Primary agents 86
RDS
administering data volumes
using command line 195
associating a volume set 201
associating volumes
using command line 196
concept 25
creating
using command line 136
displaying data volumes in RDS
using command line 162
displaying information
using command line 161
displaying replication status
using command line 164
removing
using command line 242
removing or dissociating data volumes
using command line 211
removing Primary from
using command line 242
renaming data volume
using command line 211
resizing data volumes
using command line 208
resizing SRL
using command line 223
readback buffer space on the Primary
explained 382
tunable parameter 385
recalculating analysis results 455
RECOVER state 501
recovering from failed Primary 308
recovering RLINK 474
Recovery Point Objective (RPO) 311
Recovery Time Objective (RTO) 311
region size 205
reinitializing bunker
using command line 346
removing a bunker
using command line 354
removing data volume
using command line 212
removing Primary
using command line 242
removing Secondary
using command line 241
removing VRAdvisor
on Solaris 462
on Windows 463
replacement for Primary
finding most suitable 310
Replicated Data Cluster. *See* RDC
Replicated Data Set
see RDS 25
replication
freezing 249, 286
pausing 227
planning configuration 63
resuming 227
resuming after migration 305, 308
setting up 108
shared disk group 41
starting 152
status, displaying 164
stopping to Secondary 228
terms defined 21
unfreezing 252, 481
replication modes
considerations 66
replication parameters 70
replication settings
bandwidth limit 95
defaults 144
latency protection 146
overview 227
packet size 149
replication mode 88
SRL overflow protection 148
replication state
verifying 118

replication using a bunker site
settings
using command line 339
repstatus, vradmin keyword 164
required bandwidth
synchronous replication 455
required SRL size 455
requirements
configuring VVR in VCS environment 108
resizevol
vradmin keyword 208
resizing a file system 208
resizing data volumes
about 208
using command line 210
resizing SRL
using command line 223
restore
restore
vxrlink command 281
restore command 501
RESTORING state 501
Secondary RLINK 281
vxrlink keyword 474, 499
restoring bunker
using command line 351
restoring original Primary
using command line 350
results file
location 460
results graphs
described 454
resume
resume command 501
RESUMING state 501
vxrlink keyword 473
resumerep
vradmin keyword 227
resuming replication
after migration 305, 308
using command line 227
resync
resync_active flag
definition 219
resync_paused flag
definition 173, 219
resync_paused flag, definition 173, 219
resync_started flag, definition 173
resyncfromreplica
recovering failed Secondary data volumes 269
recovering from logical corruption of data 269
using snapback 268
RLINK
concept 24
configuring 144
creating 475
displaying status 171
dissociating from RVG 473
flag settings 172
inactive 498
STALE 373
states 498
status command 515
RPO
balancing RPO
replication using a bunker site 337
RTO
Recovery Time Objective
replication using a bunker site 337
RVG
associating to clear PASSTHRU 371
concept 23
displaying status 161
flag settings 162
set up automated failover 352
start 471
RVG agent
configuring when VCS is started 119
configuring when VCS is stopped 124
SystemZones attribute 86
virtual IP requirement 117
RVGLogowner agent
configuring 125
RVGPrimary agent
SystemZones attribute 86
RVGShared agent
configuring 125
RVGSharedPri agent
configuring 125

S

sample configuration
using command line 338
sample configuration files
to configure agent
location 115
saving analysis results 459

script
collecting data
on Solaris 440
Secondary 152
See also synchronizing Secondary
acting 318
adding 139
backing up 279
identifying most up-to-date 189
removing 241
removing from RDS 241
transferring applications to 299
using data 284
Secondary backup constraints 80
Secondary backup interval
see Secondary outage 458
Secondary buffer space 382
Secondary downtime constraint 81
Secondary host
concept 25
Secondary SRL
header error 376
volume error 375
Secondary Storage Checkpoint
backing up the Secondary 279
deleting 245
recovery using 374
secondary_log_err flag 376
secondary_paused flag 375
set
vradmin keyword 144
setting bandwidth limit
using command line 150
setting network transport protocol 149
setting packet size 149
setting up
replication 108
setting up the VVR configuration 115
setting up VVR agents
best practices 108
shared disk group
replicating in
concepts 41
size of DCM 205
size of SRL
determining 77
size of tunable
DCM replay blocks 390
memory chunks 390
size of tunable *(continued)*
packet 76
slave nodes
about 41
SmartMove for VVR 157
Turn off 157
snapback
vxrvg keyword 267, 472
snapback operation
about 267
snapprint
vxrvg keyword 472
snaprefresh
vxrvg keyword 472
snaprestore
vxrvg keyword 473
snapshot
vxrvg keyword 473
snapshot methods
overview
instant full 56
instant plex breakoff 57
instant space-optimized 56
snapshot plexes
creating 266
snapshots
before DCM replay 318
before takeover 311
comparing methods 60
overview 245
traditional method 55
understanding 54
using with fast failback 321
using with resynchronization 219
snapshots of RVG
creating
cache object 254
instant plex-breakoff 258
instant snapshot 246
snapshot plexes 257
snapshot volumes 248
space-optimized 252
destroying 265
displaying information 264
online data validation 273
reattaching 267
refreshing 260
restoring 261
resuming replication 252, 256, 259

snapshots of RVG *(continued)*
taking 266
unfreezing replication 252
Solaris
installing VRAdvisor 462
uninstalling VRAdvisor 462
SRL
and performance 379
associating to a Primary RVG
using command line 136
choosing disks for 380
concept 24
how to determine size 77
layout 379
mirroring 380
resizing
using command line 223
striping and performance 380
SRL fillup 454
SRL header error 373
SRL on bunker
disaster recovery 336
SRL overflow protection
choosing 70
disable 93
explanation 148
modes of 91, 148
preventing 373
with DCM
flags and definition 219
SRL playback 318
SRL size
calculating 458
results of analysis 455
srlprot 71
srlprot=dcm
override. *See* autodcm
STALE RLINK 499
starting replication 152, 346
startrep
vradmin keyword 155
state of replication 118
statistics
displaying for VVR 177
stats
vxrlink keyword 186, 474
vxrvg keyword 473
status
vxrlink keyword 474, 515
stopping replication to Secondary
using command line 228
stoprep
vradmin keyword 228
Storage Checkpoints
backing up Secondary with 279
creating 244
data volume error
cleanup using 375
recovery using 374
deleting 245
displaying list 176
ending 244
restoring Secondary
from Primary Storage Checkpoint 281
from Secondary Storage Checkpoint 279
Secondary pause 280
understanding 49
viewing
using command line 245
striping the SRL 380
supported locales 436
supported operating systems 461
synchronization period constraint 79
synchronizing Secondary
automatic synchronization
example 156
difference-based synchronization
about 509
example 510
using command line 510
full synchronization
about 502
example 504
using command line 502
incrementally after SRL overflow
about 218
using command line 220
replication using a bunker site 334
synchronizing volumes
about 199
using difference-based synchronization 200
using full synchronization 200
synchronous attribute
notes 68
synchronous mode
considerations 67
data flow 38
fail setting 67

synchronous mode *(continued)*
override setting 67
synchronous replication
explanation 28, 89
required bandwidth 455
setting up 146
synchronous=off 145
synchronous=override 146
synchronous writes in VVR 33
syncrvg
vradmin keyword
difference-based synchronization 509
full synchronization 502–503
syncvol
vradmin keyword 199
SystemZones attribute of RVG and RVG Primary agents 86

T

takeover
vradmin keyword 310
taking over from failed Primary
example 313–314
overview 308
using command line 310
tape backups
block-level 153, 505
TCP 73
TCP ports 74
TCP protocol 141
terms
replication
in VVR context 21
timeout
heartbeat 390
traditional snapshot method 55
transferring Primary role
failing back 316
methods 300
migrating 300
taking over 308
transmission unit
network maximum 77
trial failover
example 486
tunable parameters
buffer spaces 382
tunables, VVR
displaying 181
tuning VVR 380
Turn off
SmartMove for VVR 157

U

UDP 73
UDP ports 74
UDP protocol 141
unfreezing replication
after snapshot 252
using vxibc unfreeze 481
uninstalling VRAdvisor
on Solaris 462
on Windows 463
UNIX
collecting data on 436
unrecoverable I/O error 499
up-to-date data
concept 22
up-to-date Secondary
determining 189
update ID
displaying 189
updates
vxrlink keyword 189, 474
updating Secondary
using command line 347
usage constraint
peak 78
usage types
VVR volumes 137
utility
vrnotify 192
vxrlink stats 186

V

VCS
configuring RVG agent with 119, 124
VCS agents
automated failover 352
VCS agents for VVR 41
list 28
VCS environment
configuring VVR in 100
example setting up VVR 114
generic VVR setup 109
requirements for configuring VVR 108

VCS environment *(continued)*
setting up VVR
virtual IP requirement 117
verify
vradmin keyword 197
vxrlink keyword 474
verifydata
vradmin keyword 273
verifying
VVR replication state 118
verifying DR readiness
Failover 272
Firedrill 272
offline data validation 275
online data validation 273
overview 271
verifying secondary data 273
verifying volumes
about 273
before adding to RDS 198
using offline data verification 275
using online data verification 273
VFR
about 397
administration 403
creating consistency group 407
creating file replication job 405
deleting consistency group 421
deleting file replication job 420
error recovery 400
failover after a disaster 424
file replication job information 409
limitations 398
log files 400
managing file replication job 408
modifying consistency group 416
modifying file replication job 413
protecting target file system 405
recovering a failed site 426, 428
switchover 421
vfradmin command
cgrp create 407
cgrp destroy 421
cgrp exclude add 416
cgrp exclude remove 416
cgrp include add 416
cgrp include remove 416
cgrp list 413
cgrp set mntpt 416
vfradmin command *(continued)*
job abort 408
job clear cgrp 416
job create 405
job destroy 420
job get ckpt 412
job get stats 412
job get status 410
job list 410
job set cgrp 407
job set cksum 413
job set dbg 413
job set fcl 413
job set freq 413
job set host 413
job set mode 413
job set port 413
job set srcaddr 413
job set srcmntpt 413
job set tgtaddr 413
job set tgtmntpt 413
job start 408
job stop 408
job sync 408
job wait 408
viewing
compression ratio 182
consolidated statistics 177
data volume statistics 179
data volumes in RDS 162
flag settings 172
memory tunable parameters
statistics 181
network bandwidth used by VVR 182
replication status 164
RLINK statistics 177
RVG and RDS information 161
SRL volume statistics 180
status of individual RVG 161
status of specific RLINK 171
VVR statistics 177
virtual IP
requirement 117
vol_cmpres_enabled 393
vol_cmpres_threads 393
vol_dcm_replay_size 390
vol_max_nmpool_sz 388
vol_max_nmpool_sz tunable 188
vol_max_rdback_sz 385

vol_max_wrspool_sz 387
vol_min_lowmem_sz 383, 386–387
vol_nm_hb_timeout 390
vol_rvio_maxpool_sz 383, 386–387
vol_vvr_use_nat 393
voliomem_chunk_size 390
Volume Manager
    VVR as option 19
volume set
    associating to an RDS 201
volume sets
    using agents with 109
volumes
    associating to an RDS
        using command line 196
    removing or dissociating from an RDS
        using command line 212
    resizing
        about 208
        using command line 210
    synchronizing
        using difference-based synchronization 200
        using full synchronization 200
    verifying
        before adding to RDS 198
        using offline data verification 275
        using online data verification 273
volumes, synchronizing
    about 199
vradmin command
    command reference 466
vradmin utility
    addsec 139
    addvol 196
    changeip 229
    createpri 136–137
    creating RVGs 85
    delpri 242
    delsec 241
    delvol 211
    fbsync 312, 318, 321
    ibc
        compared with vxibc 477
        using off-host processing 283
    makesec 323
    migrate 302
    pauserep 227
    printrvg 161
    printvol 163
vradmin utility *(continued)*
    repstatus 164
    resizesrl keyword 224
    resizevol 208
    resumerep 227
    resync keyword 218
    set 144
    set keyword 227
    startrep 155
    stoprep 228
    syncrvg
        difference-based synchronization 509
        full synchronization 502–503
    syncvol 197, 199
    takeover 310
    verify 197
    verifydata 273
vradmind daemon
    restarting 177
VRAdvisor Wizard
    collecting data
        on Solaris 437
        on Windows 444
vrnotify utility 192
vrport command 237
vrstat command 177
VRTSvradv.msi
    installing 462
VVR
    buffer space 381
    data flow in 32
    features 20
    option of VxVM 19
    tuning 380
VVR agents
    configuring 118
    list of 28
VVR and Network Address Translation firewall 393
VVR compression
    about 61
    determining ratio 182
    disabling 98
    enabling 97
    tuning 393
VVR configuration
    setting up 115
VVR GUIs
    about 135

VVR in a VCS environment
configuring 100
requirements 108
set up example 114
virtual IP requirement for setting up 117
VVR setup in a VCS environment 109
VVR tunables
displaying 181
vxassist command
addlog keyword 205
free keyword 224
make keyword
associating DCM 205
creating cache volumes 254
creating snapshot volumes 251
mirror keyword 222
snapback keyword 268
snapshot keyword 265
snapstart keyword 266
vxdg command
split keyword 508, 517
upgrade keyword 248
vxedit command
rename keyword
for plex-breakoff snapshots 259
in Disk Split and Join 508
renaming volumes during RLINK recovery 362
rm keyword 222
set keyword
primary_datavol 215
setting synchronous=fail 145
set remote 217
vxfsrepld
daemon
starting 403
vxfstaskd
daemon
starting 403
vxibc utility
about 477
send command 480
unfreeze command 481
vxmake command
creating cache objects
for space-optimized snapshots 254
setting attributes 252
creating RLINK 475
creating RVGs 85
vxmake command *(continued)*
creating volumes during plex recovery 222
rlink keyword 216
vxmemstat command 383
command reference 475
versus vrstat command 177
vxplex command 221
vxprint command
displaying primary_datavol field 216
displaying VVR status 171
vxrlink att 497
vxrlink command
and clusters 42
assoc keyword 473
att keyword 473
checkdelete keyword 245
cplist keyword 176
det keyword 222, 473
dis keyword 473
pause keyword
creating Secondary Storage Checkpoint 244
RLINK FAIL state 499
using Secondary Storage Checkpoints 280
recover keyword
command reference 474
two-phase transactions in 501
restore keyword 281, 474, 499
resume keyword 473
ending Secondary Storage Checkpoint 244
using Secondary Storage Checkpoints 280
stats keyword
displaying network statistics 186
reference 474
versus vrstat 177
status keyword
displaying status of RLINK 174
reference 474
synchronization after SRL overflow 220
updates keyword 189, 474
verify keyword 474
versus vrstat 177
vxrlink recover 497
vxrsyncd daemon
ports 237
vxrvg command
checkdelete keyword 245
checkend keyword 244
checkstart keyword 244
command reference 471

vxrvg command *(continued)*
cplist keyword 176
getdatavols keyword 163
recover keyword 501
reference 471–473
snapback keyword
reattaching instant snapshot plexes 260
reattaching traditional snapshot plexes 267
using FastResync 270
snapdestroy keyword 265
snapprint keyword 264
snaprefresh keyword 260
snaprestore keyword 261–262
snapshot keyword
breaking off mirrors 221
creating instant snapshots 246
creating traditional snapshots 265
using FastResync 270
using with IBC 484
start keyword 496
stop keyword 263
vxsnap command 251
vxstat command 65
versus vrstat command 177
VxVM
VVR as option 19
vxvol command
aslog keyword
associating the SRL 371
assoc keyword
associating a new volume 362
during disaster recovery 222
setting Primary volume name on Secondary volume 216
dis keyword
dissociating the SRL 371
during disaster recovery 222
set keyword
enabling FastResync 271
start keyword
during disaster recovery 222
starting data volume 369
starting SRL 371
vxvset command 202

W

what-if analysis
overview 456
performing 457
what-if analysis *(continued)*
recording and viewing the results 459
Windows
collecting data 444
installing VRAdvisor 462
uninstalling VRAdvisor 463
write buffer space on the Primary 381
write latency 33, 63
write ship buffer space 387
write-order fidelity
concept 21
writes
how VVR processes 33
VVR processing writes 33
writing to Secondary data volumes 299