Veritas™ Volume Replicator Administrator's Guide

Solaris

5.1

symantec™

Veritas™ Volume Replicator Administrator's Guide

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

Product Version: 5.1

Document version: 5.1.0

Legal Notice

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014

http://www.symantec.com

Technical Support

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

Symantec's maintenance offerings include the following:

- A range of support options that give you the flexibility to select the right amount of service for any size organization
- Telephone and Web-based support that provides rapid response and up-to-the-minute information
- Upgrade assurance that delivers automatic software upgrade protection
- Global support that is available 24 hours a day, 7 days a week
- Advanced features, including Account Management Services

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

http://www.symantec.com/techsupp/

Contacting Technical Support

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

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

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

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

- Product release level
- Hardware information
- Available memory, disk space, and NIC information
- Operating system

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

Licensing and registration

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

http://www.symantec.com/techsupp/

Customer service

Customer service information is available at the following URL:

http://www.symantec.com/techsupp/

Customer Service is available to assist with the following types of issues:

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

Maintenance agreement resources

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

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

Additional enterprise services

Symantec offers a comprehensive set of services that allow you to maximize your investment in Symantec products and to develop your knowledge, expertise, and global insight, which enable you to manage your business risks proactively.

Enterprise services that are available include the following:

Symantec Early Warning Solutions	These solutions provide early warning of cyber attacks, comprehensive threat analysis, and countermeasures to prevent attacks before they occur.
Managed Security Services	These services remove the burden of managing and monitoring security devices and events, ensuring rapid response to real threats.
Consulting Services	Symantec Consulting Services provide on-site technical expertise from Symantec and its trusted partners. Symantec Consulting Services offer a variety of prepackaged and customizable options that include assessment, design, implementation, monitoring, and management capabilities. Each is focused on establishing and maintaining the integrity and availability of your IT resources.
Educational Services	Educational Services provide a full array of technical training, security education, security certification, and awareness communication programs.

To access more information about Enterprise services, please visit our Web site at the following URL:

http://www.symantec.com

Select your country or language from the site index.

Contents

Technical Support 4

Chapter 1 Introducing Veritas Volume Replicator 15

What is VVR? 15
- How VVR processes application writes 16

Features of VVR 17

Components of VVR 18
- Replicated Volume Group (RVG) 19
- Storage Replicator Log (SRL) 20
- Replication Link (RLINK) 20
- Data Change Map (DCM) 21
- Replicated Data Set (RDS) 21

Replication terms defined 22
- Write-order fidelity 22
- Consistent data versus current or up-to-date data 22
- IPv4-only node 23
- IPv6-only node 23
- Dual-node / Dual-stack 23
- IPv6-enabled node 23

How the VVR components fit together 23
- VVR at the Primary 24
- VVR at the Secondary 24
- Local host (localhost) 25

Modes of replication 25

Chapter 2 Understanding how VVR works 27

About Veritas Volume Replicator concepts 27

How data flows in VVR asynchronous mode 28

How data flows in VVR synchronous mode 29

How data flows in an RDS containing multiple Secondary hosts 30

Replication in a shared disk group environment 32
- The role of the logowner 33

Understanding how VVR logs writes to the SRL 38

Understanding checkpoints 40

Volume sets in VVR 42

Changing membership of an RVG and a volume set 42
Using MDFS with VVR 44
Cross-platform Data Sharing in VVR 44
Understanding the VVR snapshot feature 45
Snapshots of RVGs containing volume sets 46
About the traditional snapshot feature 46
About the instant snapshot feature 47
How VVR creates instant space-optimized snapshots 48
Comparing the snapshot methods based on different features 51

Chapter 3 Understanding replication settings for a Secondary 53

About replication settings for a Secondary 53
Modes of replication 54
Asynchronous replication 54
Synchronous replication 55
The synchronous attribute 55
Protecting against SRL overflow 56
The srlprot attribute 57
Setting up latency protection 59
The latencyprot attribute 59
Controlling the network bandwidth used for replication 61

Chapter 4 Setting up replication 63

About configuring replication 63
Best practices for setting up replication 64
Creating a Replicated Data Set 66
Creating a Primary RVG of an RDS 66
Adding a Secondary 68
Changing the replication settings for a Secondary 73
Synchronizing the Secondary and starting replication 80
Methods to synchronize the Secondary 81
Using the automatic synchronization feature 82
Using the full synchronization feature 84
Using block-level backup and checkpoint 87
Using the Disk Group Split and Join feature 89
Using difference-based synchronization 91
About SmartMove for VVR 93
Starting replication when the data volumes are zero initialized 93
Example: 93

Examples for setting up a simple Volume Replicator configuration 94
Creating a Replicated Data Set for the examples 95
Example 1—Setting up replication using automatic synchronization 96
Example 2—Setting up replication using full synchronization 97
Example 3—Setting up replication using block-level backup and checkpointing 97
Example 4—Setting up replication using Disk Group Split and Join 99
Example 5—Setting up replication using differences-based synchronization 101
Example 6—Setting up replication when data volumes are initialized with zeroes 101

Chapter 5 Displaying configuration information 103

Displaying RVG and RDS information 103
Displaying RDS information 103
Displaying an individual RVG 104
Displaying information about data volumes and volume sets 105
Displaying data volumes in a Replicated Data Set 105
Displaying a list of data volumes 106
Displaying information about all failed data volumes 106
Displaying an individual data volume 106
Displaying a volume set 106
Displaying information about Secondaries 107
Displaying consolidated replication status 107
Displaying a list of RLINKs 113
Displaying a specific RLINK 114
Displaying the status of a Secondary 116
Displaying a list of checkpoints 118
Displaying statistics with the vrstat display commands 119
Displaying the consolidated statistics 120
Displaying the RLINK information for all the hosts in the RDS 120
Displaying information about all the data volumes for all the hosts in the RDS 121
Displaying information about the SRL volumes for all the hosts in the RDS 122
Displaying information about the memory tunable parameters for all the hosts in the RDS 123

Determining the network bandwidth being used by VVR 124
Collecting consolidated statistics of the VVR components 125
Understanding how VVR stores the statistics 128
Displaying network performance data 129
Displaying extended replication statistics 130
Identifying the most up-to-date Secondary 132
VVR event notification 134

Chapter 6 Administering Veritas Volume Replicator 137

Administering data volumes 137
Associating a volume to a Replicated Data Set 137
Associating a volume set to an RDS 143
Associating a Data Change Map to a data volume 146
Resizing a data volume in a Replicated Data Set 149
Dissociating a data volume from its Replicated Data Set 152
Administering the SRL 154
Protecting from SRL overflow 155
Incrementally synchronizing the Secondary after SRL overflow 155
Resizing the SRL 160
Administering replication 163
Changing the replication settings 163
Pausing and resuming replication to a Secondary 163
Stopping replication to a Secondary 164
Changing the IP addresses used for replication 165
Changing the network ports used for replication 172
Administering the Replicated Data Set 176
Removing a Secondary from a Replicated Data Set 177
Removing a Primary RVG 178
Administering checkpoints 179
Creating checkpoints 179
Ending checkpoints 179
Viewing checkpoints 180
Deleting checkpoints 180
Creating RVG snapshots 181
Using the instant snapshot feature 181
Instant full snapshot 182
Instant space-optimized snapshots 187
Instant plex-breakoff snapshots 192
Administering snapshots 194
Using the traditional snapshot feature 200
Using Veritas Volume Manager FastResync 205

Verifying the DR readiness of a VVR setup 206
Performing a failover 206
Performing a firedrill 207
Verifying the data on the Secondary 207
Backing up the Secondary 213
Checkpoint pause/resume Secondary RLINK 214
Restoring the Secondary from online backup 215
Changing the VVR tunables 216
Points to note when changing the value of the tunables 218
Changing the tunable values using vxtune 218
Changing the tunable values using the vxio.conf file 221

Chapter 7 Using VVR for off-host processing 223

About using VVR for off-host processing 223
What is off-host processing? 224
In-Band Control Messaging overview 224
How to use the data on the Secondary 224
In-BandControl Messaging explained 225
Performing off-host processing tasks 228
Tasks to perform for off-host processing 228
Using the IBC messaging command vradmin ibc 229
Examples of off-host processing 234
Example 1—Decision support using the snapshot feature and the vradmin ibc command 234
Example 2—Backing up using the snapshot feature and the vradmin ibc command 236
Example 3—Performing block-level backup of the Secondary data using the vradmin ibc command 237

Chapter 8 Transferring the Primary role 239

About transferring the Primary role 239
Migrating the Primary 240
Prerequisites for migrating the Primary 242
Important notes for migrating the Primary role 242
Example 1—Migrating from a healthy Primary 243
Example 2—Migrating the Primary role in a setup with multiple Secondaries 244
Taking over from an original Primary 247
Important notes about taking over from an original Primary 249
Example 1—Taking over from an original Primary 252

Example 2—Taking over from an original Primary in a setup with multiple Secondaries 253
Failing back to the original Primary 255
Fast failback versus difference-based synchronization 256
Failing back using fast failback synchronization 256
Failing back using difference-based synchronization 262

Chapter 9 Replicating to a bunker 267

Introduction to bunker replication 267
Bunker replication during normal operations 268
How the bunker is used for disaster recovery 269
Sample configuration 271
Setting up bunker replication 271
Requirements for bunker replication 272
Adding a bunker to an RDS 272
Changing replication settings for the bunker Secondary 276
Starting replication to the bunker 278
Reinitializing the bunker 278
Administering bunker replication 279
Using a bunker for disaster recovery 279
Updating the Secondary from the bunker 279
Restoring the original Primary in a bunker setup 281
Bunker replication in a VCS environment 284
Automating local cluster failover for a bunker 284
Bunker replay in a VCS environment 286
Bunker replay in a VCS global cluster environment 286
Removing a bunker 286
Bunker commands 287

Chapter 10 Troubleshooting VVR 289

Recovery from RLINK connect problems 289
Recovery from configuration errors 291
Errors during an RLINK attach 292
Errors during modification of an RVG 295
Recovery 300
Primary-host crash 300
Recovering from Primary data volume error 300
Primary SRL volume error cleanup and restart 303
Primary SRL volume error at reboot 304
Primary SRL volume overflow recovery 304
Primary SRL header error cleanup and recovery 305
Secondary data volume error cleanup and recovery 306

Secondary SRL volume error cleanup and recovery 307
Secondary SRL header error cleanup and recovery 308
Secondary SRL header error at reboot 310

Appendix A VVR command reference 311

Command reference 311

Appendix B Messages 321

Kernel messages 321
Error messages 321
Informational messages 339
Utility error messages 341
The vradmin error messages 343
Messages related to configuration errors 360
Messages related to the vrstat command 365

Appendix C Using the In-band Control Messaging utility vxibc and the IBC programming API 367

About the IBC messaging utility vxibc 367
In-band Control Messaging overview 368
Using the IBC messaging command-line utility 369
Registering an application name 369
Displaying the registered application name for an RVG 369
Receiving an IBC message 369
Sending an IBC message 370
Unfreezing the Secondary RVG 371
Unregistering an application name 371
Receiving and processing an IBC message using a single command 371
Sending and processing an IBC message using a single command 372
Examples—Off-host processing 372
Example 1—Decision support using the traditional snapshot feature and the vxibc utility 374
Example 2—Backing up using the snapshot feature and the vxibc utility 375
Example 3—Trial failover using the snapshot feature 376
Example 4—Decision support using the instant full snapshot feature and the vxibc utility 377
In-band Control Messaging API 378
IOCTL commands 379

Using the IBC API 384

Appendix D Veritas Volume Replicator object states 385

Veritas Volume Replicator Kernel State 385
RVG KSTATEs 385
RLINK KSTATEs 386
Veritas Volume Replicator utility states 386
RVG utility states 386
RLINK utility states 386
Inactive RLINKs 387
STALE RLINK state 387
FAIL RLINK state 388
Inconsistent RLINKs 388
Pausing, resuming, and restoring RLINK states 389

Appendix E VVR task reference 391

Mapping the name of a Secondary data volume to a differently named Primary data volume 391
Mapping disk groups 393
Decreasing the size of the SRL on the Primary 394

Appendix F Migrating VVR from Internet Protocol version4 to Internet Protocol version6 397

Overview 397
Migrating to IPv6 when the GCO and VVR agents are not installed 398
Migrating to IPv6 when the GCO and VVR agents are installed 410

Glossary 417

Index 423

Chapter 1

Introducing Veritas Volume Replicator

This chapter includes the following topics:

- What is VVR?
- Features of VVR
- Components of VVR
- Replication terms defined
- How the VVR components fit together
- Modes of replication

What is VVR?

Veritas 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.

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.

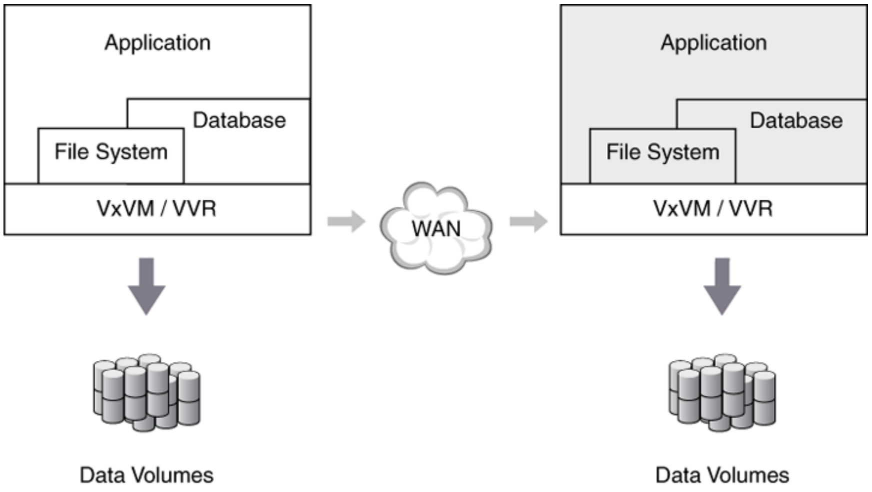
When VxVM is not being used, the application writes to a file system placed on a disk partition. In the case of applications or databases on raw devices, the database writes directly to the disk partition, instead of to a file system. In either case, the application, that is, the database or a file system, 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 1-1 shows how application writes are processed when VxVM and VVR are used.

Figure 1-1 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.

Features of VVR

Veritas Volume Replicator (VVR) includes the following features:

- 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.
- Can be configured to work with any storage hardware supported by Veritas Volume Manager.
- 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.

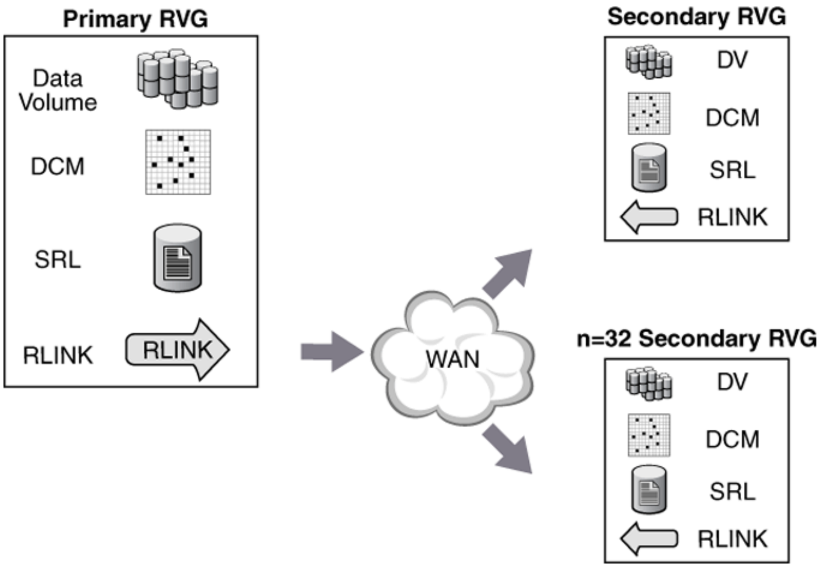
Components of VVR

This section explains the following components of VVR, which contain configuration information:

- “Replicated Volume Group (RVG)” on page 19.
- “Storage Replicator Log (SRL)” on page 20.
- “Replication Link (RLINK)” on page 20.
- “Data Change Map (DCM)” on page 21.
- “Replicated Data Set (RDS)” on page 21.

Figure 1-2 shows the VVR components for a sample configuration.

Figure 1-2 Sample configuration to illustrate the VVR components



Replicated Volume Group (RVG)

A Replicated Volume Group (RVG) is a group of volumes within a given VxVM disk group configured for replication. 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 must 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.

See "Storage Replicator Log (SRL)" on page 20.

See "Replication Link (RLINK)" on page 20.

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.

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, the SRL eliminates the need for the DRL because it provides faster resynchronization of data volumes.

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, the data volumes and SRL volumes must be configured on different physical disks to improve write performance. Note that VVR does not allow application writes to the SRL.

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 a 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.

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 contains a bitmap and can be optionally associated with a data volume on the Primary RVG.

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.

Replication terms defined

This section defines the following replication terms in the VVR context:

- Write-order fidelity
- Consistent data versus current or up-to-date data
- IPv4-only node
- IPv6-only node
- Dual-node / Dual-stack
- IPv6-enabled node

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.

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 only, that has IPv6 turned on.

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.

How the VVR components fit together

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

- "VVR at the Primary" on page 24.
- "VVR at the Secondary" on page 24.

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. The write to the data volume is written in the background and does not affect application performance. 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.

If the Primary crashes at any point 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 acknowledgement of receipt. This is known as the network acknowledgement. The network acknowledgement 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, it sends the second acknowledgement, the data acknowledgement.

The reason for the two-phase acknowledgement 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 acknowledgement 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 when using UDP, holds the packet until the all previous packets have been received, then writes to the disks in the correct sequence to maintain consistency at the Secondary. 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.

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 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 replication” on page 54.

Asynchronous 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. 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.

Chapter 2

Understanding how VVR works

This chapter includes the following topics:

- About Veritas Volume Replicator concepts
- How data flows in VVR asynchronous mode
- 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 checkpoints
- Volume sets in VVR
- Cross-platform Data Sharing in VVR
- Understanding the VVR snapshot feature

About Veritas Volume Replicator concepts

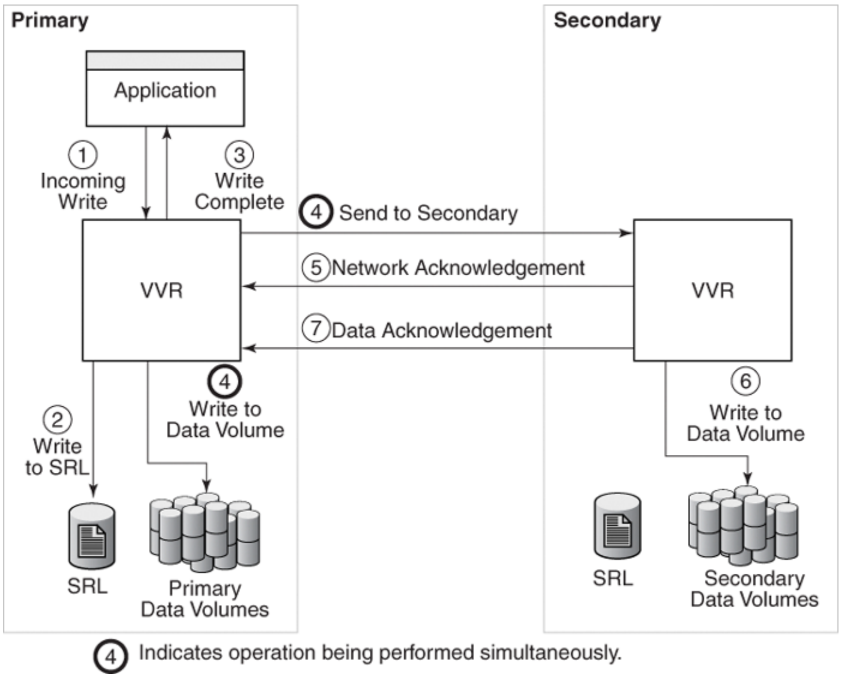
Understanding the Veritas Volume Replicator (VVR) concepts is the key to getting the maximum out of VVR. This topic explains the important concepts of VVR and thus prepares you to use VVR effectively. Read this topic before setting up replication.

How data flows in VVR asynchronous mode

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

Figure 2-1 shows how data flows in the asynchronous mode of replication.

Figure 2-1 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 acknowledgement, it knows that the write has been received in the Secondary VVR memory buffer.

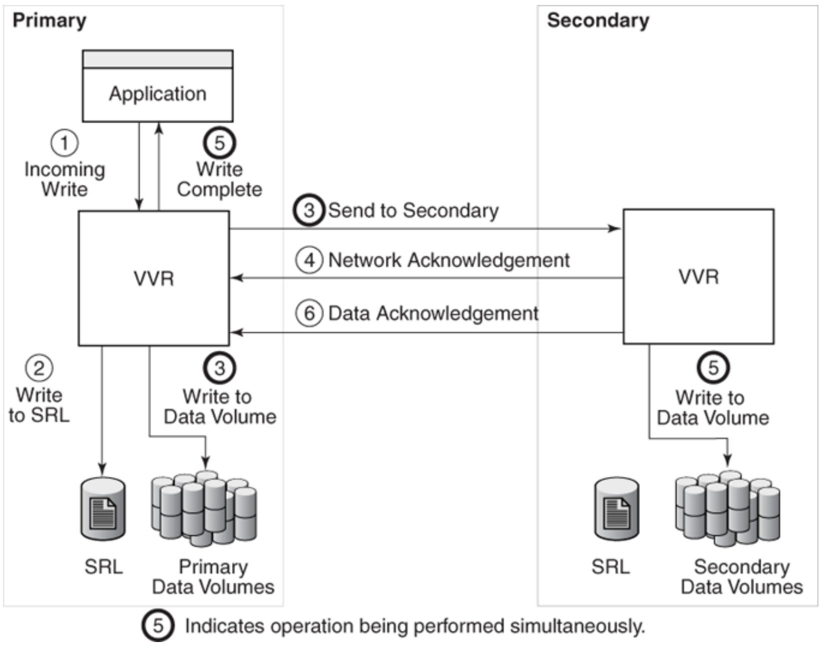
- VVR sends the writes to the data volumes on the Secondary and then sends a data acknowledgement to the Primary.
- When the Primary receives the data acknowledgement, VVR marks the write as complete in the SRL.

How data flows in VVR synchronous mode

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

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

Figure 2-2 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 acknowledgements 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 acknowledgement to the Primary.
- Sends writes to the data volumes on the Secondary; when the Primary receives a network acknowledgement from all the Secondary hosts, VVR acknowledges to the application that the write is complete.

  Note that the Secondary RVG sends the network acknowledgement 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 acknowledgement.
- When the write is written to the data volumes on the Secondary, VVR on the Secondary sends a data acknowledgement to the Primary. VVR marks the write as complete in the SRL when the Primary receives the data acknowledgement from all the Secondary hosts.

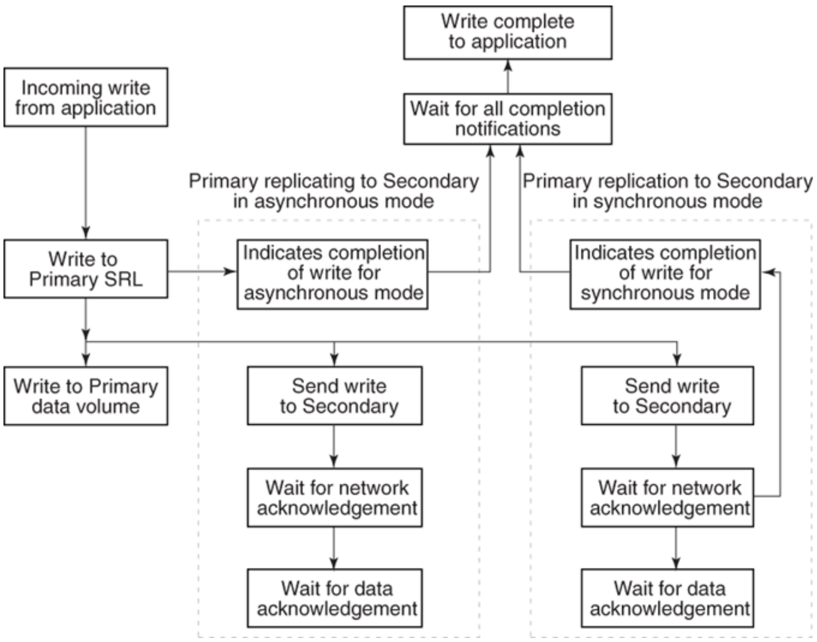
When an RDS containing multiple Secondary RVGs is replicating in synchronous mode, the application latency is determined by the slowest Secondary. Overall performance in synchronous mode is determined by the time to write to the SRL, plus the round-trip time required to send data to the Secondary RVG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

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 2-3 shows how data flows in an RDS with multiple Secondaries.

Figure 2-3 How data flows in an RDS with multiple Secondaries


Write complete to application
Incoming write from application
Wait for all completion notifications
Primary replicating to Secondary in asynchronous mode
Primary replication to Secondary in synchronous mode
Write to Primary SRL
Indicates completion of write for asynchronous mode
Indicates completion of write for synchronous mode
Write to Primary data volume
Send write to Secondary
Send write to Secondary
Wait for network acknowledgement
Wait for network acknowledgement
Wait for data acknowledgement
Wait for data acknowledgement

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 acknowledgement to the Primary.
- When the Primary receives a network acknowledgement 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 acknowledgement 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 acknowledgement 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 acknowledgement.

- When the write is written to the data volumes on the Secondary, VVR sends a data acknowledgement from the Secondary to the Primary in both synchronous and asynchronous mode.
- When the Primary receives the data acknowledgement 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 Veritas Cluster Server (VCS) 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. Disks in a shared disk group must be physically accessible from all systems that join the 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. If the Primary disk group is shared, the Secondary disk group need not be a shared disk group and 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.

For complete information on the cluster functionality (CVM) provided by VxVM, see the *Veritas Volume Manager Adminstrator's Guide*

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

For information about the VCS agents for VVR, see the *VCS Agents for VVR Configuration Guide.*

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

Note: Veritas Cluster Server is a separately licensed product. Veritas Cluster Server is not included with Veritas 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.

The administrative model of the cluster functionality of VxVM requires you to run all commands on the master node. VVR adheres to the same model as CVM for most commands that change the configuration. However, some informational and administrative commands that are specific to VVR can be performed on any node in the cluster. These commands include `vxrlink pause`, `vxrlink resume`, `vxrlink status`, and `vxrlink stats`.

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.

By default, VVR designates the CVM master as the logowner. It is not required that the master be the logowner. The decision to change the logowner must be made at the cluster-monitor level. The VCS agents for VVR software enables you to change the logowner, if you are using the VCS cluster monitor. You can configure any node in the cluster as the logowner, depending on your requirements. The node that generates the most writes should be configured as the logowner, because the logowner does not have to send messages.

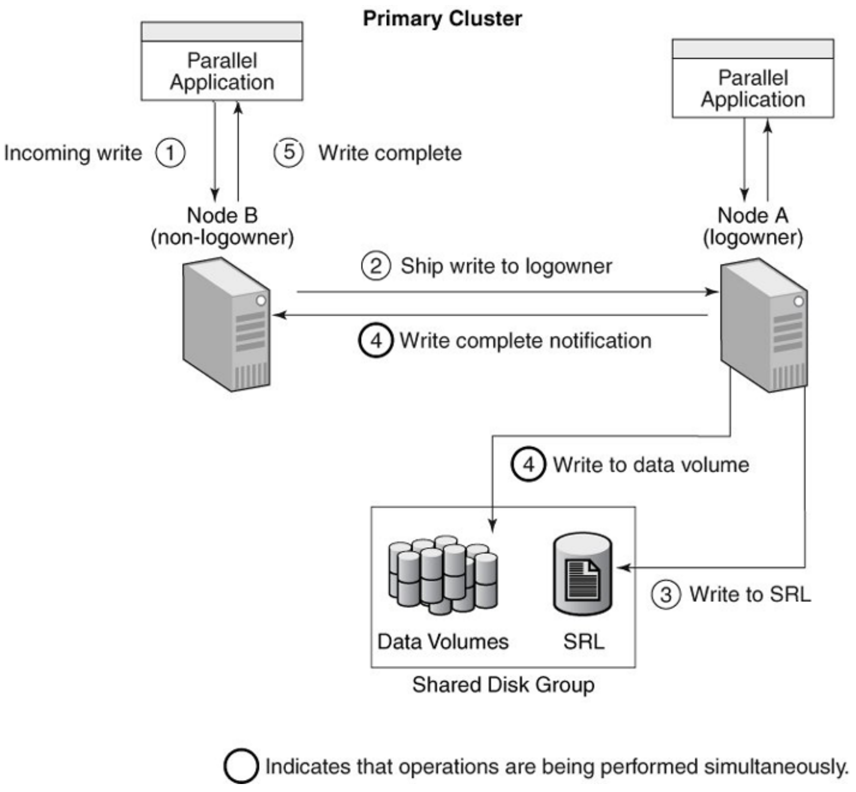
How VVR processes a write in a shared disk group

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.

Using write shipping

Figure 2-4 shows how VVR processes an incoming write on the non-logowner for an RVG that is using write shipping.

Figure 2-4 Example—how VVR processes a write on the non-logowner using write shipping



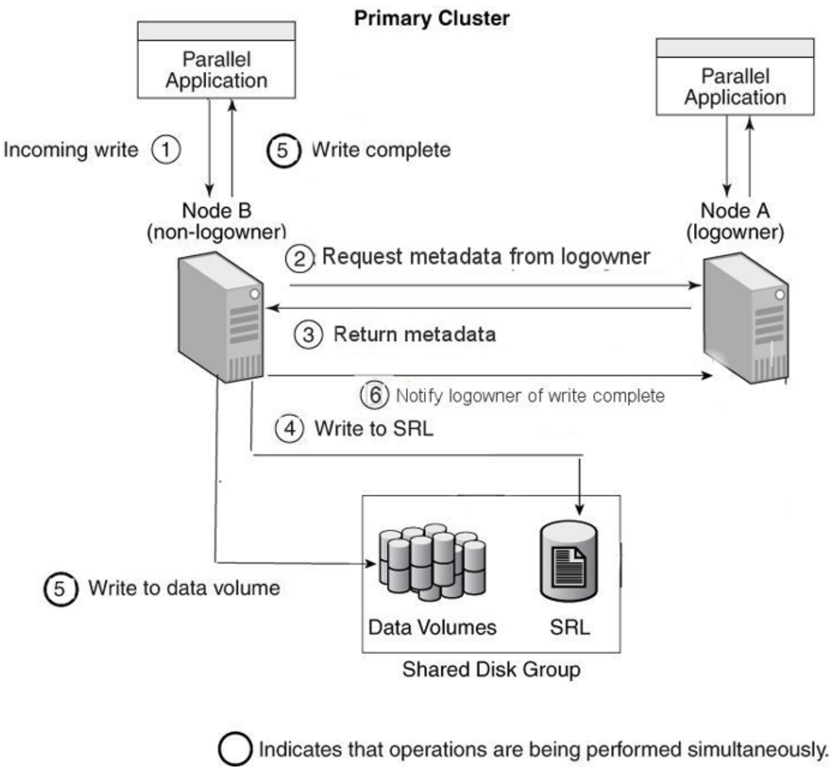
As shown in the illustration Figure 2-4, VVR processes an incoming write on the non-logowner (Node B) with the following steps in the order shown:

- 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.

Using metadata shipping

Figure 2-6 shows how VVR processes an incoming write on the non-logowner for an RVG that is using metadata shipping.

Figure 2-5 Example—How VVR Processes a Write on the Non-logowner Using Metadata Shipping



As shown in the illustration Figure 2-4, VVR processes an incoming write on the non-logowner (Node B) as follows:

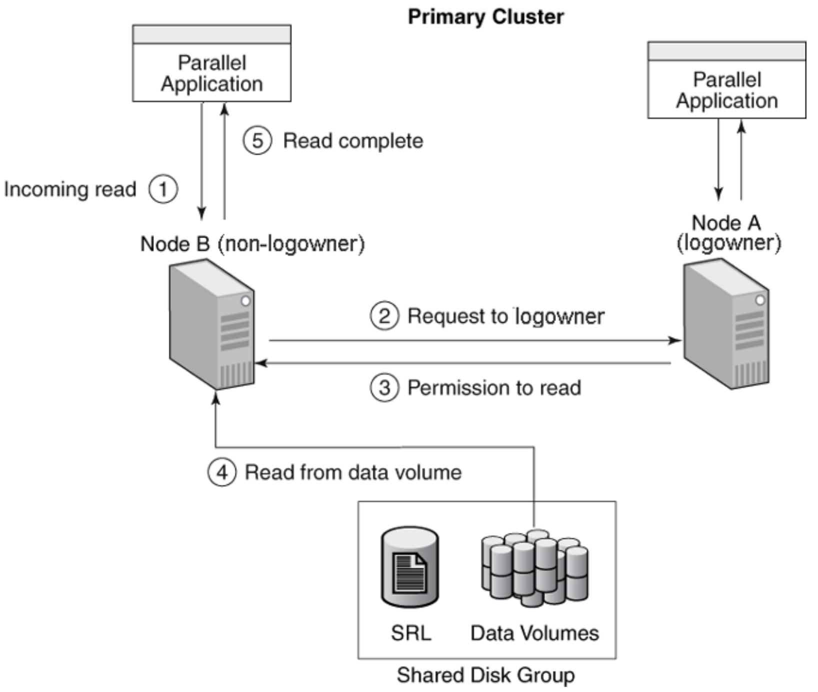
- VVR receives a write from the application on the non-logowner, Node B.
- Node B requests metadata to write to the SRL from the logowner, Node A.
- Node A sends metadata to write to Node B.
- After receiving metadata from Node A, Node B writes to the Primary SRL.
- Node B writes to the data volumes. 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 2-6 shows how VVR processes an incoming read on the non-logowner.

Figure 2-6 Example—how VVR processes a read on the non-logowner



As shown in the illustration, Figure 2-6, 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.

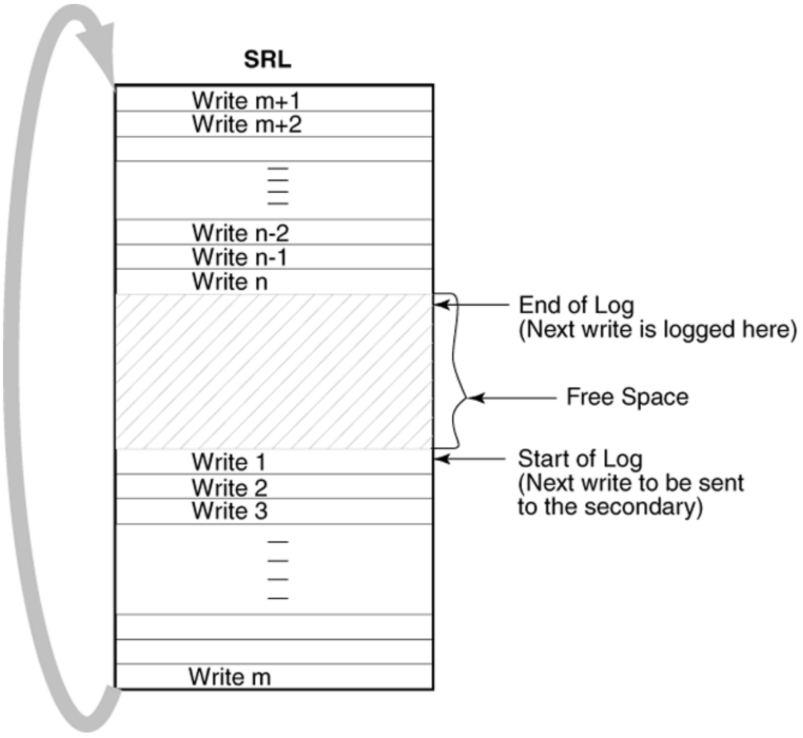
Understanding how VVR logs writes to the SRL

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 2-7 shows how writes are logged in the SRL.

Figure 2-7 Example—How VVR Logs Writes to the SRL



As shown in Figure 2-8, the earliest write that came in is Write 1, which also represents the Start of Log for the Secondary.

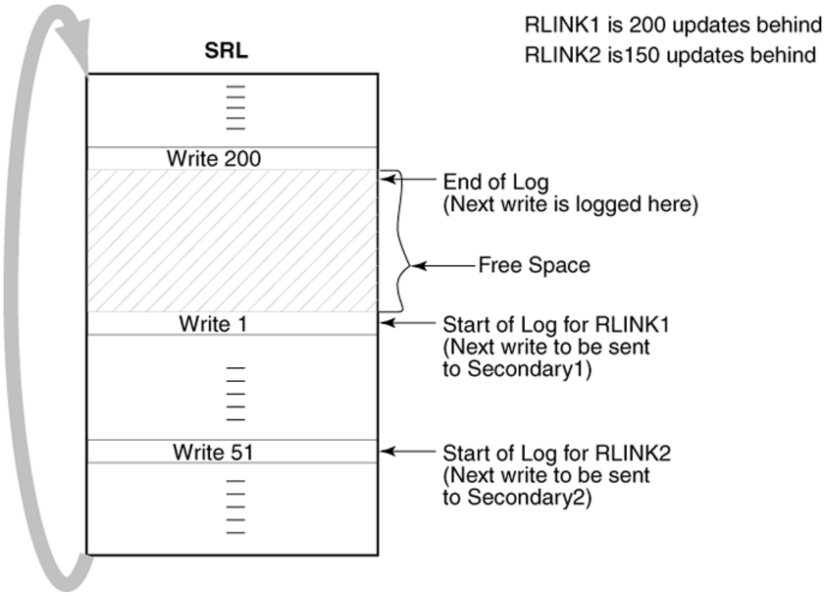
VVR logs Write 2, Write 3, Write m one after the other until it reaches the end of the SRL. Because the SRL is a circular log the next write, Write m+1 wraps around and logging continues. When the Primary receives the data acknowledgement from this Secondary host for Write 1, VVR marks the Write 1 as 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 2-8 shows how VVR logs writes to the SRL in an example configuration.

Figure 2-8 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 checkpoints

VVR checkpoints are user-defined markers in the SRL. Each checkpoint has a start (checkstart) and an end (checkend). 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.

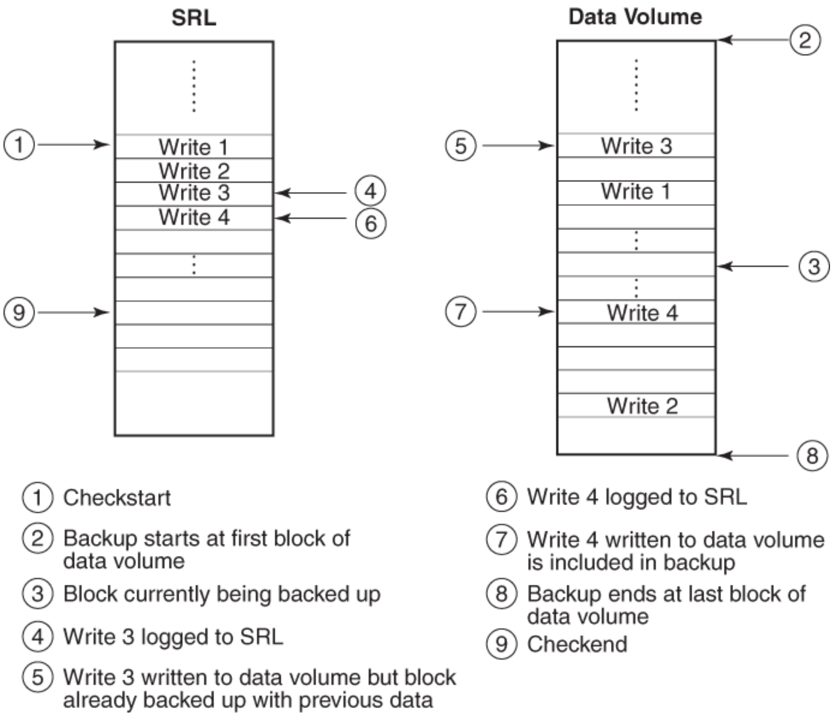
To make the Secondary consistent and up-to-date, VVR must transfer in order all the blocks that changed during the synchronization process. In a VVR environment, all writes to the Primary data volumes are logged to the SRL; therefore, VVR can transfer the writes that occurred during the synchronization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

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

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

Figure 2-9 shows how VVR uses the checkpoint.

Figure 2-9 Example—how VVR uses the checkpoint


SRL
Data Volume
Write 1
Write 2
Write 3
Write 4
Write 3
Write 1
Write 4
Write 2
1 Checkstart
2 Backup starts at first block of data volume
3 Block currently being backed up
4 Write 3 logged to SRL
5 Write 3 written to data volume but block already backed up with previous data
6 Write 4 logged to SRL
7 Write 4 written to data volume is included in backup
8 Backup ends at last block of data volume
9 Checkend

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 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

Veritas 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.

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.
`delvol vradmin` *`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 MDFS with VVR

The volume set feature supports the multi-device enhancement to Veritas File System (VxFS). This feature (MDFS) 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

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, you must include the index 0 volume, which includes metadata regarding the file system. 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 MDFS 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.

Veritas Volume Replicator (VVR) is now 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.

Refer to the *Veritas Storage Foundation Cross-Platform Data Sharing Administrator's Guide* for more information.

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 firedrill activities.
- 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 provides the following methods of creating snapshots:

- About the traditional snapshot feature
- About the instant snapshot feature

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.

See “Creating RVG snapshots” on page 181.

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.

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.

For more information on creating the plexes, refer to the *Veritas Volume Manager 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 200.

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:

- "About instant full snapshots" on page 47.
- "About Instant space-optimized snapshots" on page 47.
- "About Instant plex-breakoff snapshots" on page 48.

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 "Instant full snapshot" on page 182.

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.

See "Creating instant space-optimized snapshots" on page 189.

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 "Preparing the RVG volumes for snapshot operation" on page 188.

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 "Instant plex-breakoff snapshots" on page 192.

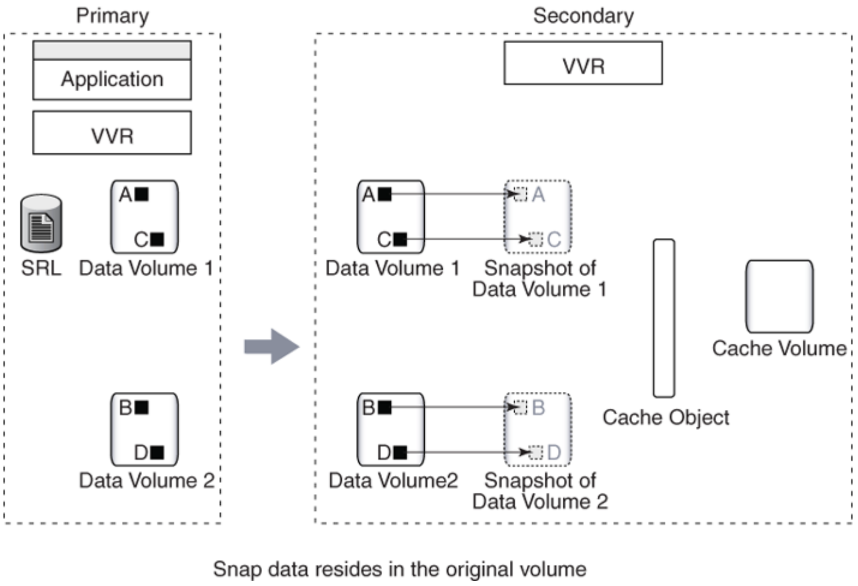
How VVR creates instant space-optimized snapshots

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.

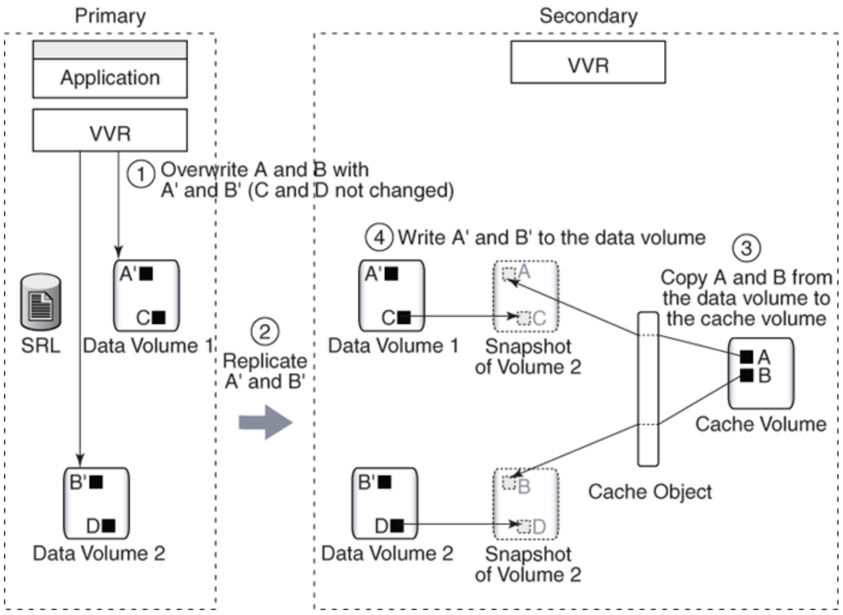
Figure 2-10 Example 1—How VVR creates instant space-optimized snapshots



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'.

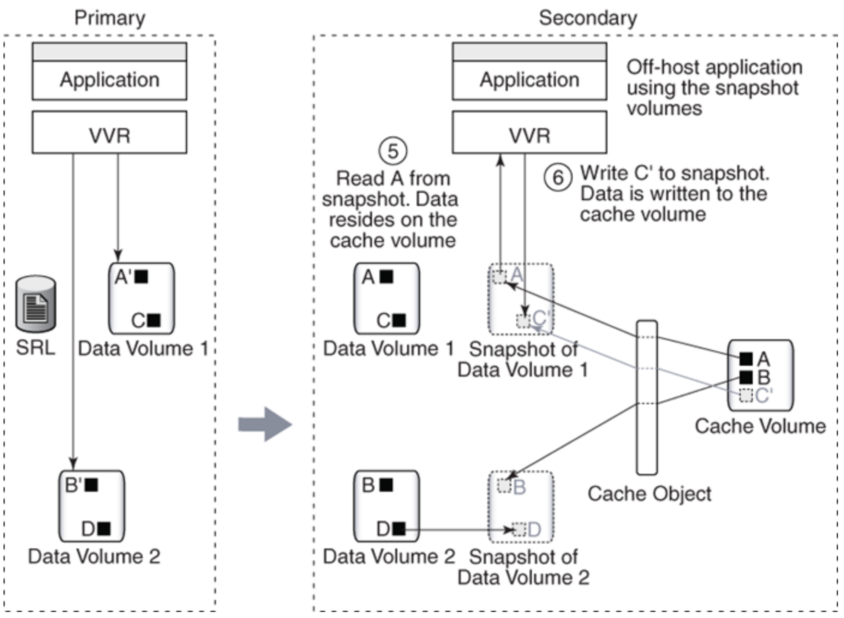
The Secondary also receives the updates A' and B'. The write to the data volumes first results in a copy-on-write on to the space-optimized snapshot. A space-optimized snapshot is created on a cache object that holds all the data for the snapshots. Hence during a copy-on-write, the blocks A and B get 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 2-11 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 2-12 Example 3—How VVR reads the changed block from the cache object


Primary
Application
VVR
SRL
A'
C
Data Volume 1
B'
D
Data Volume 2
Secondary
Application
VVR
Off-host application using the snapshot volumes
5
Read A from snapshot. Data resides on the cache volume
6
Write C' to snapshot. Data is written to the cache volume
A
C
Data Volume 1
A
C'
Snapshot of Data Volume 1
A
B
C'
Cache Volume
Cache Object
B
D
Data Volume 2
B
D
Snapshot of Data Volume 2

Comparing the 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

Chapter 3

Understanding replication settings for a Secondary

This chapter includes the following topics:

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

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 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 the *Veritas Volume Replicator Planning and Tuning Guide*.

Asynchronous 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. 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 asynchronous mode queues writes persistently and holds them at the Primary for later transmission, it is able to 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.

For more information about asynchronous mode, see the *Veritas Volume Replicator Planning and Tuning Guide*.

Synchronous 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 provides data currency but 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.

For a discussion of network latency and network bandwidth, and their effects on VVR performance, see the *Veritas Volume Replicator Planning and Tuning Guide*.

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 3-1 summarizes how the state of the RLINK affects the mode of replication.

Table 3-1 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

Table 3-1 Mode of replication and the state of the RLINK *(continued)*

Value of `synchronous` Attribute	When RLINK is Connected	When RLINK is Disconnected
`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 set the `synchronous` attribute to `override`.

synchronous=fail

Warning: If you use the `synchronous=fail` mode, you must read the section about synchronous mode considerations in the *Veritas Volume Replicator Planning and Tuning Guide*.

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 acknowledgement 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.

The srlprot attribute

VVR provides the following modes of SRL overflow protection: `autodcm`, `dcm`, 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`, or `override`. By default, the `srlprot` attribute is set to `autodcm`.

Table 3-2 summarizes how the state of the RLINK affects SRL protection when the SRL is about to overflow.

Table 3-2 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

Table 3-2 SRL protection and the state of the RLINK *(continued)*

Value of the `srlprot` Attribute	When RLINK is Connected	When RLINK is Disconnected
`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
`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`, `override`, or `dcm`, 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=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.

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

See "Setting the SRL overflow protection" on page 77.

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.

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

Table 3-3 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

Primary and Secondary 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.

Primary and Secondary 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 acknowledgements 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.

For information on how to decide the bandwidth limit to specify for VVR, refer to the *Veritas Volume Replicator Planning and Tuning Guide*.

For information about using the VRAdvisor to determine the bandwidth limit to specify for VVR, see the *Veritas Volume Replicator Advisor User's Guide*.

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 the network bandwidth being used by VVR" on page 124.

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.

Chapter 4

Setting up replication

This chapter includes the following topics:

- About configuring replication
- Best practices for setting up replication
- Creating a Replicated Data Set
- Synchronizing the Secondary and starting replication
- Starting replication when the data volumes are zero initialized
- Examples for setting up a simple Volume Replicator configuration

About configuring replication

You can configure and administer Veritas 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. The *Veritas Volume Replicator Administrator's Guide* (this guide) gives instructions on configuring, administering, and monitoring VVR using the CLI.
VVR VEA—Java-based desktop GUI	Veritas Enterprise Administrator (VEA) is a Java-based Graphical User Interface (GUI) that can be used to configure and manage storage objects. VVR VEA enables you to configure, monitor, and administer VVR in a distributed environment.

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 94.

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:

- Creating a Replicated Data Set
- Synchronizing the Secondary and starting replication

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

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 cluster package. 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. You must configure the Primary and Secondary data volumes with the same name.
- 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.
  For information on how to determine the size of the SRL, see the *Veritas Volume Replicator Planning and Tuning Guide*.
  The Veritas Volume Replicator Advisor (VRAdvisor), a tool to collect and analyze samples of data, can help you determine the optimal size of the SRL.
  For more information about VRAdvisor, see the *Veritas Volume Replicator Advisor User's Guide*.
- Include all the data volumes used by the application in the same RVG. This is mandatory.
- Provide dedicated bandwidth for VVR over a private 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 391.
- Make SRLs of equal size and the same names within the same RDS 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.
- If you are setting up replication in a shared environment, before you do so, determine the node that is performing the most writes by running the `vxstat` command on each node for a suitable period of time, and then after you set up replication, specify that node as the logowner.

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 143.
- Add a Secondary to the RDS
- Change the Replication Settings for the Secondary

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.

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.

See "Associating a Data Change Map to a data volume" on page 146.

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 137.

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.

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. In this release, 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 1

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 2

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

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: If the RDS contains the Primary only, the command must be issued on the Primary. If you issue the `vradmin addsec` command on the Secondary to be added to the RDS, the command fails as expected.

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 146.
- 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 2" on page 68.

Best practices for adding a Secondary

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. For more information, see the *Veritas Volume Replicator Planning and Tuning Guide*.
- 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 plan to have multiple secondaries in your RDS setup, we recommend that you create RLINKs between every pair of secondaries. If you do 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

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.

  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`. 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:

- "Setting the mode of replication" on page 74.
- "Setting the latency protection" on page 75.
- "Setting the SRL overflow protection" on page 77.
- "Setting the network transport protocol" on page 77.

- "Setting the packet size" on page 78.
- "Setting the bandwidth limit" on page 79.

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

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: You must read the section "Synchronous Mode Considerations" in the *Veritas Volume Replicator Planning and Tuning Guide* 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:

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:

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

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.

See "The latencyprot attribute" on page 59.

Set the `latencyprot` attribute to enable latency protection between a Primary and a Secondary.

Note: Before enabling latency protection, be sure you understand how latency protection works when the Primary and Secondary are connected or disconnected.

See "Primary and Secondary disconnected" on page 60.

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

VVR provides the following modes of SRL overflow protection: `autodcm`, `dcm`, and `override`.

See "The srlprot attribute" on page 57.

To enable SRL overflow protection

◆ Set the `srlprot` attribute of the corresponding RLINK to either `autodcm`, `dcm`, `override`

  - 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
    ```

  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

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, which is the default. If a protocol is not specified, then UDP is used as the protocol of communication between hosts.
- TCP—The hosts communicate using the TCP/IP 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 bunker replication" on page 267.

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

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. For more information on the `packet_size` attribute, see the *Veritas Volume Replicator Planning and Tuning Guide*.

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:

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

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 61.

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:

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:

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:

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. Use one of the following methods to synchronize the Secondary depending on your environment:

- Using the network
  - Automatic synchronization
  - Full synchronization with checkpoint
  - Difference-based synchronization with 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

You can synchronize the Secondary over the network either when the application is active or inactive.

Table 4-1 Synchronizing the Secondary using the network

To Synchronize the Secondary:	Perform:	Using This Command:
completely	automatic synchronization and start replication	`vradmin -a startrep`
completely	full synchronization with checkpoint	`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 checkpoint. See “Using difference-based synchronization” on page 91.	`vradmin -c checkpoint syncrvg`

Using block-level tape backup

Table 4-2 shows how to synchronize the Secondary using block-level tape backup.

Table 4-2 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 checkpoint.	`vxrvg -c checkpoint checkstart`
	2. Perform a block-level backup of the Primary.	
	3. End the Primary checkpoint.	`vxrvg checkend`
	4. Restore the tapes on the Secondary and start replication to the Secondary using the checkpoint.	`vradmin -c checkpoint startrep`

Moving disks physically

Table 4-3 shows how to synchronize the Secondary by moving disks physically.

Table 4-3 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 89.

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 116.

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.

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 checkpoint, synchronizes the data volumes, and ends the checkpoint after the synchronization completes. After the `vradmin syncrvg` command completes, use this checkpoint with the `vradmin startrep` command to start replication. To delete the Primary 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 checkpoint might overflow if the number of writes to the Primary data volumes is high when the Secondary is being synchronized.

A checkpoint that has overflowed becomes invalid and cannot be used to start replication. If the 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 93.

To synchronize the Secondary RVG with the Primary RVG using full synchronization with 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 checkpoint with the specified name. After the data volumes are synchronized, the checkpoint is ended. This 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 checkpoint of your choice.
3. After the synchronization completes, start replication to the Secondary with the 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 116.

Example—Synchronizing the Secondary using full synchronization with 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 checkpoint with the specified name. After the data volumes are synchronized, the checkpoint is ended. This checkpoint can then be used to start replication using the `vradmin startrep` command.

   The name *checkpt_presync* is the Primary checkpoint that you will create.
3. After the synchronization is complete, issue the following command to start replication using the checkpoint:

   ```
   # vradmin -g hrdg -c checkpt_presync startrep hr_rvg london
   ```

Using block-level backup and checkpoint

This method is useful for low bandwidth networks or very large data sets. You can use the block-level backup and checkpoint method to synchronize the Secondary when a backup of the data is available and a 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 "Resizing the SRL" on page 160.

Caution: During the process the checkpoint will overflow if the SRL fills up. To determine if the checkpoint has overflowed, issue the *vxrvg cplist rvg_name* command on the Primary to display the list of valid checkpoints.

See "Example—Synchronizing the Secondary using block-level backup" on page 88.

To synchronize the Secondary using backup and Primary checkpoint

1 Start a Primary 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 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 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 checkpoint.

To synchronize the Secondary using block-level backup and checkpoint

1 Start a Primary 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 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 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 *Veritas Volume Manager 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.

See "Example 4—Setting up replication using Disk Group Split and Join" on page 99.

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 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 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` 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.

For more information on MD5 checksum, refer to the `md5` manual page.

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 checkpoint with the specified name. After the data volumes are synchronized the checkpoint is ended. This 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 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 checkpoint with the specified name. After the data volumes are synchronized the checkpoint is ended. This checkpoint can then be used to start replication using the `vradmin startrep` command.

The name checkpt_presync is the Primary checkpoint that you will create.

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 resync of replicated volumes. This feature is available only when the volume being synchronized has a VxFS file system mounted on top of it. The default behavior is to use SmartMove for VVR feature for intial sync.

To turn off SmartMove for VVR

- Add `usesmartmovewithvvr=off` in the `/etc/default/vxsf` file.

The `vvradmin verifydata` command has also been enhanced to leverage VxFS knowledge of file system blocks in use for verification.

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:

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
```

Examples for setting up a simple Volume Replicator configuration

The examples in this section explain how to use Veritas 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`.

Assumptions:

- These examples assume that the Primary `seattle` and Secondary `london` have a disk group named `hrdg` with enough free space to create the VVR objects mentioned in the example.
- Examples 1, 2, 3, 4, and 5 assume that the Primary data volumes have been set up and contain data.
- The examples assume that the `/etc/vx/vras/.rdg` file on the Secondary host contains the Primary diskgroup 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. 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 1—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 95.

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
```

Example 2—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 95.

You can use full synchronization with checkpoint when the application is active or inactive.

To synchronize the Secondary using full synchronization with checkpoint

1. Synchronize the Secondary RVG `hr_rvg` on `london` with its Primary RVG on `seattle` using full synchronization with checkpoint:

```
# vradmin -g hrdg -full -c chkpt_presync syncrvg hr_rvg \
     london
```

2. Start replication with checkpoint by issuing the following command on any host in the RDS:

```
# vradmin -g hrdg -c chkpt_presync startrep hr_rvg london
```

Example 3—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 95.

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 checkpoint on the Primary:

```
# vxrvg -g hrdg -c checkpt_presync checkstart hr_rvg
```

   Note down the 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 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 checkpoint you created is still valid.

If the 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 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 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 checkpoint overflows before the Secondary becomes consistent, the synchronization process has failed. Increase the size of the SRL.

See "Resizing the SRL" on page 160.

Then restart the procedure beginning at step 1.

It is likely that there might be writes beyond the 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 checkpoint can be used to set up additional Secondary hosts while the checkpoint is still valid. If a checkpoint has overflowed, its corresponding backup cannot be used to resynchronize the Secondary. Eventually, any checkpoint that becomes STALE is unusable. There is no warning to indicate that this has occurred. However, the `vxrvg cplist` command indicates that the checkpoint has overflowed and hence is unusable.

See "Displaying a list of checkpoints" on page 118.

Example 4—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 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 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 5—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 95.

You can synchronize the Secondary using difference-based synchronization with checkpoint when the application is active or inactive.

To synchronize the Secondary using difference-based synchronization with checkpoint

1 Synchronize the Secondary RVG `hr_rvg` on `london` with its Primary RVG on `seattle` using difference-based synchronization with checkpoint:

```
# vradmin -g hrdg -c chkpt_presync syncrvg hr_rvg london
```

2 Start replication with checkpoint by issuing the following command on any host in the RDS:

```
# vradmin -g hrdg -c chkpt_presync startrep hr_rvg london
```

Example 6—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.

Chapter 5

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 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

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 an individual RVG

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 105.
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, 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 (autosync)
  Current mode:               asynchronous
  Logging to:                 DCM (contains 169728 Kbytes) (autosync)
  Timestamp Information:      N/A
  Bandwidth Limit:            30.00 Mbps
```

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
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 "RVG PASSTHRU mode" on page 304.

- **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 73.

- **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 155. needs failback synchronization—To continue replication, start failback synchronization to this Secondary. See "Failing back using fast failback synchronization" on page 256.
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 115. secondary log error—Secondary SRL has an I/O error. See "Interpreting RLINK flag settings" on page 115.
paused due to network disconnection	Replication to Secondary is paused because of some network problem.

Value	Meaning
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.
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 115. secondary log error—Secondary SRL has an I/O error. See “Interpreting RLINK flag settings” on page 115.
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`.
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 116.
  See "Identifying the most up-to-date Secondary " on page 132.

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 checkpoint only if the Primary RLINK is attached using the checkpoint or the Secondary RLINK has been restored using the checkpoint.
flags	See “Interpreting RLINK flag settings” on page 115.

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.
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.
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.

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 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 107.

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 acknowledgement 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-5-1-4348 DCM is in use on Rlink rlink_name.
DCM contains 88832 Kbytes.
```

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.
```

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 checkpoints

VVR enables you to get a list of the Primary checkpoints using the `vxrvg cplist` command; the `vxrlink cplist` command enables you to get a list of the Secondary checkpoints. The `vxrvg cplist` and `vxrlink cplist` commands can be run on the Primary only. VVR supports a maximum of 46 checkpoints, and hence the list displays a maximum of 46 checkpoints. If you try to create more than the specified number of checkpoints, an error message prompts you to delete older checkpoints before creating new checkpoints.

Primary 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 checkpoints associated with the specified RVG. The Primary checkpoints can be deleted using the *`vxrvg -c checkpoint_name checkdelete rvg_name`* command.

Secondary 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 checkpoints associated with the specified RLINK. The Secondary 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 checkpoint, such as checkpoint name, size, percentage of SRL used, and whether the checkpoint has been started or completed. If the SRL usage reaches 100%, the checkpoint overflows and becomes unusable. In this case, VVR displays the message `Checkpoint overflowed`.

To display a list of Primary checkpoints, enter the following command on the Primary:

```
# vxrvg -g diskgroup cplist rvg_name
```

To display a list of Secondary 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 stop
# /etc/init.d/vras-vradmind.sh start
```

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
```

The fields in the output are similar to those in the output of the `vxrlink stats` command.

See "Displaying network performance data" on page 129.

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      1328850.0    17.6
london   hr_srl          0           0          0           00.0     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    Readback      -     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     Readback      -     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 the network bandwidth being used by VVR

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

To view the network bandwidth currently being used by VVR

```
# 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 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.
```

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 stop
# /etc/init.d/vras-vradmind.sh start
```

You can restart `vradmind` even while the application is active and replication is in progress.

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 following headings.

See "Displaying network performance data" on page 129.

Messages

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.

Errors

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 5-1 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

Flow control

The fields displayed under this heading are the same as those displayed by the `vxrlink stats` command.

See "Displaying network performance data" on page 129.

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 512 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.

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 acknowledgement 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 5-2 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

Table 5-2 Event notifications *(continued)*

Event Type	Event Message
`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
`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 6

Administering Veritas Volume Replicator

This chapter includes the following topics:

- Administering data volumes
- Administering the SRL
- Administering replication
- Administering the Replicated Data Set
- Administering checkpoints
- Creating RVG snapshots
- Verifying the DR readiness of a VVR setup
- Backing up the Secondary
- Changing the VVR tunables

Administering data volumes

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 143.

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 210.

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 141.

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 141.

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

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 207.

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 93.

Example

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 93.

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.

Example

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 (MDFS). Associating the volume set to an RDS enables you to replicate an MDFS. For more information about volume sets, see the *Veritas Volume Manager 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 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 by default. This section describes how to associate DCMs to a data volume in an existing VVR configuration.

See "Data Change Map (DCM)" on page 21.

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 147.

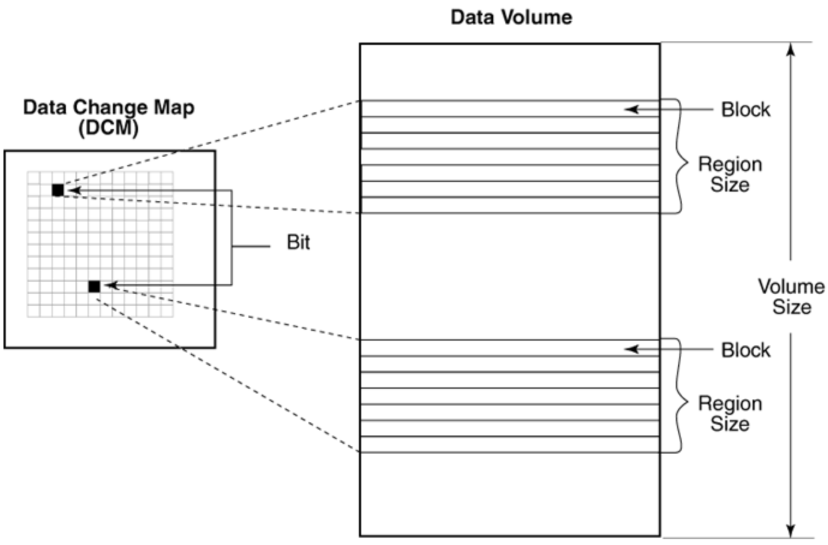
Note: If you try to grow a volume that has a DCM, an error message warns you if the DCM is not large enough for the increased size. In this case, dissociate the DCM, grow the volume, and then associate a new DCM to the volume.

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 6-1 shows the DCM and the region size.

Figure 6-1 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 6-1 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 6-1 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
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 6-2 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 6-2 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
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
```

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.

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
```

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.

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. For more information, refer to “Sizing the SRL” in the *Veritas Volume Replicator Planning and Tuning Guide*.

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 "The srlprot attribute" on page 57.

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 146.

To enable SRL protection, change the replication setting for SRL protection.

See "Changing the replication settings for a Secondary" on page 73.

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 188.

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 resynchronizing the Secondary

The following item is a prerequisite for incrementally resynchronizing the Secondary.

- The RVG must have the `dcm_logging` flag set.

To incrementally resynchronize 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 1—When snapshot plexes are not available

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 2—When snapshot plexes are not available and a disaster occurs

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 1—When snapshot plexes are not available" on page 158.

All steps are performed on the former Secondary, which is now the Primary.

See "Taking over from an original Primary" on page 247.

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.

Resizing the SRL

The size of the SRL must be large enough to meet the constraints explained in the section “Sizing the SRL” in the *Veritas Volume Replicator Planning and Tuning Guide*. 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. This section includes the following tasks:

- “Increasing the size of the SRL on the Primary and the Secondary” on page 161.
- “Decreasing the size of the SRL on the Primary” on page 162.
- “Decreasing the size of the SRL on the Secondary” on page 162.

Increasing the size of the SRL on the Primary and the Secondary

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:

  ```
  # vxdg -g diskgroup free
  ```

  If any host does not have enough space to resize the SRL, the `resizesrl` command will fail.

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
```

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.

Decreasing the size of the SRL on the Primary

For instructions on decreasing the size of the SRL on the Primary, see Decreasing the size of the SRL on the Primary.

Decreasing the size of the SRL on the Secondary

Note that you do not have to stop the applications while resizing the Secondary SRL.

To decrease the size of the SRL on the Secondary

1. Detach the RLINK:

   ```
   # vxrlink -g diskgroup det rlink_name
   ```

2. Dissociate the SRL from the RVG.

   Note: Any checkpoints that you have created will be lost after dissociating the SRL.# vxvol -g diskgroup dis srl_name

3. Decrease the size of the SRL using the `vxassist` command. For example, to decrease the size of the SRL:

   ```
   # vxassist -g diskgroup growto srl_name new_length
   ```

   Note: We recommend that the SRL reside on disks that are not being used for the data volumes. We also recommend that the Primary and the Secondary SRLs be the same size. See the `vxassist(1M)` manual page for more information.

4. Reassociate the SRL with the RVG:

   ```
   # vxvol -g diskgroup aslog rvg_name srl_name
   ```

5. Attach the RLINK:

   ```
   # vxrlink -g diskgroup att rlink_name
   ```

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 73.

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 "Primary and Secondary disconnected" on page 60.

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 6-3 shows the current configuration.

Table 6-3 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 6-4 shows the configuration after making the changes to the replication network.

Table 6-4 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 6-5 shows the current configuration.

Table 6-5 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 6-6 shows the configuration after making the changes to the replication network.

Table 6-6 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).

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

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 6-7 shows the ports which VVR uses by default when replicating data using UDP.

Table 6-7 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 6-7 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 6-8 shows the ports which VVR uses by default when replicating data using TCP.

Table 6-8 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.

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

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
```

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.

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.

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.

Note: For systems using the TCP protocol for replication, you are not required to select any data port as the connection is established with the listener port on the remote host. The listener uses this port number which is numerically same as the UDP port used for heartbeat messages.

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.

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 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 6-9 shows the sample configuration.

Table 6-9 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

The `vradmin delpri` command removes a Primary RVG from an RDS and thus deletes the corresponding RDS.

Prerequisite for deleting a Primary RVG

Observe the following prerequisites for deleting a Primary RVG:

- All Secondaries in the RDS must be removed.
  See "Removing a Secondary from a Replicated Data Set" on page 177.

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.

Example 1

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 2

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 checkpoints

A 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 80.
- restoring the Secondary data volumes from backup.
  See "Backing up the Secondary" on page 213.

See "Understanding checkpoints" on page 40.

This section describes how you can work with checkpoints.

Creating checkpoints

VVR enables you to create Primary and Secondary checkpoints. The Primary checkpoints are associated with an RVG. However, Secondary checkpoints are associated with an RLINK. VVR allows you to create a maximum of 46 checkpoints.

To create a Primary checkpoint

```
# vxrvg -c checkpt_name checkstart rvg_name
```

The argument *checkpt_name* is the name that you choose to specify for the checkpoint.

To create a Secondary checkpoint

```
# vxrlink -c checkpt_name pause rlink_name
```

The argument *checkpt_name* is the name that you choose to specify for the checkpoint.

Ending checkpoints

The end of the 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 checkpoint, for example, when the backup is complete, end the checkpoint in the SRL.

To end a Primary checkpoint

```
# vxrvg -g diskgroup checkend rvg_name
```

To end a Secondary checkpoint

For a Secondary checkpoint, resume replication to the Primary.

```
# vxrlink -c checkpt_name resume rlink_name
```

The checkpoint will end when the resume operation starts.

Secondary checkpoints are used when you back up the Secondary.

See "Backing up the Secondary" on page 213.

Viewing checkpoints

Primary checkpoints are associated with an RVG. You can display the list of Primary checkpoints by using the `vxrvg cplist` command.

Secondary checkpoints on the other hand are associated with an RLINK. You can display the list of Secondary checkpoints by using the vxrlink `cplist command` on the Primary.

See "Displaying a list of checkpoints" on page 118.

Deleting checkpoints

After you have finished using the checkpoints, you can delete the checkpoints. VVR allows you to retain a maximum of 46 checkpoints. To create any new checkpoint, delete an earlier checkpoint that you no longer require.

To delete a Primary checkpoint

```
# vxrvg -g diskgroup -c checkpt_name checkdelete rvg_name
```

The argument *`rvg_name`* is the name of the Primary RVG for which the checkpoint is to be deleted.

The argument *`checkpt_name`* is the name of the specific checkpoint to be deleted.

To delete a Secondary 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 checkpoint is to be deleted.

The argument *`checkpt_name`* is the name of the specific 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 181.

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 200.

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:

- "Instant full snapshot" on page 182.
- "Instant space-optimized snapshots" on page 187.
- "Instant plex-breakoff snapshots" on page 192.

Instant full snapshot

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 182.

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 *Veritas Volume Manager Administrator's Guide*.

Preparing the volumes

Before using the instant snapshot feature, prepare the volumes by performing the following tasks in the order shown:

- Upgrading the disk group
- Preparing volumes for an instant snapshot

Upgrading the disk group

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

Freezing or pausing replication

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 223.

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 141.

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 195.

Example

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

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 163.

   Freeze replication to the Secondary.

   See "About the IBC messaging utility vxibc" on page 367.

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 163.

If you used IBC messaging to freeze replication, unfreeze replication.

See "About the IBC messaging utility vxibc" on page 367.

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 now ready for use.

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 *Veritas Volume Manager 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

You must prepare the volumes under the RVG for the snapshot operation.

See "Preparing the volumes" on page 183.

Creating the cache object

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 command on `seattle`:

```
# vxmake -g hrdg cache cache-obj cachevolname=cache-vol \
  autogrow=on regionsize=128
```

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

For more information on freezing or pausing the replication, refer to Freezing or pausing replication.

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 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" on page 189.

   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 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.

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” on page 183.

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 *Veritas Volume Manager Administrator’s Guide*.

Freezing or pausing replication

For more information on freezing or pausing the replication, refer to Freezing or pausing replication.

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:

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 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. 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
```

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 197.

Note: Restoring data on the Primary RVG volumes using the `vxrvg snaprestore` command deletes all the existing 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
   ```

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 2003
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 2003
```

```
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 2003
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 2003
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 2003
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:

- "Creating snapshot plexes for data volumes in an RVG" on page 200.
- "Freezing or pausing replication" on page 201.
- "Taking a snapshot" on page 201.
- "Unfreezing or resuming replication" on page 201.
- "Reattaching the snapshot plexes to the data volumes (Snapback)" on page 202.

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 367.

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 200.

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 *Veritas Volume Manager 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 81.

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 240.

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 checkpoint, make sure that any Primary checkpoints that were taken before the snapback operation are not used to resynchronize the Secondary. VVR may show these checkpoints as valid if they have not overflowed; however, the checkpoints are not valid. You can only use Primary 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 215.

If you choose to restore the Secondary using checkpoints, you must ensure that:

- The snapshot volume being used for the `resyncfromreplica` operation corresponds to the checkpoint to be used in the `vxrlink restore` command for the RLINK.
- The checkpoint is still valid before proceeding with the `resyncfromreplica` snapback operation. Issue the following command to determine whether the 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 *Veritas Volume Manager 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 *Veritas Volume Manager 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 firedrill 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 181.

VVR enables you to validate the DR Readiness of the Secondary by using one of the following methods.

- “Performing a failover” on page 206.
- “Performing a firedrill” on page 207.
- “Verifying the data on the Secondary” on page 207.

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 239.

Performing a firedrill

Firedrill 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 181.

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 firedrills:

- A firedrill 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 firedrill no IBC messages are required to be sent so that a failover scenario similar to a real one is simulated.

Automating the firedrill procedure

The firedrill 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 firedrill 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 firedrill 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 139.

See "About SmartMove for VVR" on page 93.

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 scripts" on page 233.

To perform online data verification

1. Prepare the volumes that need to be included in the snapshot.

   See "Preparing the volumes" on page 183.

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 188.

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 91.

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 checkpoint at the Secondary node. This causes a request to be sent to the Primary node to pause updates and record the Secondary 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 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 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 checkpoints on the Secondary using the `vxrlink cplist` command.

See "Displaying a list of checkpoints" on page 118.

Checkpoint pause/resume Secondary RLINK

If a Secondary data volume fails and there is a 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 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 checkpoints are about to overflow.

The checkpoint string can be up to 19 characters long for either a Primary or a Secondary checkpoint. Only the most recent Primary checkpoint string is displayed in the Primary RVG.

Unlike a simple Secondary pause, a 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 checkpoint.

To create a Secondary checkpoint

On the Secondary:

1. Pause the RLINK using a checkpoint.

   ```
   # vxrlink -g diskgroup -c sec_checkpointname pause \
    rlink_name
   ```

   Note: Pausing the RLINK with a checkpoint on the Secondary creates the Secondary 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 checkpoint

1 Pause the RLINK using a checkpoint on Secondary.

```
# vxrlink -g diskgroup -c sec_checkpointname pause \
 rlink_name
```

2 Delete the Secondary 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:

- "Restoring from Secondary checkpoints" on page 215.
- "Restoring a Secondary RLINK " on page 215.

Restoring from Secondary 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 checkpoint. A `restore` is not guaranteed to succeed though because 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 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 checkpoint exists. If a Primary checkpoint still exists, it can be used if there is no Secondary 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 checkpoint from the Primary.

As with Primary checkpoints, if the checkpoint is not used before the SRL wraps around and the SRL overflows, the checkpoint will become STALE. If the 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 81.

To prevent the 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.

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.

Changing the VVR tunables

VVR provides you with a number of tunable parameters that can be tuned to a specific value, according to your requirements. For a detailed explanation of the VVR tunables, see the *Veritas Volume Replicator Planning and Tuning Guide*.

The following table provides you with a quick reference to the tunables that can be modified using the `vxtune` utility.

All tunables can be modified using the system-specific method.

Table 6-10 VVR Tunables

Tunable Name	Modifying Tunables Using the vxtune Utility	Values
vol_rvio_maxpool_sz	Yes	bytes
vol_min_lowmem_sz	Yes	bytes
vol_max_rdback_sz	Yes	bytes
vol_max_nmpool_sz	Yes	bytes
vol_max_wrspool_sz	Yes	bytes
vol_dcm_replay_sz	No	bytes
vol_nm_hb_timeout	No	bytes
voliomem_chunk_size	No	bytes
vol_vvr_use_nat	No	0 or 1
volpagemod_max_memsz	Yes	bytes

Note: The `volpagemod_max_memsz` is a VxVM tunable that is used to specify the amount of memory, measured in kilobytes, required for caching FastResync and cache object metadata. For more information on using this tunable, refer to “Performance Tuning and Monitoring” in the *Veritas Volume Manager Administrator’s Guide*.

The tunables can be tuned either using the `vxtune` utility or by using the system-specific interface. Some of the tunables can be tuned using only the system-specific method, while the others such as the memory tunables can be tuned using both the methods. The advantage of using the `vxtune` utility to tune the parameters is that you need not reboot the system after modifying the tunable value. This is especially useful if you want to experiment with different values to arrive at an optimum value to suit your requirements. However, the changes to the tunables using the `vxtune` utility are non-persistent. To make the changes persistent you still must use the system-specific method. However, you must reboot the system for the changes to take effect.

The current values for the tunables are defined in the `/etc/vx/vxtunables` file after you have used the `vxtune` utility for the first time.

Points to note when changing the value of the tunables

Note the following points when changing the value of the tunables:

- When decreasing the value of the `vol_rvio_maxpool_sz` tunable, all the RVGs on the host must be stopped.
- When decreasing the size of the tunables `vol_max_rdback_sz` and `vol_max_nmpool_sz` pause the RLINKs.

  Note: `vol_max_wrspool_sz also` pauses the RLINKs.

- The `vol_min_lowmem_sz` tunable is auto-tunable; depending on the incoming writes VVR increases or decreases the tunable value.

In a shared disk group environment you may choose to set only those tunables that are required on each host. However, we recommend that you configure the tunables appropriately even if the tunables are currently not being used. This is because if the logowner changes, then tunables on the new logowner will be used. The following list of tunables are required to be set only on the logowner and not the other hosts:

- `vol_max_rdback_sz`
- `vol_max_nmpool_sz`
- `vol_max_wrspool_sz`
- `vol_dcm_replay_size`
- `vol_nm_hb_timeout`
- `vol_vvr_use_nat`

The tunable changes that are done using the `vxtune` command affects only the tunable values on the host on which it is run. Therefore, in a shared disk group environment, you must run the command separately on each host for which you want to change the tunable values.

Changing the tunable values using vxtune

You can use `vxtune` to display, set, or change the memory tunables that are used by VVR. The advantage of using the vxtune utility to tune the parameters is that you need not reboot the system after modifying the tunable value.

The `vxtune` utility allows you to use suffixes of K or M to specify values, and enables you to modify the values of the following memory tunables:

```
    vol_rvio_maxpool_sz
    vol_min_lowmem_sz
    vol_max_rdback_sz
    vol_max_nmpool_sz
    vol_max_wrspool_sz
```

These tunable values are then updated in the `/etc/vx/vxtunables` file.

You can enable auto-tuning for the `vol_min_lowmem_sz` tunable by setting the tunable value to -1. However, if you do not want it to be auto-tuned then you must set it to a required value. Auto-tuning is only supported for the tunable vol_min_lowmem_sz.

To display the tunables that vxtune supports

Issue the following command on the required system to display the tunables that VVR supports along with their current values and brief description:

```
# vxtune
```

The output of this command resembles:

```
Tunable                   Value         Description
-----------------        --------      ---------------
vol_rvio_maxpool_sz        102325       RVIO Pool Size (KBytes)
vol_min_lowmem_sz             528       Low Memory Threshold (KBytes)
vol_max_rdback_sz           65536       Readback Pool Size (KBytes)
vol_max_nmpool_sz           16384       NMCOM Pool Size (KBytes)
vol_max_wrspool_sz          16384       WriteShippingPoolSize(KBytes)
volpagemod_max_memsz         6144       Cache SizeFMRMetadata(KBytes)
```

Note: The default unit for value is bytes and default display value of the tunables is in kilobytes (K).

To display the output in bytes, use the `vxtune` command with the `-r` option as follows

```
# vxtune -r
```

The output resembles:

```
 Tunable                  Value         Description
-----------------        --------      ---------------
vol_rvio_maxpool_sz     104780185       RVIO Pool Size (Bytes)
vol_min_lowmem_sz          540672       Low Memory Threshold (Bytes)
vol_max_rdback_sz        67108864       Readback Pool Size (Bytes)
```

```
vol_max_nmpool_sz        16777216       NMCOM Pool Size (Bytes)
vol_max_wrspool_sz       16777216       Write ShippingPoolSize(Bytes)
```

To display the values of specific tunables

Use the following command on the required host to display the value of the specific tunable:

```
# vxtune tunable_name
```

This command displays the value of the specified tunable in kilobytes (K).

To display the output in bytes, use the `vxtune` command with the -r option.

```
# vxtune -r tunable_name
```

For example, to view the value of the tunable `vol_rvio_maxpool_sz`, use the following command:

```
#  vxtune vol_rvio_maxpool_sz
```

The output resembles:

```
65536(K)
```

To modify the values of the tunables

Use the following command on the required host to modify the value of the specific tunable:

```
# vxtune tunable_name value
```

The *`value`* value of the tunable can be specified in units of K, MB, or GB. However, the value is always displayed in kilobytes (K) regardless of the units that you have specified it in. For example, if you specify a value 500MB the value will be automatically converted and will be displayed as 512000K. You can use the command with the `-r` option to display the value in bytes.

For example, to change the default value of the `vol_rvio_maxpool_sz` tunable to 128MB, use the following command.

```
# vxtune vol_rvio_maxpool_sz 128M
```

To view the changed value for the tunable use the following command

```
# vxtune vol_rvio_maxpool_sz
```

Caution: Do not edit the tunable values directly in the `vxtunables` file because these changes will be ignored.

Changing the tunable values using the vxio.conf file

You can also tune all the VVR parameters by editing `the /kernel/drv/vxio.conf` file. The changes made to the tunables using this method are persistent. However, to have persistent tunable values it is essential to reboot the system after the value of the tunable has been modified. For an explanation of VVR tunables, see the *Veritas Volume Replicator Planning and Tuning Guide*. To modify tunables, you will either need to add a tunable to the `vxio.conf` file or edit an existing tunable in the `/kernel/drv/vxio.conf` file.

To change the value of a tunable

1. Navigate to the `/kernel/drv/` directory which contains the `vxio.conf` file.
2. Open this file using any editor. Add or edit the VVR tunable in the `/kernel/drv/vxio.conf` file, using the following format:

   ```
   tunable_name=value;
   ```

   The change will take effect only after the next system reboot.

Example 1

To change the value of the tunable `vol_rvio_maxpool_sz` to 128M, append the following line to the file `/kernel/drv/vxio.conf`:

```
vol_rvio_maxpool_sz=134217728;
```

The specified value for `vol_rvio_maxpool_sz` is now applicable system-wide.

Note: The value for the tunables using the vxio.conf file must be specified in bytes.

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 Veritas Volume Replicator (VVR) for off-host processing. The In-Band Control (IBC) Messaging feature with the Snapshot feature of VVR and the optional FastResync (FR) feature of Veritas Volume Manager (VxVM) enable you to perform off-host processing.

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 367.

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 split off a mirror, 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 369.

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

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 VVR.

See "Using the instant snapshot feature" on page 181.

VVR also provides sample IBC scripts that can be used for creating snapshots.

See "Sample scripts" on page 233.

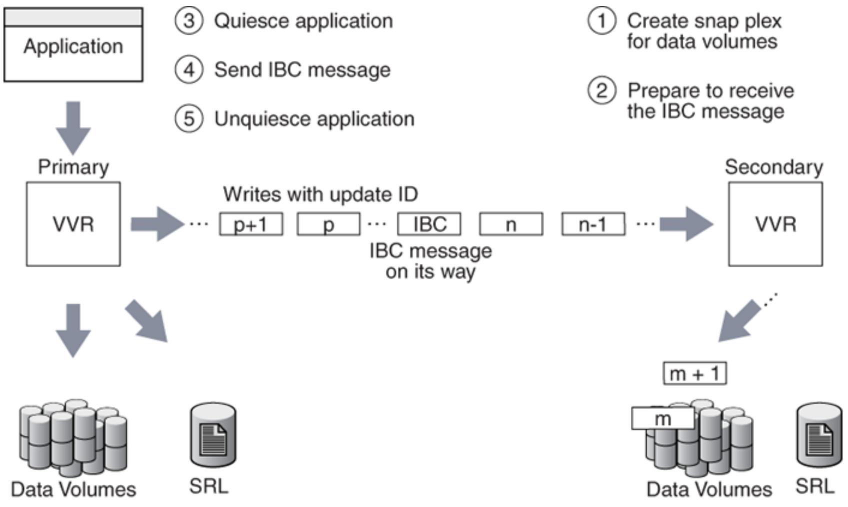
With the traditional snapshot feature, depending on the size of the volume, the time required for initial synchronization of the plexes can be very large.

See "Using the traditional snapshot feature" on page 200.

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.

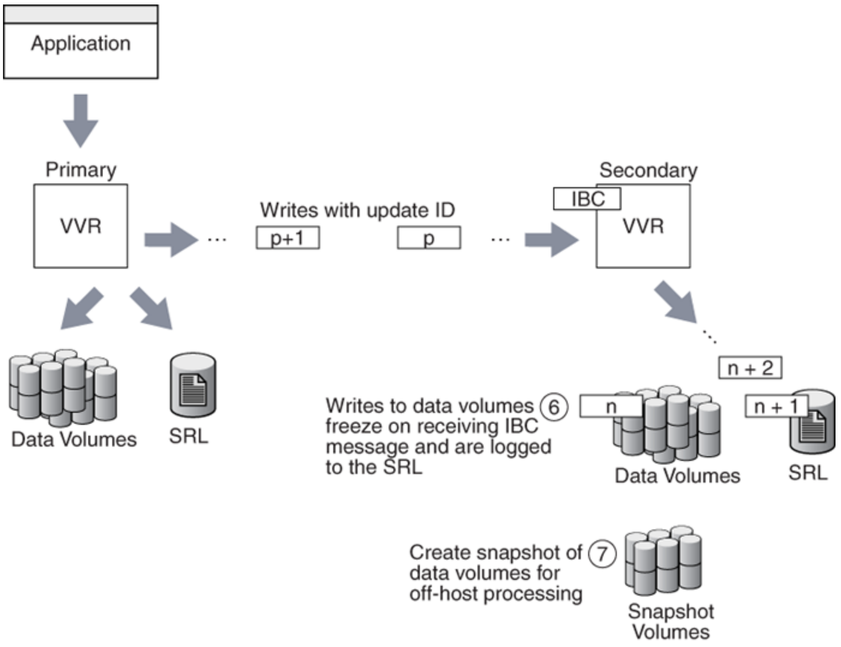
In-BandControl Messaging explained

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 and any new updates received after the IBC message are logged into the Secondary SRL. As a result, the Secondary does not reflect further updates to data volumes until the user acknowledges the IBC message.

The Secondary data volumes are now consistent at the application level.


Application
Primary
VVR
Data Volumes
SRL
Writes with update ID
p+1
p
Secondary
IBC
VVR
n + 2
Writes to data volumes freeze on receiving IBC message and are logged to the SRL
6
n
n + 1
Data Volumes
SRL
Create snapshot of data volumes for off-host processing
7
Snapshot Volumes

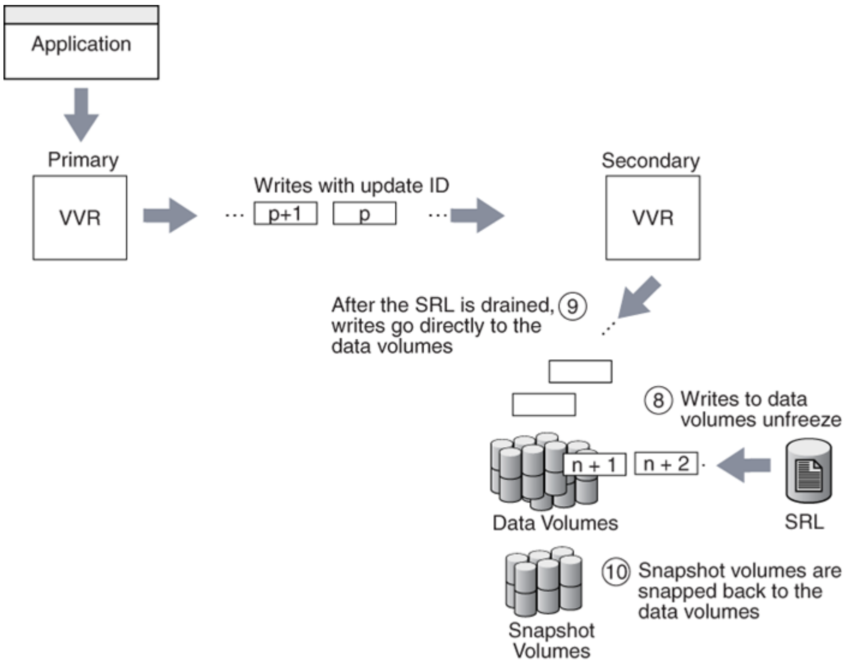
While replication is frozen, take snapshots of the data volumes on the Secondary. The created snapshot volumes are consistent at the application level and require less time to recover when the application on the Secondary starts.

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 subsequent updates are logged into the SRL, and 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.


Application
Primary
VVR
Writes with update ID
p+1
p
Secondary
VVR
After the SRL is drained, writes go directly to the data volumes
9
8
Writes to data volumes unfreeze
n + 1
n + 2
Data Volumes
SRL
10
Snapshot volumes are snapped back to the data volumes
Snapshot Volumes

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 be applied immediately to the Secondary data volumes, or logging the subsequent updates into the SRL 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 Secodary.
- 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 231.

You can also use the `vxibc` commands to perform off-host processing operations.

See "Using the IBC messaging command-line utility" on page 369.

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` pemissions for `root` user.

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 231.

4. Run the following command from any host in the RDS:

   ```
   # vradmin -g diskgroup ibc rvg_name task_name [sechost]...
   ```

   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
   ```

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.

Location of the scripts

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 7-1 shows the locations of the scripts for off-host processing.

Table 7-1 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.

The following section describes how each script is used with the `vradmin ibc` ommand:

- 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.

Sample 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 `/etc/vx/vvr/ibc_scripts` directory for instructions on how to use the sample scripts 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 1—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/vxassist -g hrdg snapstart hr_dv01
   /usr/sbin/vxassist -g hrdg snapstart hr_dv02
   ```

   In the `onfreeze` script, include the following command to take the snapshot of the Secondary volumes:

   ```
   /usr/sbin/vxrvg -g hrdg snapshot hr_rvg
   ```

   Note: This example does not need a `postfreeze` script.

   ---

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 2—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`, `onfreeze`, and `postfreeze` scripts and copy them in 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/vxassist -g hrdg snapstart hr_dv01
/usr/sbin/vxassist -g hrdg snapstart hr_dv02
```

In the `onfreeze` script, include the following command to take the snapshot of the Secondary volumes:

```
/usr/sbin/vxrvg -g hrdg snapshot hr_rvg
```

In the `postfreeze` script, include the following commands to back up the snapshot volumes and to reattach the snapshot plexes to the original volumes after the backup is complete:

```
 # Back up data from SNAP-hr_dv01 and SNAP-hr_dv02
 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
 # Reattach the snapshot volumes to the Secondary data
volumes
 /usr/sbin/vxrvg -g hrdg snapback 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 3—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 8

Transferring the Primary role

This chapter includes the following topics:

- About transferring the Primary role
- Migrating the Primary
- Taking over from an original Primary
- Failing back to the original Primary

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:

- "Migrating the Primary" on page 240.
- "Taking over from an original Primary" on page 247.
- "Failing back to the original Primary" on page 255.

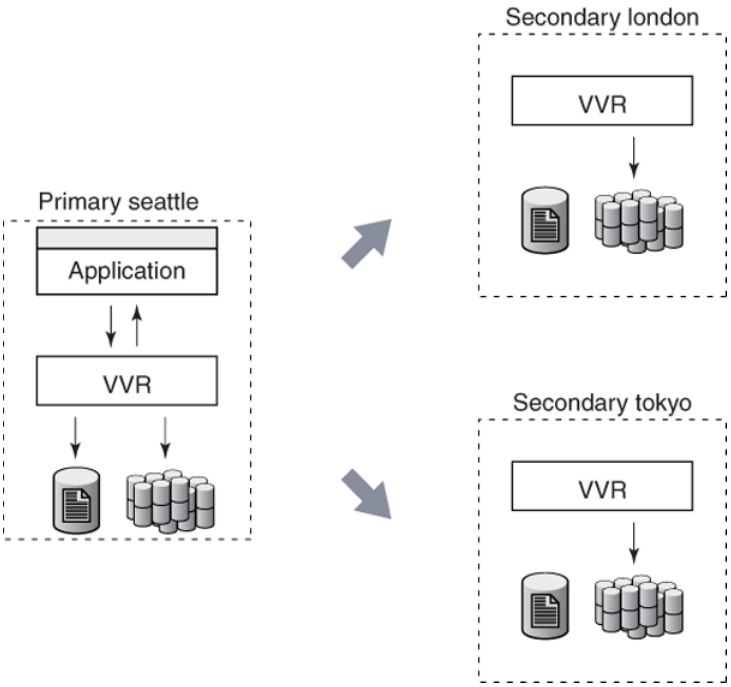
This chapter explains how to transfer the Primary role using each of these methods.

Note: If the RDS is configured in a Veritas Cluster Server (VCS) environment, use the `hagrp` command to offline and online the corresponding resources. For more information on offlining and onlining resources, see the *Veritas Cluster Server User's Guide*. If you are using the RVGPrimary agent for VVR to manage role transfer, refer to the *VCS Agents for VVR Configuration Guide*.

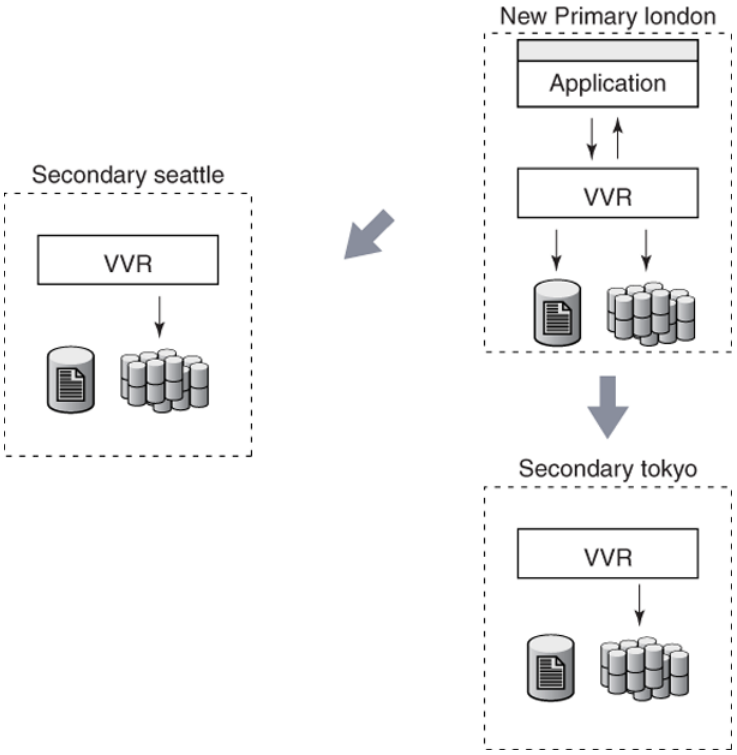
Migrating the Primary

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 RDS 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

Observe the following 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 "Resizing the SRL" on page 160.
- 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 391.
- 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 2–Migrating the Primary role in a setup with multiple Secondaries" on page 244.

Example 1—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	`seattle`	london
displayed by the vradmin printrvg command		
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 2—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_seattle_hr_rvg (active)`	`rlk_seattle_hr_rvg (active)`
	`rlk_tokyo_hr_rvg (active)`	`rlk_tokyo_hr_rvg`	`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 UDP/IP. If necessary, you can change the protocol to TCP/IP.

   See "Setting the network transport protocol" on page 77.

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.

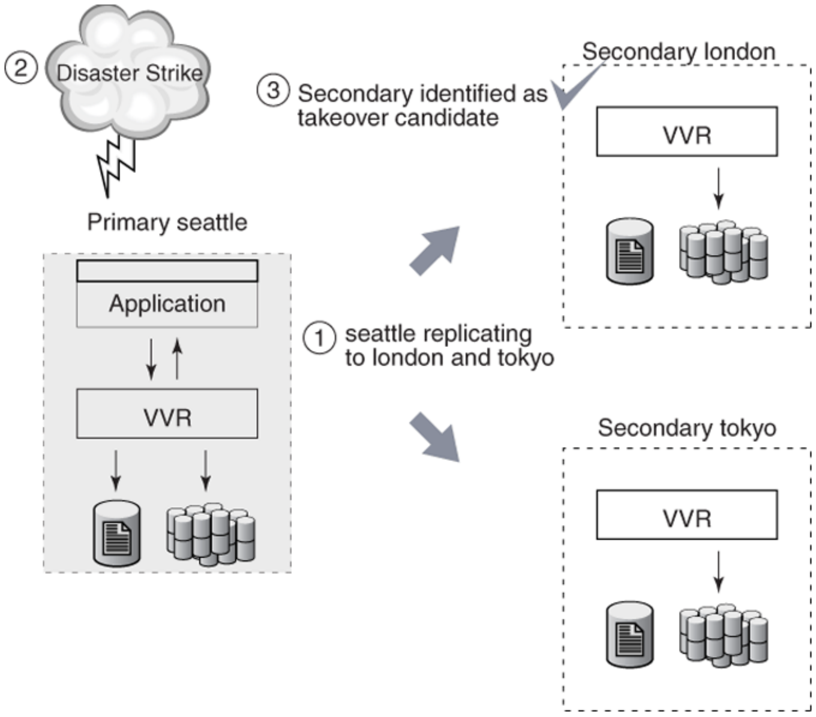
After migration, the configuration of the RDS looks like this:

	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_london_hr_rvg (active)`	rlk_london_hr_rvg `(active)`
	`rlk_tokyo_hr_rvg (active)`	`rlk_tokyo_hr_rvg`	`rlk_seattle_hr_rvg`

Taking over from an original Primary

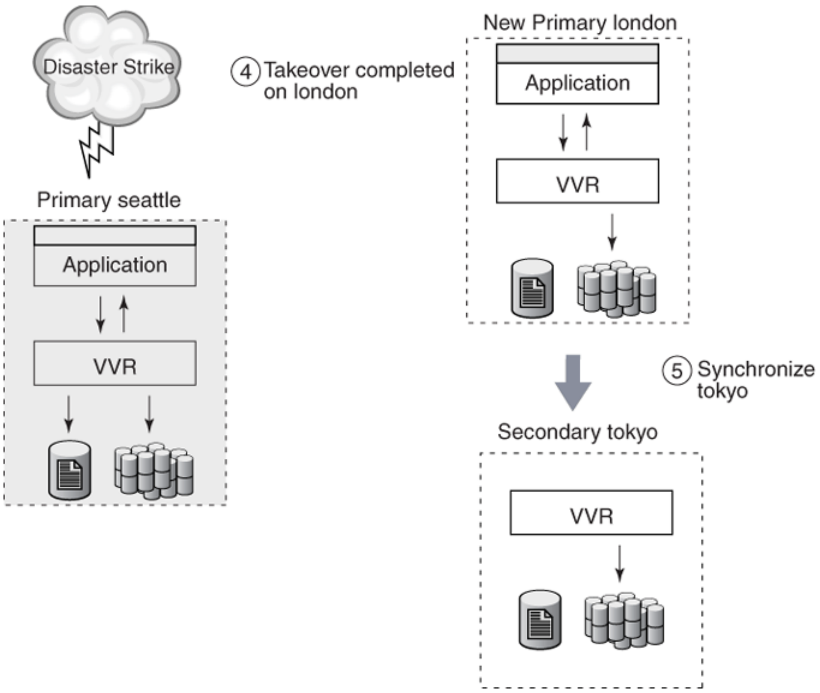
The takeover procedure involves transferring the Primary role from an original Primary to a Secondary. When the original Primary fails or is destroyed because of a disaster, the takeover procedure enables you to convert a consistent Secondary to a Primary. The takeover of a Primary role by a Secondary is useful when the Primary experiences unscheduled downtimes 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.


② Disaster Strike
Primary seattle
Application
VVR
③ Secondary identified as takeover candidate
Secondary london
VVR
① seattle replicating to london and tokyo
Secondary tokyo
VVR

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 132.

Important notes about taking over from an original Primary

Observe the following 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.

Example:

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 256.

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 2—Taking over from an original Primary in a setup with multiple Secondaries" on page 253.

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 1—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 2—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 UDP/IP. If necessary, change the protocol to TCP/IP.

   See "Setting the network transport protocol" on page 77.

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 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 checkpoint with the specified name, `checkpt`, in this example. After the data volumes are synchronized, the checkpoint is ended.

   - Start replication to `tokyo` using the 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:

- "Failing back using fast failback synchronization" on page 256.
- "Failing back using difference-based synchronization" on page 262.

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.

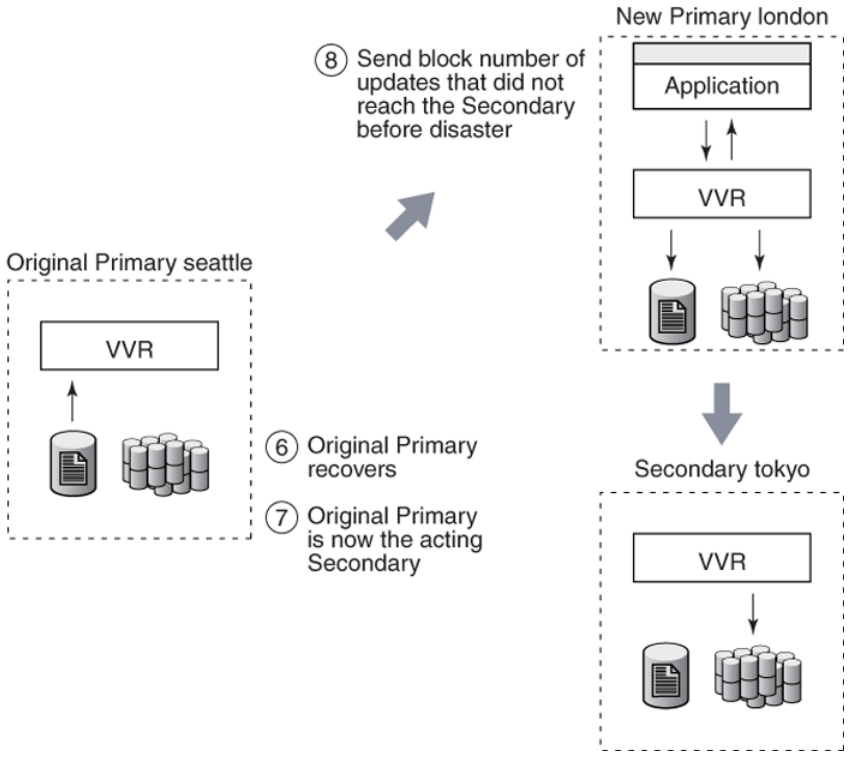
   It is possible that the Primary and Secondary data volumes are not up-to-date because all updates to the original Primary might not have reached the Secondary before the takeover. The failback process takes care of these writes by replaying the SRL or DCM of the original Primary. After the original Primary 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.

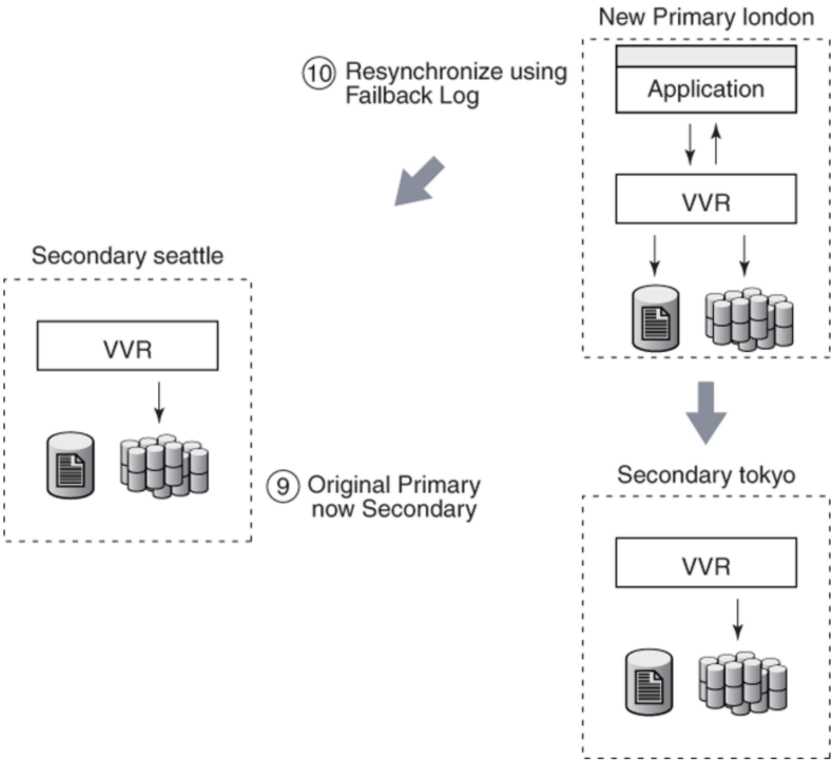
3. Migrating the Primary Role back to the original Primary and starting replication.

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.


New Primary london
Application
VVR
8 Send block number of updates that did not reach the Secondary before disaster
Original Primary seattle
VVR
6 Original Primary recovers
7 Original Primary is now the acting Secondary
Secondary tokyo
VVR

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 1—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 188.

   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 2—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 1—Failing back to the original Primary using fast failback" on page 259.

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 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 checkpoint with the specified name, `checkpt`, in this example. After the data volumes are synchronized, the checkpoint is ended.

- Start replication to `tokyo` using the 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

1. Converting the original Primary to a Secondary of the new Primary.
2. Synchronizing the data volumes on the original Primary with the data volumes on the new Primary using difference-based synchronization using checkpoint.
3. Starting replication to the Secondary (original Primary) using checkpoint.
4. Migrating the Primary Role to the original Primary and starting replication by default.

The examples in the following section explain how to fail back to the original Primary using VVR.

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 3—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 142.

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 checkpoint. To synchronize the Secondary based on differences using a 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 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 4—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 3—Failing back to the original Primary using difference-based synchronization" on page 263.

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 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 checkpoint with the specified name, `checkpt`, in this example. After the data volumes are synchronized, the 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.

Chapter 9

Replicating to a bunker

This chapter includes the following topics:

- Introduction to bunker replication
- Sample configuration
- Setting up bunker replication
- Administering bunker replication
- Using a bunker for disaster recovery
- Bunker replication in a VCS environment
- Removing a bunker
- Bunker commands

Introduction to bunker replication

Veritas Volume Replicator (VVR) offers different modes of replication, synchronous and asynchronous.

Bunker replication 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.

Bunker replication 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. Bunker replication 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 SRL 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.

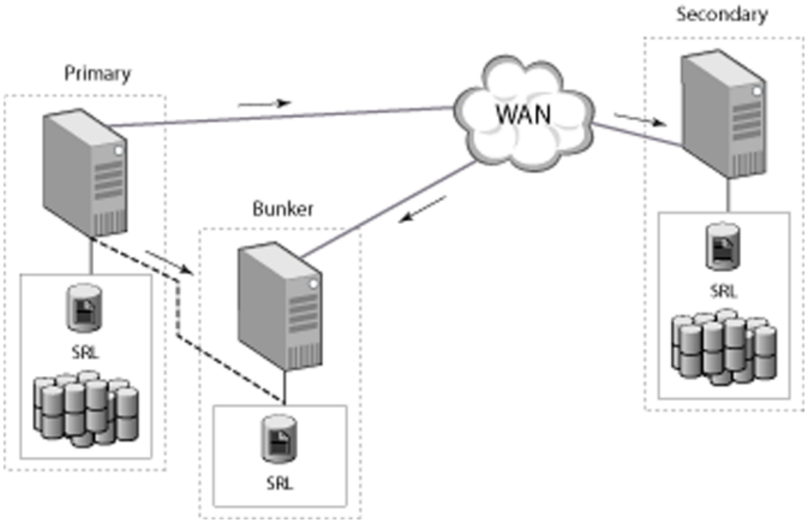
Bunker replication can be performed using an IP network or using direct connectivity to the storage, such as IP over Fiber Channel, Direct Attached Storage (DAS) or Network Attached Storage (NAS). If replication is performed over IP, the Primary host sends writes to a host at the bunker site, known as 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.

Bunker replication 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.

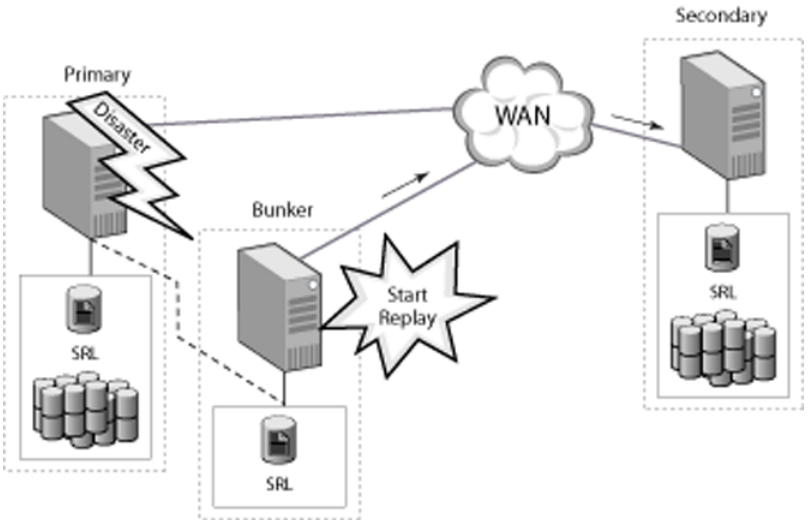


How the bunker is used for disaster recovery

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 on the bunker to update the Secondary. You activate the bunker, 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.

Bunker replication 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.

Using bunker replication, 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.

Sample configuration

The examples in this chapter assume the following configuration:

```
# vradmin printrvg
       Replicated Data Set: hr_rvg
       Primary:
               HostName: seattle
               RvgName: hr_rvg
               DgName: hrdg
       Secondary:
               HostName: london
               RvgName: hr_rvg
               DgName: hrdg
```

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 bunker replication

Setting up bunker replication involves the following steps:

- "Adding a bunker to an RDS" on page 272.
- "Changing replication settings for the bunker Secondary" on page 276.
- "Starting replication to the bunker" on page 278.

Requirements for bunker replication

Observe the following requirements for bunker replication:

- 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.
- In a shared disk group environment, you cannot create the bunker itself in a shared disk group. However, you can attach a bunker to an RDS for which the Primary RVG or a Secondary RVG is on a shared disk group.

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.

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 or a shared 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
     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`.

   In a shared disk group environment, import the bunker disk group on the logowner node of the Primary cluster.

   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
   ```

   In a shared disk group environment, the bunker disk group must be imported on the logowner node. If the VCS agents for VVR are configured, the bunker disk group fails over automatically when the logowner fails over to another node on the cluster.

   See “Automating local cluster failover for a bunker” on page 284.

   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 53.

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 Secondary, 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 `UDP`. 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 "The latencyprot attribute" on page 59.
`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 checkpoints; therefore, the bunker does not support attaching or resuming the Secondary from a 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 bunker replication

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 163.

To display the status of replication to a bunker, use the `vradmin repstatus` command.

See "Displaying consolidated replication status" on page 107.

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 279.

2. After the Secondary is up-to-date, the Secondary can take over the role of Primary.

   See "Taking over from an original Primary" on page 247.

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 281.

Updating the Secondary from the bunker

When disaster strikes and the Primary host goes down, the Secondary can be updated using the bunker. as described in this section.

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.

VVR commands relating to bunker replication 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

1. If the bunker is using the `STORAGE` protocol, import the disk group containing the bunker SRL onto the bunker host and then recover it. Issue the following commands on the bunker host:

   ```
   # vxdg -C import hrdg2
   # vxrecover -g hrdg2 -ns
   ```

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 "Taking over from an original Primary" on page 247.

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 281.
- If the original Primary recovers after the original Secondary has taken over the Primary role.
  See "Failing back to the original Primary" on page 283.

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 and recover the objects. Issue the following commands on the Primary host:

```
# vxdg import hrdg2
# vxrecover -g hrdg2 -ns
```

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 255.

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 283.

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 and recover the objects. Issue the following commands on the Primary host:

   ```
   # vxdg import hrdg2
   # vxrecover -g hrdg2 -ns
   ```

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
   ```

Bunker replication in a VCS environment

This section provides information about configuring and using bunker replication in a VCS environment.

Automating local cluster failover for a bunker

This section describes how to 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.

In a shared disk group environment, the application RVG and the bunker RVG must be online on the logowner host. If the logowner fails over, the bunker RVG must be deported from the original logowner host and imported on the new logowner host. In this case, the RVGLogowner agent handles the failover process.

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 above 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 the *Veritas Cluster Server Agents for Veritas Volume Replicator Configuration Guide* for more information.

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
```

Bunker replay in a VCS environment

The procedure for bunker replay in the VCS environment is the same as in the case of a non-clustered Primary.

Bunker replay in a VCS global cluster environment

Bunker replication does not support using the automated failover capability of a VCS global cluster. Because the data on the bunker cannot be used after the Secondary is converted to the Primary role, you must replay the bunker before the failover. If a VCS global cluster is configured on the Primary and Secondary clusters, make sure that the failover to the Secondary is not automated if a bunker is configured. To do this, set the `ClusterFailOverPolicy` attribute for the global group to `Manual`.

If the Primary cluster fails, update the Secondary from the bunker.

See "Updating the Secondary from the bunker" on page 279.

After the Secondary is updated, do the failover to the Secondary.

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
```

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
```

Troubleshooting VVR

This chapter includes the following topics:

■ Recovery from RLINK connect problems

■ Recovery from configuration errors

■ Recovery

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, issue the following command on the logowner on the Primary and Secondary:

  ```
  # ping -s remote_host
  ```

  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 -I 2 remote_host 8192
```

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 state of the heartbeat port is `Idle` by issuing the following command:

  ```
  # netstat -an -P udp
  ```

  The output looks similar to this:

  ```
  UDP: IPv4
      Local Address         Remote Address        State
      --------------------  --------------------  -------
      *.port-number                                  Idle
  ```

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

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-5-1-0 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 INFO V-5-1-3614 Secondary data volumes detected
     with rvg hr_rvg as parent:
VxVM VVR vxrlink INFO V-5-1-0 vol1: len=12800 primary_datavol=hr_dv01
VxVM VVR vxrlink INFO V-5-1-0 vol1: len=12800 primary_datavol=hr_dv02
VxVM VVR vxrlink INFO V-5-1-0 vol1: len=12800 primary_datavol=hr_dv03
```

Volume set errors

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" on page 297.

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

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

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

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

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" on page 296.

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:

```
    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:

```
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

This section describes how to recover from various types of disasters, such as a Primary host crash or an error on the Primary data volume.

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 1" on page 301.
- See "Example 2" on page 301.
- See "Example 3" on page 302.

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 4” on page 302.

Example 1

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 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 2

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 3

As an alternative to the procedures described in Example 1 and Example 2, the Primary role can be transferred to a Secondary host.

See “Migrating the Primary” on page 240.

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 4

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 "RVG PASSTHRU mode" on page 304.

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. Completely synchronize the Secondary.

   See "Synchronizing the Secondary and starting replication" on page 80.

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:

```
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 305.

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 update 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 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 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 the SRL be mirrored.

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 81.

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 checkpoint, detach the RLINKs on both the Primary and Secondary, and then synchronize the RLINKs.

See "Restoring the Secondary from online backup" on page 215.

Recovery using a Secondary checkpoint

This section explains how to recover from a Secondary data volume error using a Secondary 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 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 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 checkpoint backup to all data volumes. Unlike restoration from a Secondary checkpoint, the Primary checkpoint data must be loaded onto all Secondary data volumes, not just the failed volume. If a usable Primary checkpoint does not already exist, make a new checkpoint.

   See "Example—Synchronizing the Secondary using block-level backup" on page 88.

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 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 is 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 checkpoint backup is available, restore from the Secondary 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 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 checkpoint is available, synchronize the Secondary using a block-level backup and Primary checkpoint.

See “Example—Synchronizing the Secondary using block-level backup” on page 88.

As an alternative, you can also use automatic synchronization.

```
# vxrlink -g hrdg det rlk_seattle_hr_rvg
```

On the Secondary, restore the Primary 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-5-1-0 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-5-1-0 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-5-1-0 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/adm/messages` file for a message such as:

   ```
   WARNING: VxVM VVR vxio V-5-0-0 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.

VVR command reference

This appendix includes the following topics:

- Command reference

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`
`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`*

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`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 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.
`vradmin -g` *`diskgroup`* `-full -c` *`checkpt_name`* `syncrvg` *`local_rvgname sec_hostname....`*	Performs full synchronization with checkpoint to the Secondary when the application is active or inactive.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`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`*	Resizes a data volume in an RDS.
`vradmin -g` *`diskgroup`* `resizesrl` *`local_rvgname length`*	Resizes the SRL in an RDS.
`vradmin -g` *`diskgroup`* `delvol` *`local_rvgname volume_name`*	Removes a data volume from an RDS.
`vradmin -g` *`diskgroup`* `ibc` *`local_rvgname task_name`* `[`*`sec_host`*`]...`	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.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`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.
`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.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`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`* `[-1] getdatavols` *`rvg_name`*	Displays the names of all data volumes that are associated with the specified RVG.
`vxrvg -g` *`diskgroup`* `[-1] 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.
`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.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vxrvg -g` *`diskgroup`* `-c` *`checkpt_name`* `checkstart` *`rvg_name`*	Marks the beginning of a Primary checkpoint by pointing to the current location of the SRL.
`vxrvg -g` *`diskgroup`* `checkend` *`rvg_name`*	Marks the end of a Primary checkpoint by pointing to the current location of the SRL.
`vxrvg -c` *`checkpt_name`* `checkdelete` *`rvg_name`*	Deletes the specified Primary checkpoint.
`vxrvg -g` *`diskgroup`* `cplist` *`rvg_name`*	Displays information about all existing 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. 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.
`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.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`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, 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.
`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 checkpoint.
`vxrlink -c` *`checkpt_name`* `checkdelete` *`rlink_name`*	Deletes the specified Secondary checkpoint. **Note:** This command must be run only on the Primary.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`vxrlink -g` *`diskgroup`* `verify` *`rlink_name`*	Displays the configuration status of the given RLINK.
`vxrlink -g` *`diskgroup`* `[-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 checkpoints associated with the RLINK, which includes the name of the 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.
`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`.
`vxmemstat [-i` *`interval`* `[-t` *`count`*`]] [-e]`	Display memory statistics for Veritas Volume Manager.

Table A-1 VVR command reference *(continued)*

VVR Command	Command Description
`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 Veritas 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.

Messages

This appendix includes the following topics:

- Kernel messages
- Utility error messages
- The vradmin error messages
- Messages related to the vrstat command

Kernel messages

This section describes the diagnostic messages displayed by the VVR Kernel that appear on the console and in the `/var/adm/messages`file. Error messages are listed first, followed by a list of informational messages.

The Kernel messages are categorized by type, for example, RLINK, SRL and DCM, and communication errors.

Error messages

This section lists messages related to RLINKs, the SRL and the DCM, communication errors, configuration errors, I/O failure, shared objects, and kernel logging.

This section lists messages related to RLINKs, the SRL and the DCM, communication errors, configuration errors, I/O failure, shared objects, and kernel logging.

Messages related to RLINKs

Table B-1 shows messages related to RLINKs.

Table B-1 Messages related to RLINKs

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-114	Disconnecting rlink *rlink_name* to permit transaction to proceed.	Ignore message if seen occasionally. *Action:* If the message persists, contact Veritas Customer Support.
V-5-0-208	Log over 80% full for rlink *rlink_name*	This indicates that SRL is almost full. *Action:* Check that the `srlprot` attribute is set correctly in order to avoid failing of writes, throttling of writes or overflow of SRL.
V-5-0-267	Rlink *rlink_name* disconnecting due to ack timeout on *msg type* message	Ignore message if seen occasionally. *Action:* If the message persists, contact Veritas Customer Support.
V-5-0-329	Unable to connect rlink *rlink_name* on rvg *rvg_name*: Unknown error (*errno*)	VVR encountered an unknown error. *Action:* Contact Veritas Customer Support.
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Disk group or rlink not found on remote	The Primary RLINK indicates a Secondary disk group and RLINK combination that does not exist on the Secondary. *Action:* Ensure that the disk group and the RLINK specified by the Primary RLINK exist on the Secondary.
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Rlink detached on remote	The RLINK on the Secondary node is in the DETACHED or STALE state. *Action:* Attach the RLINK and retry.

Table B-1 Messages related to RLINKs *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Not ready on remote	VVR was not able to connect the named RLINK for the reason mentioned in the message. *Action:* Ignore message if seen occasionally. VVR automatically recovers from the error. If the errors persist, they may indicate a problem with the network configuration.
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Rlink already connected on remote	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Stream error on remote	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Checksum error on remote	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Unexpected command on remote	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Out of space on remote	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Port closing on remote	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Too many threads on remote	

Table B-1 Messages related to RLINKs *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Invalid port on remote	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Send error on remote	
V-5-0-330	Unable to connect rlink *rlink_name* on rvg *rvg_name*: KTLI connect failed	
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Time out on remote	The Secondary machine is unreachable. Will clear up when the network or the Secondary node recovers.
V-5-0-0	Disconnecting rlink *rlink_name* due to error in sending (*error-code*)	VVR disconnected the named RLINK for the reason mentioned in the message. *Action:* Ignore message if seen occasionally. VVR automatically recovers from the error. If the errors persist, they may indicate a problem with the network configuration.
V-5-0-0	Disconnecting rlink *rlink_name* due to loss of TCP connection	
V-5-0-0	Disconnecting rlink *rlink_name* due to stream error	
V-5-0-0	Disconnecting rlink *rlink_name* due to header checksum error	
V-5-0-0	Disconnecting rlink *rlink_name* due to unexpected message	
V-5-0-0	Disconnecting rlink *rlink_name* as Secondary data volumes are stopped	

Table B-1 Messages related to RLINKs *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-0	Disconnecting rlink *rlink_name* due to bad message.	
V-5-0-0	Disconnecting rlink *rlink_name* due to error : *error-code*	
V-5-0-0	Disconnecting rlink *rlink_name* remote already connected	
V-5-0-0	Disconnecting rlink *rlink_name* due to pending transaction	These errors are caused by transient network issues. *Action:* Ignore message if seen occasionally. VVR automatically recovers from the error. If the errors persist, they may indicate a problem with the network configuration.
V-5-0-0	Received from unexpected port. Expected from port *port*, received from port *port*	
V-5-0-0	Header checksum error	
V-5-0-0	Received unexpected message. Expected message with opcode *operation-code*, received opcode *operation-code*	
V-5-0-0	Data checksum error for handshake message id (*message-id*), data checksum computed (*checksum-value*) but header contains (*checksum-value*)	
V-5-0-0	Data checksum error. Received message id *message-id* with data checksum : *checksum-value* expected checksum : *checksum-value*	

Messages related to the SRL and DCM

Table B-2 shows messages related to the SRL and DCM.

Table B-2 Messages related to the SRL and DCM

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-100	DCM Logs not accessible, dcm logging aborted	DCM logs are not accessible. This might have happened due to a media failure, in which case, other errors may have been logged to the console. This error is unlikely to occur, because the default is to mirror the DCM.
V-5-0-102	DCM volume vol name is detached	The DCM volume became detached because a DCM log entry cannot be written. This might have happened due to a media failure, in which case other errors may have been logged to the console. Any RLINKs currently using the DCM will be detached and marked STALE. This error is unlikely to occur, because the default is to mirror the DCM.
V-5-0-107	Detaching rlink *rlink_name* due to I/O error on remote SRL during recovery	When the original Primary comes up after it has been taken over, an I/O error has occurred on the original Primary's SRL while performing recovery. The new Primary has detached the RLINK that connects to the failed Primary because now there is no way to make the old Primary consistent and convert it to a Secondary of the new Primary. *Action:* Fix the I/O error on the SRL, convert the original Primary into a Secondary of the new Primary, and perform a complete synchronization.

Table B-2 Messages related to the SRL and DCM *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-280	Rlink *rlink_name* stale due to log overflow	The specified RLINK has fallen too far behind, and overflowed the SRL. It will be detached and marked STALE. The Secondary will require a complete resynchronization. This can be avoided in the future by using the RLINK's `srlprot` attribute.
V-5-0-287	rvg *rvg_name*, SRL *srl*: Inconsistent log - detaching all rlinks	SRL contains corrupt data, and so is not usable. All RLINKs are detached and marked STALE. All secondaries will require a complete resynchronization. This error probably indicates a bug in VVR. *Action:* Contact Veritas Customer Support.
V-5-0-288	Secondary log overflowed. Pausing rlink *rlink_name*	Specified Secondary RLINK gets paused. *Action:* Associate the SRL of the same size as that of Primary to the Secondary RVG and resume the RLINK.
V-5-0-293	SRL for RVG *rvg_name* contains old version of SRL header	Specified SRL associated with the RVG is of older version. You may not have followed the proper procedure for upgrading VVR as described in Release Notes. *Action:* Refer to the latest Release Notes for the correct upgrading procedures.

Table B-2 Messages related to the SRL and DCM *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-1-435	Cannot allocate space for 20480 block volume or not enough disks for creating dcm with 2 mirrors	By default, the DCM is mirrored; therefore, it requires space on two disks. An attempt to associate a DCM to a new data volume or an existing data volume failed because sufficient space is not available. *Action:* We recommend that you provide the required space. Alternatively, you can specify `nlog=1` in the `vxassist make` or `vxassist addlog` command.

Messages related to communication errors

Table B-3 shows messages related to communication errors.

Table B-3 Messages related to communication errors

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-18	startdaemon_deferred: Could not create volkmsgd	Ignore message if seen occasionally. *Action:* If the message persists, contact Veritas Customer Support.
V-5-0-44	Cannot alloc bind structure (*errno*)	
V-5-0-45	Cannot alloc bind structure for listen server (*errno*)	
V-5-0-46	Cannot alloc connection indication call structure (*errno*)	
V-5-0-47	Cannot alloc ktli header (*errno*)	
V-5-0-40	Cannot bind the acceptor (*errno*)	

Table B-3 Messages related to communication errors *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-49	Cannot connect rlink *rlink_name* due to protocol mismatch: remote rlink is using *protocol_name* protocol, local rlink is using *protocol_name* protocol	VVR is not able to communicate with the remote host because the selected protocol differs between local and remote hosts. *Action:* Verify that the Primary and Secondary RLINKs are set to use the same network communication protocol. See "Setting the network transport protocol" on page 77.
V-5-0-50	Cannot connect to host *ip address* port *port_number* (*errno*)	You may see this message when `vxnetd` has not been run on the remote host. *Action:* Run `vxnetd` by issuing the command: `/etc/init.d/vxnm-vxnetd` If the message persists, contact Veritas Customer Support.
V-5-0-51	Cannot connect to remote due to header format mismatch or checksum error	*Action:* Contact Veritas Customer Support.
V-5-0-54	Cannot find any free port to bind	All the ports in the specified list have been utilized. *Action:* Add the ports to be used for replication using the `vrport` command.
V-5-0-75	Cannot open ktli (*errno*)	Ignore message if seen occasionally. *Action:* If the message persists, contact Veritas Customer Support.
V-5-0-76	Cannot open ktli for listen server (*errno*)	

Table B-3 Messages related to communication errors *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-77	Cannot open the acceptor port (*errno*)	
V-5-0-78	Cannot open the server port (*errno*)	
V-5-0-80	Cannot send out a message	
V-5-0-84	Cannot spawn server thread	Ignore message if seen occasionally. *Action:* If the message persists, contact Veritas Customer Support.
V-5-0-611	Could not send heartbeat *id* to node *node*	
V-5-0-174	header checksum error	
V-5-0-175	Heartbeat unacknowledged from node *ip-addr* for t seconds	
V-5-0-206	listen server port in use (*errno*)	Port assigned to VVR is in use by another application. *Action:* Change the assigned port using the `vrport` command.
V-5-0-219	nmcom client cannot bind ktli port (*errno*)	Ignore message if seen occasionally. *Action:* If the message persists, contact Veritas Customer Support.
V-5-0-220	nmcom client cannot open ktli port (*errno*)	

Table B-3 Messages related to communication errors *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-236	Number of ports available (*total-port*) is less than the number of rlinks (*numreplicas*) in the system	The specified number of ports are less than total number of RLINKs in the system. Some of the RLINKs will be disconnected. *Action:* For all RLINKs that are to take part in replication, configure at least *n* number of ports using the `vrport` command, where *n* is *>=num_of_rlink* in the system.
V-5-0-246	port *port-id* is in use by another application	Port assigned to VVR is in use by another application. *Action:* Change the assigned port using the `vrport` command.
V-5-0-247	port in use (*port id*)	Port assigned to VVR is in use by another application. *Action:* Change the assigned port using the `vrport` command.
V-5-0-253	Received 100 duplicate packets. Check network configuration	Indicates a possible network misconfiguration, such as multiple NICs with the same address. *Action:* Check network configuration and make sure that the IP assigned to the RLINK is unique on the system.

Table B-3 Messages related to communication errors *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-376	VVR listener thread exiting	If VVR is using the TCP protocol, a thread listens for incoming RLINK connections from remote hosts. When this thread encounters a fatal error, it exits with this message. *Action:* Run `vxnetd` by issuing the following command: `/etc/init.d/vxnm-vxnetd` If the message persists, contact Veritas Customer Support.
V-5-0-0	2a: Received invalid message block in TCP stream	Ignore message if seen occasionally. *Action:* If the message persists, contact Veritas Customer Support.
V-5-0-0	2b: Received invalid message block in TCP stream	
V-5-0-0	2c: Received invalid message block in TCP stream	
V-5-0-0	2d: Received invalid message block in TCP stream	
V-5-0-0	Could not save message block, TCP stream error	
V-5-3-0	Received invalid message block in TCP stream	

Messages related to configuration errors

Table B-4 shows messages related to configuration errors.

Table B-4 Messages related to configuration errors

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-29	Configuration error on rvg *rvg_name* pausing rlink *rlink_name*	Indicates configuration error. *Action:* Clear the mentioned error.
V-5-0-30	Cannot find matching volume for volume *pri_datavol* on primary	
V-5-0-31	Name on Secondary *datavol_name* does not match name on primary *pri_datavol*	
V-5-0-32	Size of volume *pri_datavol* on primary (*vol_size*) does not match size on Secondary (*sec_vol_sz*)	
V-5-0-511	Replica has an unsupported message format	Remote host is running older version of VVR. The RLINKs will go into the STALE state. *Action:* For replication to take place, all the hosts in the VVR configuration must be running the same version of VVR. The exception is during certain upgrade scenarios that enable you to upgrade the Primary and the Secondary at different times. This type of upgrade is usually supported only between the current release and the immediately prior release.
V-5-0-512	Replica is running an unsupported version	
V-5-0-858	Received from unexpected port	These errors are caused by transient network issues. VVR handles these errors so they can be ignored if seen occasionally. If the errors persist, it may indicate a problem with the network configuration.

Table B-4 Messages related to configuration errors *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-859	Received unexpected message	
V-5-1-10128	Operation requires transaction	Indicates that another configuration change is in progress. *Action:* Issue the command again.

Message related to I/O failure

Table B-5 shows messages related to I/O failure.

Table B-5 Messages related to I/O failure

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-167	Failing writes on rvg *rvg_name*. Remote is now a primary rvg.	When the original Primary comes up after it has been taken over and the application on the original Primary tries to perform writes, the writes will fail because this RVG is no longer the Primary. *Action:* Stop the application on the original Primary.

Messages related to shared objects

Table B-6 shows messages related to shared objects.

Table B-6 Messages related to shared objects

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-232	Node *nodename* is logowner for Rvg *rvg_name*	The specified node has become the logowner for the specified RVG. Action: No action required.

Table B-6 Messages related to shared objects *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-350	vol_rv_ioctl: rvg *rvg_name* already has owner	An attempt to set the logowner of an RVG failed because the RVG already has a logowner. Action: Clear the logowner on the node where it currently resides first.
V-5-0- 439	vol_rv_send_request_start: unknown type	An illegal call was made to send a request to the cluster master. Action: Contact Veritas Customer Support.
V-5-0-352	vol_rv_wrship_start: Corrupted memory list	Memory corruption occurred during a remote write. Action: Contact Veritas Customer Support.
V-5-0-79	cannot recover vol *vol_name*	A memory allocation failure occurred during a cluster reconfiguration. Action: Contact Veritas Customer Support.

Messages related to bunker replication

Table B-7 shows messages related to bunker replication.

Table B-7 Messages related to bunker replication

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-1008	Cannot connect rlink due to protocol mismatch with	For STORAGE rlink the protcol field of the other rlink is not set to STORAGE, hence rlink can not be connected. *Action:* Set the protocol of both the rlink as STORAGE.

Table B-7 Messages related to bunker replication *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-1009	Cannot connect rlink to remote rlink since secondary rvg %s is not ready.	The Secondary rvg has stopped volumes, hence cannot connect the rlink. Action: Start the volumes in the secondary rvg and then connect the rlink.
V-5-0-0	Size of primary SRL ("VOFFDSTR") does not match size wth secondary SRL"	Indicates that the size of the SRL on the bunker secondary does not match the size of the SRL on the Primary. Action: Make the bunker Secondary SRL size same as primary SRL size and then reconnect the Primary to bunker rlink.
V-5-0-0	Disconnecting rlink %s to reconnect from new position,	When a Secondary switches connection from Primary rvg to bunker Primary or vice versa, then after the first connect it disconnects the rlink to reconnect from new position. Action: No action is required. It will reconnect in the next try from new position. If it happens frequently then contact Symantec consultant.

Messages related to kernel logging

Table B-8 shows messages related to kernel logging.

Table B-8 Messages related to kernel logging

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-59	cannot log *error* for rvg *rvg_name*	The messages in this set mean that VVR attempted to log the specified event in the kernel log; however, the attempt failed. The attempted write to the log failed either because the kernel log is full or because of a write error to the drive. Action: If error messages were seen from the disk driver, it is likely that the last copy of the log failed due to a disk error. The failed drive in the disk group must be replaced and the log will then be re-initialized on the new drive. If there were no error messages from the disk driver, contact Veritas Customer Support.
V-5-0-62	cannot log atomic commit changes for rvg *rvg_name*	
V-5-0-63	cannot log changes for rvg *rvg_name*	
V-5-0-65	cannot log datavol error for rvg *rvg_name*	
V-5-0-68	cannot log error during failback for rvg *rvg_name*	
V-5-0-70	cannot log srl error for rvg *rvg_name*	
V-5-0-71	cannot log srl error for rvg *rvg_name*	
V-5-0-72	cannot log unfreeze error for rvg *rvg_name*	
V-5-0-73	cannot log update commit state for rvg *rvg_name*	

Table B-8 Messages related to kernel logging *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-95	could not log config error for rvg *rvg_name*	
V-5-0-160	failed to flush log on detach of rvg *rvg_name*	The messages in this set mean that VVR logged the event(s) specified in the kernel log but was unable to flush the log. Action: Contact Veritas Customer Support.
V-5-0-161	failed to flush log on errors on rvg *rvg_name*	
V-5-0-165	Failed to log the detach of the DCM volume *vol name*	The messages in this set mean that VVR attempted to log the specified event in the kernel log; however, the attempt failed. The attempted write to the log failed either because the kernel log is full or because of a write error to the drive. Action: If error messages were seen from the disk driver, it is likely that the last copy of the log failed due to a disk error. The failed drive in the disk group must be replaced and the log will then be re-initialized on the new drive. If there were no error messages from the disk driver, contact Veritas Customer Support.
V-5-0-348	vol_rp_set_state_atomic_commit_done: cannot log atomic commit state for rvg *rvg_name*	
V-5-0-349	vol_rp_set_state_atomic_commit_start: cannot log atomic commit state for rvg *rvg_names*	

Informational messages

The messages that appear in this section do not require an action. For the most part, they are informational.

Informational RLINK messages

Table B-9 shows informational RLINK messages.

Table B-9 Informational RLINK messages

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-113	Disconnecting rep *rlink_name* to shift to DCM protection	This indicates a temporary disconnect during shifting to DCM protection.
V-5-0-149	Encountered IBC while flushing SRL to DCM - IBC dropped	The IBC residing on the SRL will be dropped when DCM protection gets triggered. You need to reissue the IBC after the DCM replay completes.
V-5-0-265	Rlink *rlink_name* connected to remote	The named RLINK has succeeded in connecting to its remote RLINK.
V-5-0-266	Rlink *rlink_name* disconnected from remote	The named RLINK has disconnected from its remote RLINK. This can have many different causes, including: ■ The network is down ■ The Secondary node is down ■ The Primary RLINK was paused or detached ■ The Secondary RLINK was detached The RLINK will connect automatically when the situation clears.
V-5-0-270	Rlink *rlink_name* has a Secondary config error	Indicates different states of the specified RLINK.
V-5-0-271	Rlink *rlink_name* has a Secondary log error	Indicates different states of the specified RLINK.

Table B-9 Informational RLINK messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-274	Rlink *rlink_name* is in failed state	Indicates different states of the specified RLINK.
V-5-0-275	Rlink *rlink_name* is inconsistent	Indicates different states of the specified RLINK.
V-5-0-276	Rlink *rlink_name* is primary paused	Indicates different states of the specified RLINK.
V-5-0-277	Rlink *rlink_name* is Secondary paused	Indicates different states of the specified RLINK.
V-5-0-278	Rlink *rlink_name* is stale and not replicating	Indicates different states of the specified RLINK.
V-5-0-330	Unable to connect to rlink *rlink_name* on rvg *rvg_name*: Time out on remote	The Secondary machine is unreachable. Will clear up when the network or the Secondary node recovers.
V-5-0-467	Paused replay on RVG `rvg_name`	The DCM replay gets paused on Primary. This can have many different causes, including: ■ The network is down ■ The Secondary node is down ■ The Primary RLINK was paused or detached. ■ The Secondary RLINK was paused or detached. DCM replay will start automatically when the RLINK gets connected.
V-5-0-0	Disconnecting rlink *rlink_name* due to kernel reset	This may happen due to a user initiated action, which causes the kernel objects to be recreated.
V-5-0-0	Disconnecting rlink *rlink_name* to permit transaction to proceed	The RLINK may get temporarily disconnected if a transaction is in progress. Once the transactions is completed, the RLINK will get reconnected.

Informational SRL and DCM messages

Table B-10 shows informational SRL and DCM messages.

Table B-10 Informational SRL and DCM messages

Unique Message Identifier (UMI)	Message	Message definition
V-5-0-79	cannot recover vol *vol_name*	During a reboot, the specified data volume was found to be disabled, and so cannot be recovered properly. The data volume may no longer be considered valid.
V-5-0-101	DCM replay complete on rlink *rlink_name*	DCM replay gets completed on the specified RLINK.
V-5-3-0	Resumed replay on RVG *rvg_name*	DCM replay gets resumed on the Primary. DCM replay will start automatically when the RLINK gets connected.

Utility error messages

Utility messages are issued by the `vxconfigd` configuration daemon and are displayed on the terminal where the command was executed. The messages listed here are not all-inclusive. In most cases, the messages are self-explanatory. Only those messages that require a message definition are listed here.

Table B-11 Utility messages

Message	Message definition
Out of kernel memory	The user request failed because the kernel could not allocate sufficient memory to complete the request successfully. If possible, attempt the request again. If it continues to fail, this may indicate that the system has insufficient resources.
IBC error	On the Secondary, an attempt was made to dissociate or delete an RLINK from an RVG against which there are outstanding IBC receive ioctls. All such outstanding ioctls must complete before the request will be processed.

Table B-11 Utility messages *(continued)*

Message	Message definition
Duplicate master_datavol mapping	Two volumes on the Secondary are mapped to the same volume on the Primary.
Multiple attached rlinks on Secondary	There could be some error during the recovery. The Secondary can only have one active RLINK. Clear the utility fields and then dissociate any inactive RLINKs.
Replicated volume may not have a drl	An attempt was made to associate a volume with the DRL log to an RVG. A volume with a DRL log is not allowed to be associated to an RVG.
SRL cannot be resized	An attempt was made to resize the SRL. The resize operation is not allowed if the volume is an SRL.
V-5-1-3265 WARNING: DCM log size is smaller than recommended due to increased volume size.	To resize the DCM, remove it and use the `vxassist addlog` command to set the correct size.
V-5-1-3265 WARNING: DCM log size will be smaller than recommended due to increased volume size.	Remove DCMs from volume and add them back specifying no `loglen` attribute, to get the default (recommended) size. Then re-issue the `vxassist maxgrow` command to find out how large the volume can be grown with the new DCM size.
VSet can not be associated to RVG for dg version less than 140	Indicates that the disk group for the RVG has a previous version that does not support associated volume sets. To correct this error, upgrade the disk group using the following command: `# vxdg upgrade` *`diskgroup`*
Rlink cannot be specified for making bunker secondary	Indicates that an RLINK name was specified to the make Secondary command for a bunker RVG. Action: Do not specify the RLINK name in the makesecondary command for a bunker RVG.
Cannot make bunker rvg %s a secondary because it has one attached rlink	A bunker Primary RVG cannot be converted to bunker Secondary RVG when it has an attached rlink. First detach the bunker primary to the secondary rlink and then convert bunker primary to bunker secondary.

Table B-11 Utility messages *(continued)*

Message	Message definition
-b option is valid only for attaching rlink from bunker primary rvg.	"-b" option can only be used to connect rlink from bunker primary to secondary.
Only -a option is valid for attaching bunker rlinks from non-bunker primary	"-a" option can only be used to connect rlink from primary to bunker secondary.
Rlink %s cannot be attached since bunker SRL doesn't contain updates to make secondary consistent after atomic commit.	Bunker site does not have enough data to perform replication to secondary site, hence it can not be used to perform replication to secondary site. This is possible only in certain limited situations where bunker site is not uptodate as primary due to network outage between primary and bunker or bunker site down for some other reasons.
Rlink %s cannot be attached since Bunker srl does not contain the update expected by secondary.	
Rlink %s cannot be attached because secondary has received more uptodate writes already	Secondary site has received more uptodate data than bunker site, hence bunker site can not be used to recover the secondary.

The vradmin error messages

This section lists and describes some of the error messages that may be displayed by `vradmin`, if a problem occurs. It also suggests the appropriate action that needs to be taken if a problem occurs.

Table B-12 vradmin error messages

Unique Message Identifier (UMI)	Message	Message definition
V-5-36-2086	Server volume access error during [assign volids] volume path: *`[/dev/vx/dsk/dgname/volumename]`*	This error may occur while performing the `vradmin syncvol` command if the remote volume, that is, the volume being synchronized, is not enabled or available for read and write. Action: Make sure the remote volume is available for read and write. Start the volume if it is currently stopped. If the volume is part of an RVG, make sure the RVG or the replication (i.e. RLINK) is in a certain state which allows read/write to the data volume.
V-5-36-2125	Server volume access error during [assign volids] volume path: */dev/vx/dsk/dgname/ volumename* reason: [this could be because a target volume is disabled or an rlink associated with a target volume is not detached during sync operation]	This error may occur while performing the `vradmin syncvol` command if the remote volume, that is, the volume being synchronized, is not enabled or available for read and write. *Action:* Make sure the remote volume is available for read and write. Start the volume if it is currently stopped. If the volume is part of an RVG, make sure the remote volume is in ENABLED state and the RLINK associated to the remote volume is in DETACHED state.
V-5-52-12	`vradmind` server not running on this system.	Action: Start the `vradmind` server by running: `/etc/init.d/vras-vradmind.sh start` Then, run the command again.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-16	`vradmind` stopped running - Exiting.	The `vradmind` server got terminated during command execution. This could be due to an administrative action or due to some problem in `vradmind`. Action: Contact Veritas Customer Support to look into the problem. In the mean time, restart `vradmind` to resume normal operations.
V-5-84-162	Terminating the collection of statistics.	This error may occur while trying to run the `vrstat` command on the Secondary, if the `vradmind` server on the Primary was terminated while the `vrstat` command was in progress. This could be due to an administrative action or due to some problem in `vradmind`. Action: If you had stopped the `vradmind` server manually, then restart it and run the vrstat command again. If the `vradmind` server stopped due to some problem, contact Veritas Customer Support.
V-5-52-242	Attribute `cachesize` is not allowed with the `-k cache` option.	When cache objects are created using the `cachesize` option they get deleted when snapshots are deleted. Action: Either use the `-k snap` option or create a `cache` object and use the cache attribute.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-401	RDS *rds* is processing another command or a configuration change.	An attempt to run a `vradmin` command on RDS *rds* failed because another `vradmin` command is already running on this RDS or the RDS is undergoing a configuration change. Action: Retry running the command that you had specified after some time.
V-5-52-405	Primary `vradmind` server disconnected.	The `vradmind` server on Primary is not running. Action: Start the `vradmind` server on Primary.
V-5-52-418	Volumes on host *host* are incorrectly mapped.	One or more data volumes on the Primary do not have corresponding Secondary data volumes mapped to it. Action: Check the volume mapping between the Primary and the corresponding Secondary using `vradmin printvol` command. There must be one-to-one mapping between Primary and Secondary data volumes.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-421	`vradmind` server on host *host* not responding or hostname cannot be resolved.	The `vradmind` server on the host *host* is not running, or it's running on a different port, or *host* is not reachable. Action: Make sure *host* is the correct host name and is reachable. Also make sure that `vradmind` is running on *host* and is using the same port as that on the Primary. Use `vrport` command to check and/or set the ports.
V-5-52-422	Primary disk group *dg* is not authenticated.	Before adding a new Secondary (using `addsec`) or overwriting data on a remote host (using syncvol), `vradmin` performs some authentication. This is done by checking that the `/etc/vx/vras/.rdg` file on the remote host contains an entry for the Primary disk group ID. The `vradmin addsec` or syncvol command fails if `/etc/vx/vras/.rdg` file on the remote host does not have such an entry. Action: Add the Primary disk group ID to the `/etc/vx/vras/.rdg` file on the remote host. To find disk group ID, run the `vxprint -l diskgroup_name` command on the Primary.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-429	Secondary is processing a configuration change or its Primary is still reachable.	This error may occur only when performing the `vradmin takeover` command when the Primary is still reachable from the Secondary or when the Secondary is undergoing a configuration change. To know if the Primary is reachable or not from this Secondary, run `vradmin -l printrvg` *`rvg_name`* . If the output of the `printrvg` command shows the `Pri or Sec IP not available or vradmind not running` or the `No Primary RVG` error in the Config Errors, it indicates that the Primary is not reachable from this Secondary. Action: Check whether the Primary is reachable. If it is reachable then takeover cannot take place. In such a case, run the `vradmin migrate` command instead, to migrate the role of the Primary. If the Secondary is undergoing a configuration change, retry running the command after some time.
V-5-52-447	RDS has configuration error. Check information about this RDS.	Some `vradmin` commands will not proceed if the RDS has any configuration error(s). Action: Check the Config Errors listing for the specific RDS in the output of the `vradmin -l printrvg` command to identify the errors. Correct the errors and then retry running the command.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-448	Secondary on *sec_host* has configuration error. Check information about this RDS.	Some `vradmin` commands will not proceed if the specified Secondary has any configuration error(s). (The command will still proceed if any of the other Secondary RVGs in the RDS has a configuration error). Action: Check the Config Errors listing for the specific RDS in the output of the `vradmin -l printrvg` command to identify the error. Correct the error and then retry running the command.
V-5-52-449	Secondary *rvg_name* does not have an active Primary.	The Secondary cannot determine its Primary due to one or more of the following reasons: ■ it has some configuration error. ■ the Primary is not reachable. ■ it does not have any RLINK. Action: Run the `vradmin -l printrvg` on this Secondary to verify the cause of the problem. Correct the configuration error reported in the output of the `printrvg` command and then retry running the command.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-451	Volumes `vol_name` on Primary and Secondary differ in size; +/- not allowed in the size specification.	When Primary and Secondary data volume `vol_name` differ in size then specifying a relative size using +/- is not allowed in the `vradmin resizevol` command. Use the `vradmin printvol` command to verify the size of the Primary and Secondary data volume. Action: Specify absolute size for the volume instead of a relative size in the `vradmin resizevol` command.
V-5-52-456	Volumes in RVG `rvg_name` do not have DCMs required for failback logging.	This error occurs if the `vradmin takeover` command is run with failback logging enabled but at least one of the data volumes in the Secondary RVG does not have DCM. Failback logging requires that all volumes must have DCMs. Action: If you want to run the `takeover` command with failback logging enabled, first add DCMs to all the data volumes. Then, run the `takeover` command. If you do not want to have failback logging enabled in the `takeover` command, then use it with the `-N` option. This requires either full synchronization or difference-based synchronization of the data volumes on the original Primary.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-465	Primary RVG *rvg_name* still has one or more Secondary RVGs.	The `vradmin delpri` command performs the required deletion only when Primary RVG does not have any configured Secondaries. Action: Before deleting a Primary RVG, you must delete all the Secondaries. Use the `vradmin delsec` command to delete a Secondary.
V-5-52-467	Script *script_name* does not exist.	If the missing script is the `onfreeze` script, then it must exist on the Secondary host, where the off-host processing task is performed. Action: Put the `onfreeze` script on the Secondary. Make sure that the script is located in the following directory: */etc/vx/vvr/ibc_scripts/task_name* Retry running the `vradmin ibc` command.
V-5-52-468	*script_name* is not executable.	Action: Change the permission for the script *script_name* to make it executable by root user.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition

V-5-52-469	Script *script_1* is provided; script *script_2* must also be provided.	The `vradmin ibc` command requires that both the quiesce and the unquiesce script, if provided, must be present together on the Primary. If you only need one script, you will still need to provide the other one so that the `vradmin ibc` command executes properly. Therefore, for the script that you don't need create an empty script, that is, a script which contains only the following lines:
		
		`#!/sbin/sh`
		`exit 0`
		
		Action: Create *script_2* on the Primary and retry running the `ibc` command.

V-5-52-471	*host* has failback flag set.	Failback logging was enabled when the Secondary on *host* took over as a Primary. When the failback logging is enabled you cannot use the `makesec` command to convert original Primary to Secondary. Action: Use the `vradmin fbsync` command to convert the original Primary to a Secondary.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-478	Cannot perform incremental synchronization: RLINK *rlink_name* on host *host* not in CONNECT state.	The `vradmin fbsync` command will proceed only if the RLINK *rlink_name* on the original Primary *host* is attached. This error occurs when the RLINK *rlink_name* somehow gets paused or is not in connect state. Action: Check the RLINK *rlink_name* on the original Primary *host*. If the RLINK is paused, resume the RLINK then retry running the `vradmin fbsync` command. If it is already detached, then failback synchronization cannot be performed. In this case you will need to use the `vradmin makesec` command to convert the original Primary to Secondary of the new Primary and then synchronize the original Primary (new Secondary) with the new Primary.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-479	Host *host* not reachable.	This error generally occurs in either of the following: ■ network connection to host *host* is down. ■ `vradmind` server on *host* is not running or is running on a different port. Action: Correct the network if it's a network connection problem. In the case the `vradmind` server is not running start `vradmind` on host *host* using `/etc/init.d/vras-vradmind.sh start` OR Make sure `vradmind` on *host* runs on the same port as the local `vradmind`. Use `vrport` command to check and reset the port.
V-5-52-480	Operation requires a disk group.	The command for which the specified operation failed requires a valid disk group to be present.
V-5-52-481	Cannot perform the operation in a cross-version replication environment.	In a cross-version replication environment, operations which can result in configuration changes, like `vradmin addvol, delvol` and `resizevol`, are not allowed. Action: Upgrade VVR on all the hosts running earlier version of VVR to the same version of VVR running on the other hosts in the RDS.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-482	Cannot perform the operation: Secondary has an earlier version of VVR.	This error may occur in a cross-version replication environment while: ■ Performing the `vradmin addsec` command to add a host running an earlier version of VVR to an RDS which has at least one host running a later version of VVR. ■ Performing the `vradmin syncvol` command to synchronize volumes to a host running an earlier version of VVR. Action: Upgrade VVR version on the host to the later version of VVR.
V-5-52-483	Cannot perform the operation: the cross-version feature of VVR does not support the VVR version installed on *host*.	The cross-version replication in VVR is only supported between two immediate major releases. Action: Upgrade the VVR version on the host *host* to the same VVR version as on the other hosts in the RDS.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-489	Cannot determine local IP address: multiple IP addresses configured for the Primary RVG.	This error occurs when running the `vradmin syncrvg` command to multiple Secondaries for an RDS in which the local_host field of the Primary RLINKs resolve to different IP addresses. Action: If possible, make the local_host field of all the Primary RLINKs the same. If you still need to configure different interfaces for the Primary RLINKs, then run the `vradmin syncrvg` command to synchronize one Secondary at a time.
V-5-52-491	Cannot perform the operation: *host* is not a Primary (acting secondary).	The `vradmin fbsync` command requires that the specified host be an acting Secondary. Action: Check the `vradmin printrvg` output to see whether the host *host* is an acting Secondary.
V-5-52-492	*host* is an acting secondary. Complete the failback synchronization before making any configuration changes.	When an RDS has an acting Secondary, no other configuration command is allowed on this RDS except the `vradmin fbsync` command. Action: Run the `vradmin fbsync` command to convert the acting Secondary to a Secondary before making any configuration changes.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-493	Cannot perform the operation: none of the Secondaries is in DCM logging mode.	None of the Primary RLINKs are in DCM logging mode, hence the resync operation cannot be performed. Action: none
V-5-52-494	The command is not available in earlier version of VVR.	This message results when a new command, added in a later VVR version, is run in a cross-version replication environment. Action: To be able to make use of the new commands, make sure that VVR on all hosts in the RDS is upgraded to the appropriate higher version.
V-5-52-502	Host name or IP *host* is not configured or available on the Primary.	This happens during `addsec` command, when the host name or IP address specified for the Primary host is not configured or unavailable for use. Action: Make sure the hostname or IP address specified for the Primary host is correct and reachable.
V-5-52-604	Primary RLINK *rlink_name* not up-to-date.	The command for which this error occurred requires the Primary RLINK *rlink_name* to be up-to-date. Action: Verify that Primary RLINK *rlink_name* is up-to-date before running the command. Use `vradmin repstatus rvg` command or `vxrlink status rlk_name` command to verify current status of the RLINK.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-605	RLINK *rlink_name* in CONNECT state.	The command for which this error occurred requires that the Primary RLINK *rlink_name* must not be in CONNECT state. Action: Use `vradmin stoprep` command to stop replication.
V-5-52-609	Volume *vol_name* in disk group *dg_name* not of equal length as Primary volume.	The `vradmin addsec` and addvol commands require that the Secondary data volume(s) must be of the same size as the corresponding Primary data volume(s). Action: Resize either the Secondary or Primary data volume *vol_name* to make them of the same size. Then retry running the command. Use the `vxresize` command to resize a data volume.
V-5-52-610	Primary RLINK *rlink_name* not in CONNECT state.	The command for which this error occurred requires that the Primary RLINK *rlink_name* must be in CONNECT state. Action: If the RLINK is not attached, use the `vradmin startrep` command to start replication. If the RLINK is already attached, but not in CONNECT state, identify and correct the problem. Recovery from RLINK connect problems

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-52-611	RLINK *rlink_name* is inconsistent or failed.	This error occurs when the specified Secondary data is inconsistent as a result of which the specific Secondary cannot be converted to Primary. Action: Use another Secondary which is consistent or restore a consistent data from the backup to the Secondary data volumes.
V-5-52-803	Lost connection to host *host*; terminating command execution.	During command execution, the host *host* became unreachable or `vradmind` server on host *host* stopped running. Action: Retry running the command after host *host* becomes reachable or after restarting `vradmind` on host *host*. This could also be due to some problem with `vradmind`. In such a case, contact Veritas Customer Support.
V-5-52-2406	Cannot allocate space to grow volume to new_size blocks.	The system does not have enough free space to accommodate the requested growth in volume size. Action: Retry running the command when there is enough free space in VM configuration.

Table B-12 vradmin error messages *(continued)*

Unique Message Identifier (UMI)	Message	Message definition
V-5-4-2411	Volumes under RVG *rvg_name* are not prepared for the instant snapshot.	The command requires the data volumes to be prepared before using the instant snapshot operations. The error occurred when trying to create instant snapshots of all the data volumes under an RVG, because the volumes were not prepared. *Action:* For the first time, before creating instant full snapshot of the volumes under an RVG, prepare the volumes using the command: `vxsnap -g diskgroup prepare volume` For more information, refer to the `vxsnap`(1M) manual page.

Messages related to configuration errors

This section describes the configuration related errors that are displayed by the `vradmin -l printrvg` command and the `vradmin -l repstatus` command. The Config Errors section appears in the output of the commands only if there are configuration errors in an RDS. The *host* variable in the error messages is the host name on which the error has occurred.

Note: Sometimes, the Config Errors section in the `printrvg` command output may display multiple error messages separated by commas, for a specific host.

Table B-13 Messages related to configuration errors

Message	Message definition
host: Pri or Sec IP not available or vradmind not running	The Primary IP or the Secondary IP address is not available, or the `vradmind` daemon on the host is not running or is running on a different port. Action: If it is a network problem, correct it. If `vradmind` server is not running on *host*, start it. If `vradmind` is running and network connection is fine, make sure that `vradmind` is using the same port, as `vradmind` on other hosts in the RDS. Use `vrport` command to check or set the ports.
host: disk group missing.	Host *host* does not have any disk group with the same name as that specified in the *remote_dg* attribute, of the Primary RLINK pointing to this host. Action: Make sure the *`remote_dg`* attribute of the Primary RLINK contains the correct remote disk group name. Use *`vxprint -l rlink_name`* to see the RLINK settings. If the disk group needs to be imported on *host*, and it hasn't been done, then import the disk group on *host*.
host: RLINK missing.	Primary RVG has an RLINK to *host*, but *host* does not have corresponding rlink to this Primary RLINK. Action: This error can occur due to the following reasons: ■ One or more of the following attributes in the Primary RLINK is incorrect: *remote_host*, *remote_dg*, and *remote_rlink*. Make sure these attributes are correct. ■ The corresponding Secondary RLINK on *host* is missing. To fix this, you can dissociate the Primary RLINK, delete it from the configuration, and then use `vradmin addsec` command to add *host* to the RDS.
host: RLINK dissociated.	Host *host* does have an RLINK corresponding to the Primary RLINK. However, it is not associated with the Secondary RVG. Action: Reassociate the RLINK to the Secondary RVG on *host*.

Table B-13 Messages related to configuration errors *(continued)*

Message	Message definition
host: disk-group mismatch.	The *remote_dg* attribute of either the Primary RLINK or Secondary RLINK is incorrect. Action: Make sure the *remote_dg* attribute of the Primary RLINK contains the Secondary disk group name and the *remote_dg* attribute of the Secondary RLINK contains the Primary disk group name.
host: RLINK mismatch.	The *remote_rlink* attribute of either the Primary RLINK or Secondary RLINK is incorrect. Action: Make sure the *remote_rlink* attribute of the Primary RLINK contains the Secondary RLINK name and the *remote_rlink* attribute of the Secondary RLINK contains the Primary RLINK name.
host: host mismatch.	The *local_host* and/or *remote_host* attribute of either the Primary RLINK or Secondary RLINK is incorrect. Action: Make sure the value of the *local_host* attribute of the Primary RLINK is the same as that of the *remote_host* attribute value of the Secondary RLINK. Also make sure the value of the *remote_host* attribute of the Primary RLINK is the same as that of the *local_host* attribute of the Secondary RLINK.
host: Primary-Primary configuration.	The two Primary RVG RLINKs are pointing to each other. This situation will arise after the original Primary comes up after a Primary failover. Action: Use `vradmin fbsync` or `vradmin makesec` command to convert the original Primary to a Secondary.
host: multiple Primary error.	The same Secondary RVG has more than one Primary RVGs. Action: Check both Primary RVGs to verify which one is the intended Primary then do the following: ■ Dissociate the Secondary RLINK pointing to the unwanted Primary RVG. ■ Dissociate the Primary RLINK from the unwanted Primary RVG. ■ Remove the disassociated RLINK(s) from the configuration.

Table B-13 Messages related to configuration errors *(continued)*

Message	Message definition
host: two or more nodes on same host.	Two or more RVGs in the same RDS are located on the same host, *host*. This configuration is not supported. Action: Keep only one RVG on that host and remove the rest.
host: platform mismatch.	Replication is allowed only between hosts that are on the same platform. This is an invalid configuration.
No Primary RVG.	Ignore this error if it displays only for a few seconds and disappears. If the error persists, then the problem may be due to the Secondary not being able to determine its Primary because it has some configuration error, the Primary is not reachable, or it does not have any RLINK. Action: If the Secondary has an RLINK to a Primary, check to see if the Primary RVG and the corresponding Primary RLINK exist. If yes, make sure that `vradmind` is running on Primary and the network between Primary and Secondary is good. If everything is good, then run `vradmin -l printrvg` command on the Primary to see what type of configuration error is displayed. Correct the configuration error(s). If Primary RVG does not exist or the Secondary RVG does not have any RLINKs, then just remove the Secondary RVG.
host: Primary and Secondary have same disk-group ID.	This condition happens in the cases when the split mirrored plexes of the Primary volumes are exported without using the Disk Group split option and then imported on the Secondary with force option. Action: Contact Veritas Customer Support.
host: unknown	The configuration status is currently unknown. Action:
host: stale information	The configuration status may be stale. Action:
host: no data volume	This error can occur if one of the RVGs in the RDS does not have a data volume associated with it. Action: Create the appropriate data volume and associate it with the RVG.

Table B-13 Messages related to configuration errors *(continued)*

Message	Message definition
host: network-protocol mismatch	The `protocol` attribute of the Primary RLINK is different from that of the Secondary RLINK. Action: Make sure the value of the `protocol` attribute for the Primary and Secondary RLINK is the same.
host: VVR-heartbeat-port mismatch	The `local_port` attribute setting for the Primary RLINK is different from that of the Secondary RLINK. Action: Make sure the value of the `local_port` attribute for the Primary and Secondary RLINK is the same.
host: unsupported VVR version in cross-version replication	The cross-version replication in VVR is only supported between two immediate major releases. Action: Upgrade the VVR version on the host *host* to the same VVR version as on the other hosts in the RDS.
host: no contact from Primary	This error can occur if the `vradmind` server on the Secondary host is unable to establish contact with the `vradmind` server on the Primary. This can be because the Primary RVG of this RDS cannot be found or the `vradmind` server is not running or is unreachable. Action: If the `vradmind` server is not running, start it.
host: vxconfigd disabled	The `vxconfigd` daemon is currently disabled on the host on which this error occurred. Action: Start the `vxconfigd` daemon.
host: volume-number mismatch	This error can occur if the number of volumes in the Primary and Secondary RVGs of the RDS are different. Action: Use `vradmin printvol` to find out which RVG has the extra data volumes and then either remove them from that RVG or associate corresponding data volumes for these extra data volumes to the other RVGs in the RDS.

Table B-13 Messages related to configuration errors *(continued)*

Message	Message definition
host: volume-size mismatch	This error can occur if the sizes of some or all of the data volumes in the Primary and Secondary RVGs of the RDS do not match. Action: Use `vradmin printvol` to find out which data volumes have mismatching sizes and then use the `vradmin resizevol` command to correct the size mismatch error.
host: volume-name mismatch	This error can occur if some or all of the volumes in the Primary and Secondary RVGs of the RDS are not mapped correctly. Action: Use `vradmin printvol` to find out which data volumes are not mapped correctly and correct the mapping error. See "Mapping the name of a Secondary data volume to a differently named Primary data volume" on page 391.
host: Primary SRL missing	This error can occur if the Primary SRL was disassociated from the Primary RVG or is missing. Action: If the Primary SRL is missing, create it and associate it to the Primary RVG. If it was disassociated, re-associate it back to the Primary RVG.
host: Secondary SRL missing	This error can occur if the Secondary SRL was disassociated from the Secondary RVG or is missing. Action: If the Secondary SRL is missing, create it and associate it to the Secondary RVG. If it was disassociated, re-associate it back to the Secondary RVG.

Messages related to the vrstat command

This section lists and describes some of the error messages that may be displayed when running the `vrstat` command. It also suggests the appropriate action that needs to be taken if a problem occurs.

Table B-14 Messages related to the vrstat command

Unique Message Identifier (UMI)	Message	Message definition
V-5-84-162	Terminating the collection of statistics.	This error may occur while trying to run the `vrstat` command on the secondary, if the `vradmind` server on the Primary was terminated while the `vrstat` command was in progress. This could be due to an administrative action or due to some problem in `vradmind`. *Action:* If you had stopped the `vradmind` server manually, then restart it and run the `vrstat` command again. If the `vradmind` server stopped due to some problem, contact Veritas Customer Support.

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. The In-Band Control (IBC) Messaging feature with the Snapshot feature of Veritas Volume Replicator (VVR) and optional FastResync (FR) feature of Volume Manager (VxVM) enable you to perform off-host processing. 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 228.

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.

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 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:

- "Registering an application name" on page 369.
- "Displaying the registered application name for an RVG" on page 369.
- "Receiving an IBC message" on page 369.
- "Sending an IBC message" on page 370.
- "Unfreezing the Secondary RVG" on page 371.
- "Unregistering an application name" on page 371.
- "Receiving and processing an IBC message using a single command" on page 371.
- "Sending and processing an IBC message using a single command" on page 372.

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 ten minutes before being discarded. 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 `vx ibc` 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 -F vxfs /dev/vx/rdsk/hrdg/trial-hr_dv01
# fsck -F 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

IOCTL commands

This section describes the IOCLTL 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 <vxvm/voldefs.h>
    # include <vxvm/volioctl.h>
    # include <vxvm/volibc.h>
```

```
int ioctl(int fd, int cmd, void *arg);
```

Include the path `/opt/VRTSvxvm/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.

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/rc2.d/S94vxnm-vxnetd
```

On failure, possible values returned in `errno` are:

`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 `RV_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_MESSAGE_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.

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.
`EBUSY`	IBC deliver or unfreeze pending.

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.

Veritas Volume Replicator object states

This appendix includes the following topics:

- Veritas Volume Replicator Kernel State
- Veritas Volume Replicator utility states

Veritas 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

This section lists the RVG KSTATEs and their descriptions.

- 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.
- RECOVER— You cannot do I/O to the volumes that belong to the RVG. This state is triggered after a diskgroup import or if the RVG object was not recovered properly after a reboot or a crash. Issue the `vxrvg recover` command.

RLINK KSTATEs

This section lists the RLINK KSTATEs and their descriptions.

- 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 diskgroup import or if the RLINK object was not recovered properly after a reboot or a crash. Issue the `vxrlink recover` command.

Veritas 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 388.
- 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" on page 77.

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 "The srlprot attribute" on page 57.

To change the state of an RLINK from stale to active, use one of the methods described in the following sections:

- Using the automatic synchronization feature
- Example—Synchronizing the Secondary using block-level backup

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. This command must be used with great care because it enables data volumes on the Secondary to be written. 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 checkpoint if a Secondary checkpoint is not available, but the Primary 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 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 Veritas 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. Veritas Volume Replicator commands that do two-phase transactions report an error message and have a nonzero exit code if they fail.

VVR task reference

This appendix includes the following topics:

- Mapping the name of a Secondary data volume to a differently named Primary data volume
- Mapping disk groups
- Decreasing the size of the SRL on the Primary

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
```

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 offlining resources, see the *Veritas 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 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 onlining resources, see the *Veritas 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.

Migrating VVR from Internet Protocol version4 to Internet Protocol version6

This appendix includes the following topics:

- Overview
- Migrating to IPv6 when the GCO and VVR agents are not installed
- Migrating to IPv6 when the GCO and VVR agents are installed

Overview

Replication between the Primary and the Secondary sites within VVR currently takes place through the IPv4 network. VVR can be migrated from IPv4 to the IPv6 network.

The migration can take place in the following two scenarios:

- A set up in which the GCO and VVR agents are not installed.
- A set up in which the GCO and VVR agents are installed

The following prerequisites must be met before initiating the migration:

- IPv6 must be configured on all the systems.
- The virtual IP must also be configured to use IPv6

- The hostnames of the systems configured with IPv6 must be the same as the ones in the IPv4 network. The hostnames of virtual IPs in the IPv6 network must be the same as in the IPv4 network.
- The IP and NIC of the VCS failover service group which is created for the HA of the virtual IP must be configured to use IPv6.

The migration process involves the following features:

- Creating new resources does not affect the VVR replication in IPv4.
- To migrate the VVR from IPv4 to IPv6, the `vradmin changeip` command is used:

  ```
  vradmin -g dg changeip RVG newpri=Virtual IPv6_1 Addr newsec=Virtual IPv6_2
  ```

- After the migration from IPv4 to IPv6, the IPv4 network must be removed.

Migrating to IPv6 when the GCO and VVR agents are not installed

Before migrating VVR from IPv4 to IPv6, let us first consider the existing setup:

- 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 1 IP resource and 1 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:

  ```
  swlx25:~ # 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
  swlx25:~ #
  ```

- On both the systems the state of the failover service group is similar to the following:

```
swlx25:~ # hastatus -summ | grep -i VVRGRP
B  VVRGRP          swlx25                Y          N          ONLINE
B  VVRGRP          swlx27                Y          N          OFFLINE
swlx25:~ #
```

- 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
        //          }
        //      }
```

- On creation of service group VVRGRP, the output of the `ifconfig` file is displayed as follows:

```
swlx25:~ # ifconfig
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.
          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)

swlx25:~ #
```

To migrate VVR from IPv4 network to the IPv6 network, follow the steps given below:

1. In the existing failover service group created for IPv4 virtual IP, create an IP resource and/or NIC resource for the virtual IPv6 address.

   - Enable write operations on the VCS configuration using the`haconf -makerw` command
   - Add the NIC resource for the IPv6 address and configure the related attributes:

     ```
     hares -add  nicres1  NIC  VVRGRP
     hares -modify nicres1 Device  eth2
     hares -modify nicres1 Enabled 1
           hares –probe nicres1 –sys node1
           hares –probe nicres1 –sys node2
     ```

   - Add the IP resource for the IPv6 address and configure the necessary attributes. For example:

     ```
     hares -add  ipres1  IP  VVRGRP
     hares -modify ipres1 Device  eth2
     hares -modify ipres1 Address fd4b:454e:205a:111:211:43ff:feaa:af71
     hares -modify ipres1 Enabled 1
     hares -modify ipres1 PrefixLen 64
     hares -probe ipres1 -sys node1
     hares -probe ipres1 -sys node2
     hares -online ipres1 -sys node1
     ```

     In the above example, the new IP resource ipres1 is configured for the virtual IPv6 address.

     Note: If the IPv6 address is configured on the same NIC which is used for IPv4 then a new NIC resource need not be created. Instead, use the existing NIC resource and link the new IP resource to it. If the IPv6 address is configured on a separate interface then create a new NIC resource (nicres1) for the IPv6 interface.

2. Link the IP resource to the NIC resource using the`hares -link ipres1 nicres1` command

3. Save the changes made to the configuration using the`haconf -dump -makero` command

4 Repeat steps x to y on both the Primary and Secondary sites.

5 Verify if the virtual IP on both the sites is different.

Note: Do not stop VCS; do not modify the existing service group in the `main.cf` file. These changes will affect the replication between the sites as the service group will go offline.

---

6 After the configuration is saved, the existing service group (VVRGRP) settings will be as follows:

- The state of resources after configuring the Virtual IPv6 address.

```
swlx25:~ # hares -state | grep -i res
#Resource     Attribute        System     Value
ipres         State            swlx25     ONLINE
ipres         State            swlx27     OFFLINE
ipres1        State            swlx25     ONLINE
ipres1        State            swlx27     OFFLINE
nicres        State            swlx25     ONLINE
nicres        State            swlx27     ONLINE
nicres1       State            swlx25     ONLINE
nicres1       State            swlx27     ONLINE
swlx25:~ #
```

- State of the failover service group (VVRGRP)

```
swlx25:~ # hastatus -summ | grep -i vvr
B  VVRGRP        swlx25          Y          N          ONLINE
B  VVRGRP        swlx27          Y          N          OFFLINE
swlx25:~ #
```

- 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 ipres1 (
           Device = eth2
           Address = "fd4b:454e:205a:111:211:43ff:feaa:af71"
           PrefixLen = 64
           )

   NIC nicres (
           Enabled = 1
           Device = eth0
           )

   NIC nicres1 (
Enabled = 1
           Device = eth2
           )

   ipres requires nicres
   ipres1 requires nicres1


   // resource dependency tree
   //
   //      group VVRGRP
   //      {
   //      IP ipres
   //          {
   //          NIC nicres
   //          }
   //      IP ipres1
   //          {
   //          NIC nicres1
   //          }
   //      }
```

- Output of the `ifconfig` file after modifying the service group (VVRGRP)

```
swlx25:~ # ifconfig
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.
          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:Globa
        inet6 addr: fd4b:454e:205a:111:211:43ff:fe37:da69/64 Scope:Globa
        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)

swlx25:~ #

Virtual IPv4 Address: eth0: 10.209.87.186
Virtual IPv6 Address: eth2: fd4b:454e:205a:111:211:43ff:feaa:af71
```

Note: Replication still continues to use the virtual IPv4 address. Virtual IPv6 address is successfully configured without affecting the existing VVR replication.

Both the virtual IPv6 and IPv4 addresses are online. As a result, migration of the Primary and Secondary VVR sites from the IPv4 network to the IPv6 network can now occur. This activity does not cause any disruption to the replication between the sites.

7 Migrate the VVR from the IPv4 to IPv6 network using the `vradmin changeip` command:

```
vradmin -g dg changeip RVG newpri=Virtual IPv6_1 \
Addr newsec=Virtual IPv6_2
```

For example:

```
swlx25:~ # vradmin -g dg1 repstatus VVG
Replicated Data Set: VVG
Primary:
  Host name:                    10.209.87.186
  RVG name:                     VVG
  DG name:                      dg1
  RVG state:                    enabled for I/O
  Data volumes:                 1
  VSets:                        0
  SRL name:                     srlvol
  SRL size:                     800.00 M
  Total secondaries:            1

Secondary:
  Host name:                    10.209.87.171
  RVG name:                     VVG
  DG name:                      dg1
  Data status:                  consistent, up-to-date
  Replication status:           replicating (connected)
  Current mode:                 synchronous
  Logging to:                   SRL
  Timestamp Information:        behind by 0h 0m 0s

swlx25:~ #


swlx25:~ # vradmin -g dg1 changeip VVG 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 V
VxVM VVR vxrlink INFO V-5-1-6183 vol1:   len=10485760     primary_datavol=v


swlx25:~ # vradmin -g dg1 repstatus VVG
Replicated Data Set: VVG
Primary:
  Host name:                   fd4b:454e:205a:111:211:43ff:feaa:af70
  RVG name:                    VVG
  DG name:                     dg1
  RVG state:                   enabled for I/O
  Data volumes:                1
  VSets:                       0
  SRL name:                    srlvol
  SRL size:                    800.00 M
  Total secondaries:           1

Secondary:
  Host name:                   fd4b:454e:205a:111:211:43ff:feaa:af71
  RVG name:                    VVG
  DG name:                     dg1
  Data status:                 consistent, up-to-date
  Replication status:          replicating (connected)
  Current mode:                synchronous
  Logging to:                  SRL
  Timestamp Information:       behind by 0h 0m 0s
```

8. Verify that the RLINKs now use the IPv6 network for replication.
9. The IPv4 network is still present.You must remove the IPv4 network to complete the migration process.

   To remove the IPv4 network, See "How to remove the IPv4 network" on page 407.

How to remove the IPv4 network

1. Delete the IP and NIC resources that belong to the IPv4 network.

   ```
   haconf -makerw
   hares -offline ipres -sys swlx25
   hares -unlink ipres nicres



   hares -delete ipres
   hares -delete nicres
   haconf -dump -makero
   ```

2. Use the ifdown script to turn off the interface: `ifdown eth0`

3 Delete the configuration file of the IPv4 interface:

```
swlx25:/etc/sysconfig/network # ifconfig
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:1732 errors:0 dropped:0 overruns:0 frame:0
          TX packets:135 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:161354 (157.5 Kb)  TX bytes:24781 (24.2 Kb)
          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:80275 errors:0 dropped:0 overruns:0 frame:0
          TX packets:43501 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:7808285 (7.4 Mb)  TX bytes:6331346 (6.0 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:170073 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3689 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:18431248 (17.5 Mb)  TX bytes:1414437 (1.3 Mb)

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)


swlx25:/etc/sysconfig/network # ifdown eth0
    eth0  device: Broadcom Corporation NetXtreme BCM5701
    Gigabit Ethernet (rev 15)

swlx25:/etc/sysconfig/network # ifconfig
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:80387 errors:0 dropped:0 overruns:0 frame:0
          TX packets:43559 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:7818777 (7.4 Mb)  TX bytes:6338386 (6.0 Mb)

eth2      Link encap:Ethernet  HWaddr 00:11:43:37:DA:69
          inet6 addr: fd4b:454e:205a:111:211:43ff:fe37:da69/64 Scope:Gl
          inet6 addr: fe80::211:43ff:fe37:da69/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:170382 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3726 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:18464082 (17.6 Mb)  TX bytes:1421995 (1.3 Mb)

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)

swlx25:/etc/sysconfig/network # ls
config  dhcp6r  ifcfg-eth0  ifcfg-eth2  ifcfg.template  ifroute-lo
providers  routes.YaST2save
dhcp    dhcp6s  ifcfg-eth1  ifcfg-lo    if-down.d       if-up.d
routes     scripts
```

```
swlx25:/etc/sysconfig/network # rm ifcfg-eth0
```

4 Remove the IPv4 network cable

Note: Do not restart the network service. This will fault the failover service group created and can obstruct the replication.

VVR is now successfully migrated to the IPv6 network.

Migrating to IPv6 when the GCO and VVR agents are installed

Overview of the existing setup:

- The GCO and VVR setup uses the IPV4 addresses
- There are two sites; Primary and Secondary. Replication is running between these two sites using the IPV4 addresses.
- VCS VVR agents are used to control the VVR functionality.
- GCO 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
```

The following prerequisites must be met before initiating the migration:

- Modify the `/etc/resolv.conf` file to include both v4/v6 DNS servers and domain names.
  The hostnames of the systems using IPv6 must be the same as the ones in the IPv4 network. The hostnames of virtual IPs in the IPv6 network must be the same as in the IPv4 network.
- If DNS servers are not available, add the IPv6 address entries to the `/etc/hosts` file.
- The hostnames of virtual IPs for GCO and RVG logowner service groups on the IPv4 network and the IPv6 network must be the same.
- The setup must be configured in a dual stack mode. This can be achieved by assigning the IPv6 address to a separate link interface on both the sites or to the same interface.
- Use a separate network interface for enabling IPv6 connectivity.

To migrate VVR to the IPv6 network:

1. First migrate the GCO service group. An online migration of the GCO service group is not supported. Turn the service group offline and then modify the required parameters.

   In the following example, the CLusterService service group on both the primary and secondary sites is turned offline using the following command:

   ```
   # hagrp -offline -force <service group> -sys <node_name>

   # hagrp -offline -force ClusterService -sys <node_name>
   ```

   Note: Putting the service group offline does not impact the replication between the sites.

2. To migrate GCO to the IPv6 network, follow steps 3-6:
3. On the Primary site, enable write configuration on the VCS configuration using the `haconf -makerw` command

4 Modify the VCS GCO related attribute values to support the IPv6 environment.

For example, change the GCO ClusterService group related attributes as follows:

```
#haclus -modify ClusterAddress fd4b:454e:205a:111:213:72ff:fe5b:2f67 -cl
 #haclus -modify ClusterAddress fd4b:454e:205a:111:211:43ff:fede:1e11 -c
 #hahb -modify Icmp Arguments fd4b:454e:205a:111:211:43ff:fede:1e11 -clu
 #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
 #haconf -dump -makero
```

5 Modify the VCS GCO related attributes on the Secondary site.

6 Bring the GCO ClusterService group on both the sites online by using the following command:

```
# hagrp -online ClusterService -sys <node_name>
```

The VCS `main.cf` configuration file for the ClusterService will be 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
```

7 Modify the IP and NIC attributes of the service group which has RVG logowner agent resource for controlling the logowner. Perform the modifications on both the Primary and Secondary sites.

```
#haconf -makerw
 #hares -modify logowner_ip Device eth1
 #hares -modify logowner_ip Address fd4b:454e:205a:111:213:72ff:fe59:4a2
 #hares -modify logowner_ip Enabled 1
 #hares -modify nic Device eth1
 #hares -modify nic Enabled 1
 #hares -modify logowner_ip PrefixLen 64
 #haconf -dump -maker
```

Note: VVR replication between sites is not impacted by these modifications.

---

8 To migrate the VVR replication from the IPv4 network to the IPV6 network, execute the following command:

```
# vradmin -g shared_dg changeip rac1_rvg newpri=fd4b:454e:205a:111:213:7
         newsec=fd4b:454e:205a:111:213:72ff:fe58:3d8b
```

9 Verify if the replication RLINKs on both the sites now use the IPv6 addresses using the following command : `# vxprint -PVl`

The service group which monitors or controls the logowner will now be 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
                )
```

10 The entire GCO 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. Remove the IPv4 network to complete the migration process.

To remove the IPv4 network, See “How to remove the IPv4 network” on page 407.

Glossary

ACTIVE	On a Secondary RLINK, the ACTIVE state indicates that it is ready to receive updates from the Primary.
asynchronous	Asynchronous mode queues writes persistently and holds them at the Primary for later transmission.
automatic synchronization	A feature of VVR that synchronizes the Secondary while the application is running on the Primary.
can_sync	If the `inconsistent` and `can_sync` flags are set, there is enough information in the Secondary SRL to make it consistent again. In this case, the RLINK does not need to be fully resynchronized and is still a suitable candidate for takeover.
cant_sync	If the RLINK flag is `cant_sync`, the RLINK is inconsistent and the Secondary needs a full synchronization before it can take part in replication again.
checkpoint	A feature of VVR that allows replay of the SRL from an earlier point than the current position. A checkpoint delineates with starting and ending points the section of the SRL to be replayed later.
CLEAN	If the Primary RVG is in the CLEAN ACTIVE state, the RLINK must be attached.
copy-on-write	A technique for preserving the original data. Before data is modified by a write operation, the original copy of data is copied to a different location.
consistent	A term indicating that data is recoverable by the system or application using it; for example, a file system or database. In VVR, a Secondary that is consistent can be used for takeover.
data volume	A volume that is associated with an RVG and contains application data.
DCM (Data Change Map)	An object containing a bitmap that can be optionally associated with a data volume on the Primary RVG. The bits represent regions of data that are different between the Primary and the Secondary. The bitmap is used during synchronization and resynchronization.
disaster recovery	Disaster Recovery (DR) has a wide scope, which ranges from the storage of backup tapes off site to the setup and maintenance of a duplicate remote standby node. VVR aids disaster recovery by providing timely replication of data to a remote site.

distributed command	A command or task that can be performed on an RDS from any host in the RDS environment; a related task is performed in sequence on all hosts in the RDS, if appropriate.
EMPTY	If the Primary RVG state is EMPTY, an RLINK can be attached with no special options.
FAIL	A Secondary RLINK can enter the FAIL state when the Secondary data volume has an error or when an ACTIVE Secondary RLINK is paused with the `-w` option. A Primary RLINK enters the FAIL state when the corresponding Secondary RLINK enters the FAIL state.
failover	Failover is a term associated with the Veritas Cluster Server environment. See Primary takeover for a description in the VVR environment.
FastResync	A feature that is used to perform quick and efficient resynchronization of stale mirrors, and to increase the efficiency of the snapshot mechanism. See also Persistent FastResync, Non-Persistent FastResync.
heartbeat protocol	The heartbeat protocol is a continuous exchange of messages to ensure that the nodes in an RDS will detect any network outage or a node crash. The protocol allows the nodes to reestablish a connection later.
IBC	See In-Band Control Messaging.
IBC (in-band control messaging)	A facility that enables applications to inject control messages in the replication stream. The contents of the control message itself are application-defined and opaque to the replication process.
inconsistent	In VVR, a Secondary is inconsistent if it is not a viable candidate for takeover, because it is known that the application will not be able to recover.
latency protection	For RLINKs operating in asynchronous mode, which may fall behind, the latency protection attribute (`latencyprot`) of the RLINK is used to limit the maximum number of outstanding write requests. The maximum number of outstanding write requests cannot exceed the value set in `latency_high_mark`, and cannot increase until the number of outstanding writes falls to the `latency_low_mark`.
latencyprot	See latency protection.
logowner	The node on which VVR performs replication when replicating in a shared disk group environment. For synchronous RLINKs, VVR also performs writes on the logowner node.
metadata shipping	The process of exchanging information between the non-logowner nodes that issue writes and the logowner, and then writing locally on the non-logowner nodes, when replicating in asynchronous mode.

nmcom pool	The amount of buffer space available for updates coming in to the Secondary over the network.
Non-Persistent FastResync	A form of FastResync that cannot preserve its maps across reboots of the system because it stores its change map in memory. See FastResync.
object recovery	The process of making an object usable after a system crash.
Passthru	Typically, writes to a Primary RVG are written to the SRL first, and then to the RLINKs and data volumes. If there is no SRL or the SRL is detached, writes are no longer written to the SRL and the RVG is in PASSTHRU mode. No replication takes place.
Persistent FastResync	A form of FastResync that can preserve its maps across reboots of the system by storing its change map in a DCO volume on disk.
plex	A copy of a volume and its data in the form of an ordered collection of subdisks. Each plex is a copy of the volume with which it is associated. The terms mirror and plex can be used synonymously.
Primary checkpoint	A method for synchronizing a Secondary with the Primary without stopping the application. A command marks the start of the checkpoint. All Primary data volumes are backed-up and then the end of the checkpoint is marked. The data is restored on the Secondary and the Primary can then begin from the start of the checkpoint. The Secondary becomes consistent when the end of the checkpoint is reached.
Primary pause	An administrator at the Primary volume node may pause updates to any particular RLINK of an RVG. During a pause, the Primary continues to keep a history of volume updates, but active update of the RLINK ceases and the network connection between the Primary and Secondary is broken. A Primary pause is intended as a maintenance feature and allows certain configuration changes to be made, such as a change to the network connecting two nodes.
Primary migration	The Primary role of a volume can be migrated from a Primary node to a Secondary node, within certain restrictions. The process is manual and requires cooperative administrative action on the Primary and all Secondary nodes. During this process, update of the former Primary must be halted and cannot be resumed on the new Primary until the migration is complete. The Primary role can only be migrated to an up-to-date RLINK that is consistent and is not in an error state. Any out-of-date Secondaries must be fully resynchronized with the new Primary.
Primary node	The Primary node is where the application is running, and from which data is being replicated to the Secondary.

Primary node crash	If a Primary node crash occurs, the Primary SRL and all data volumes in the RVG must be recovered. Both are recovered using VVR specific recovery, rather than the usual Volume Manager volume recovery.
Primary node data volume failure	If there is an error accessing a Primary data volume, the data volume and the RVG are automatically detached, and the RVG state changes to FAIL. RLINKs are not affected. If the SRL volume was not empty at the time of the volume error, the updates continue to flow from the SRL to the Secondary RLINKs.
Primary node SRL overflow	Because the Primary SRL is finite, prolonged halts in update activity to any RLINK can exceed the SRL's ability to maintain all the necessary update history to bring an RLINK up-to-date. When this occurs, the RLINK is marked as STALE and requires manual recovery before replication can proceed.
Primary Replicated Volume Group	See RVG (Replicated Volume Group).
Primary SRL failure	See Passthru.
Primary takeover	The Primary role can be arbitrarily taken over by a Secondary node. This process is similar to a Primary role migration but presumes that the old Primary is inoperable and unable to participate in the migration process. Primary 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 takeover because they leave the RLINK in an inconsistent state. All such scenarios require the hardware failure of a data volume or SRL.
RDS (replicated data set)	The group of the RVG on a Primary and the corresponding RVGs on one or more Secondary hosts.
readback	The process of retrieving a write request from the SRL in order to send it across the RLINK.
readback pool	The amount of buffer space available for readbacks.
RLINK	An RLINK represents the communication link between the corresponding RVGs on the Primary and Secondary nodes.
RVG (replicated volume group)	A component of VVR that is made up of a set of data volumes, one or more RLINKs, and an SRL. VVR replicates from the Primary RVG, on the node where the application is running, to one or more Secondary RVGs.
RVIO pool	The amount of buffer space allocated within the operating system to handle incoming writes.
Secondary checkpoint	See checkpoint.
Secondary data volume failure	Secondary data volume failure causes the RLINK to be put in the fail state. The Primary stops sending requests to the RLINK, but logging continues on the Primary

	node. When the failure has been corrected, it can be restored from a backup made earlier using a Secondary checkpoint.
Secondary node	The node to which VVR is replicating data from the Primary.
Secondary pause	An administrator at the Secondary node can `pause` updates to the RLINK. Unlike a Primary `pause`, the Primary-Secondary network connection is maintained. This enables the Secondary to notify the Primary when it wants updates of the RLINK to continue. A Secondary `pause` can be effected even when the Primary and Secondary have lost contact.
Secondary Replicated Volume Group	See RVG.
snapshot	A point-in-time image of a volume or file system that can be used as a backup.
snapshot volume	An exact copy of a volume, at a specific point in time. The snapshot volume is created by breaking a snapshot plex from the original volume and placing it in a new volume, which is then used for online backup purposes.
SRL (Storage Replicator Log)	A circular buffer of writes for an RVG. Writes stored in the SRL are waiting to be shipped to the Secondary from the Primary, or waiting to be written to the data volumes in the RVG.
SRL overflow protection	A feature of VVR that ensures that a Secondary RVG does not require a full resynchronization after a Primary node SRL overflow.
STALE	The RLINK state that indicates that the RLINK has either not been attached yet or it has overflowed.
synchronization	The process of making the data on the Secondary identical to the data on the Primary.
synchronous	In synchronous mode, the Secondary is kept up-to-date with the Primary by waiting for each write request to be acknowledged by the Secondary before the application sees the successful completion of the write on the Primary.
unrecoverable errors	Some errors, in particular hard errors such as media failures, require manual intervention to repair. The chances of such failures can be minimized by using standard VxVM setups to maintain mirrored copies of each data volume and the SRL.
update	Data from the Primary corresponding to an application write sent to the Secondary.
Volume Replicator Objects	The objects used for replication such as RVG, SRL, RLINK, and DCM.
write-order fidelity	A feature of VVR that applies writes on the Secondary in the same order as they are received on the Primary. This ensures that the data on the Secondary is consistent with the data on the Primary.

write shipping	The process of sending the writes issued on nodes other than the logowner over the cluster network to the logowner.

Index

A

acting Secondary 257
activating a bunker
    using command line 279
adding a data volume
    using command line 137
adding a Secondary
    best practices 70
    using command line 68
application-defined control messages 379
applications
    on Secondary 239
associating a data volume
    using command line 137
associating DCM to a data volume
    using command line 146
associating SRL to a Primary RVG
    using command line 66
asynchronous mode
    defined 417
asynchronous replication
    data flow 28
    explained 25, 54
    setting up 74
    using bunker replication 267
autodcm
    srlprot setting 57, 77
autofb option
    takeover command 251
automated failover
    using command line 285
automatic fast failback
    using command line 251
Automatic Synchronization
    synchronizing Secondary example 96
automating firedrill 207
autosync flag, definition 115

B

backup
    block-level tape 81, 87

backup *(continued)*
    primary checkpoint 307
    using snapshot and vradmin ibc
        example 236
    using snapshot and vxibc
        example 375
balancing RPO with RTO
    using bunker replication 270
bandwidth limit
    controlling
        used for replication. *See* using command line
        used to synchronize volumes. *See* using command line
Bandwidth Throttling 61, 79
    See also bandwidth limit
    about 61, 79
    disabling
        using command line 79
bandwidth used by VVR
    determining 124
block-level tape backup 81, 87
breaking off mirrors 158
bunker commands
    list 287
bunker configuration
    using command line 272
bunker replay
    using command line 281
bunker replication
    in a VCS global cluster 286
    settings
        using command line 271
bunker Secondary
    activating
        using command line 279
    administering
        using command line 279
    deactivating
        using command line 279
    using command line 276

bunker setup
    using command line 284
bunker site
    see Primary SRL 268
bunker SRL
    using synchronous mode 268

C

cache object 156, 260
can_sync 389
cant_sync 389
changing IP addresses 165
changing network ports for replication 172
changing replication settings 163
changing VVR tunables 216
checkdelete
    vxrlink keyword 180
    vxrvg keyword 180
checkend
    vxrvg keyword 317
checkpoints
    backing up Secondary with 213
    creating 179
    data volume error
        cleanup using 307
        recovery using 306
    deleting 180
    displaying list 118
    ending 179
    restoring Secondary
        from Primary checkpoint 215
        from Secondary checkpoint 214
    Secondary pause 214
    understanding 40
    viewing
        using command line 180
checkstart
    vxrvg keyword 317
clusters
    cluster-shareable disk groups 32
commands 311
    See also individual command names
    reference 311
comparing
    snapshot methods 51
component volumes of a volume set, associating to RDSs 143
configuration
    sample 271
configuration *(continued)*
    using command line 276
configuration errors
    recovery from 291
consistent data
    concept 22
consistent RLINK 216
converting original Primary to Secondary
    using command line 241
cplist
    vxrlink keyword 118
    vxrvg keyword 118
creating
    cache object 189
    instant plex breakoff snapshot 192
    instant snapshot 181
    snapshot plexes 192
    snapshot volumes 183
    snapshots
        overview 181
    space-optimized snapshot 187
creating a Primary RVG
    using command line 66
creating Replicated Data Set
    using command line 64
creating RVGs
    using vradmin vs. vxmake 65
current data
    concept 22

D

DAS Direct Attached Storage 268
data
    consistent vs. up-to-date 22
Data Change Map
    see DCM 21
data differences, see data verification 207
data flow
    VVR asynchronous mode 28
    VVR synchronous mode 29
data loss
    and takeover 250
data on Secondary
    using for off-host processing 224
data verification
    concept 207
    performing 207
data volume errors
    missing 295

data volume errors *(continued)*
  name mismatch 297
  resize 296, 298
data volumes
  administering
    using command line 137
  associating
    using command line 137
  dissociating or removing
    using command line 152
  mapping Secondary name 391
  resizing
    using command line 149
  verifying data
    about 207
    before adding volumes to RDS 140
    using offline verification 210
    using online verification 208
DCM
  and fast failback 251
  associating to a data volume 146
  concept 21
  required for takeover 250
dcm
  srlprot setting
    about 57
    changing 77
DCM playback
  after SRL overflow 155
  during failback 251, 257
DCM size 147
dcm_logging flag, definition 156
deactivating a bunker
  using command line 279
Decision Support
  example 234, 372
deport bunker
  using command line 275
detach bunker
  using command line 279
difference-based synchronization
  about 91
  failing back using 262
disabling Bandwidth Throttling 79
disaster recovery
  failed Primary 247
  using a bunker 269, 279
  verifying readiness 45, 206
Disk Group Split and Join
  synchronizing Secondary using 89
disk groups
  displaying disk group ID 71
  shared
    replicating in 32
displaying
  checkpoints, list 118
  data volumes in RDS 105
  RVG and RDS information 103
dissociating data volumes
  using command line 152
DR readiness
  verifying 206
DSS. *See* Decision Support
Dual-node / Dual-stack 23

E

error messages 341
errors
  configuration 291
  data volume size mismatch 296, 298
  during an initial RLINK attach 292
  during modification of an RVG 295
  missing data volume 295
  name mismatch
    data volume 297–298
  vradmind daemon 344
event notification
  using command line 134
event types 134
examples
  using command line
    creating an RDS 95
    migrating Primary 243
    off-host processing 234, 372
    setting up replication 96
    synchronizing the Secondary 88
    taking over from failed Primary 252–253

F

fail
  latencyprot setting 60
  mode of replication 55
  srlprot setting 77
FAIL state 388, 390
failback logging
  about 251

failback logging *(continued)*
  example 258
failed Primary
  example takeover 252–253
failing back
  using command line
    about 255, 283
    fast failback versus difference-based synchronization 256
    using difference-based synchronization 262
    using fast failback or failback logging 256
fast failback feature
  about 251
  enabling with vradmin takeover 251
  versus difference-based synchronization 256
features of VVR 17
firedrill
  performing 207
firewalls
  using VVR in 70
  VVR ports in 173
flag settings
  RLINK 115
  RVG 105
flags and definitions
  autosync 115
  can_sync 389
  cant_sync 389
  Data Change Map 156
  inconsistent 389
  resync_active 156
  resync_paused 116, 156
  resync_started 116
  SRL Overflow Protection With DCM 156
freezing replication
  about 224
  before taking snapshot 184
  off-host processing 226, 229
  using ibc messaging 368
full synchronization
  about 85
  example 86

G
global mapping 391
GUIs, VVR
  about 63

H
heartbeats
  ports used for 174
high mark
  latency
    setting. *See* using command line
host names
  changing 165

I
IBC Messaging
  API 378
  explanation 225
  function 378
  off-host processing
    using for 225
  overview 224, 368
  vradmin ibc
    using for off-host processing 229
  vxibc
    individual commands 369
In-Band Control Messaging
  see IBC Messaging 225
inactive RLINKs 387
inconsistency
  inconsistent flag 389
  inconsistent RLINKs 388
  RLINK during restores 216
initial VVR setup
  using command line 94
interfaces of VVR
  about 63
IP addresses
  changing 165
IPv4-only node 23
IPv6-enabled node 23
IPv6-only node 23

L
latency low and high mark
  setting
    using command line 75
latency protection
  about 59
  disabling
    using command line 76
  enabling
    using command line 76

latency protection *(continued)*
  using command line 278
local host
  definition 25
local RVG
  represents RDS 25
log overflow
  STALE 387
low mark
  latency
    setting. *See* using command line

M
make Secondary
  using command line 262
master node
  about 32
memory statistics
  displaying 319
messages
  debugging 334
  kernel 321
  utility error 334
migrate
  vradmin keyword 241
migrating Primary
  example 243
  overview 240
  using command line 241
mirrors
  breaking off before synchronization 158
missing data volume error 295
modes of replication
  setting 74
most up-to-date Secondary
  identifying 132
multiple Secondaries in RDS
  migrating Primary with 242

N
network bandwidth used by VVR
  determining 124
network performance data 120, 123–124, 129
network ports
  changing 172
network requirements for bunker replication
  list 272
network transport protocol
  attribute
    STORAGE 78
    TCP 77
    UDP 77
  planning 70
  setting 77
notify
  events 134

O
object states
  VVR 385
off
  latencyprot setting 60
  mode of replication 55, 74
  srlprot setting 77
off-host processing
  backup example 375
  Decision Support example 234, 372
  examples 234, 372
  IBC Messaging
    using 225
  tasks to perform 228
  trial failover example 376
offline data verification
  concept 207
  using command line 210
onfreeze script
  about 229
  creating 231
onfreeze task
  off-host processing 229
online backup 215
online data validation 208
original Primary
  recovering using command line 281
  restoring using command line 281
overflow protection
  SRL 77
override
  latencyprot setting 60
  mode of replication 55
  srlprot setting 57, 77
  synchronous 277

P

packet size
    setting 78
pause
    vxrlink keyword 388
pause/resume
    temporary states 390
pauserep
    vradmin keyword 163
pausing replication
    using command line 163
Pausing state 390
performing
    offline data validation 207
    online data validation 208
performing firedrill 207
ports used by VVR
    changing 172
Primary checkpoint
    cleanup using 307
    deleting 180
    restoring from 215
    restoring Secondary from 388
Primary fails
    disaster recovery 279
Primary host
    concept 22
    failure recovery 300
Primary role takeover
    example 252–253
    overview 247
Primary RVG
    creating
        using command line 66
    removing
        using command line 178
Primary SRL
    error
        cleanup after 303
    header error 305
    overflow recovery 304
primary_datavol field 391
printrvg, vradmin keyword 103
printvol, vradmin keyword 105
protocol
    network 70

R

RAID-5 volumes and VVR 66
RDS
    administering data volumes
        using command line 137
    associating a volume set 143
    associating volumes
        using command line 137
    concept 21
    creating
        using command line 66
    displaying data volumes in RDS
        using command line 105
    displaying information
        using command line 103
    displaying replication status
        using command line 107
    removing
        using command line 178
    removing or dissociating data volumes
        using command line 152
    removing Primary from
        using command line 178
    resizing data volumes
        using command line 149
    resizing SRL
        using command line 160
RECOVER state 390
recovering from failed Primary 247
recovering RLINK 318
Recovery Point Objective (RPO) 250
Recovery Time Objective (RTO) 250
region size 147
reinitializing bunker
    using command line 278
removing a bunker
    using command line 286
removing data volume
    using command line 152
removing Primary
    using command line 178
removing Secondary
    using command line 177
replacement for Primary
    finding most suitable 249
Replicated Data Set
    see RDS 21
replication
    freezing 184, 226
    pausing 163
    resuming 163

replication *(continued)*
  resuming after migration 244, 247
  shared disk group 32
  starting 80
  status, displaying 107
  stopping to Secondary 164
  terms defined 22
  unfreezing 187, 371
replication settings
  bandwidth limit 61
  defaults 73
  latency protection 75
  overview 163
  packet size 78
  replication mode 54
  SRL overflow protection 77
repstatus, vradmin keyword 107
resizevol
  vradmin keyword 149
resizing a file system 150
resizing data volumes
  about 149
  using command line 151
resizing SRL
  using command line 160
restore
  restore
    vxrlink command 215–216
  restore command 390
  RESTORING state 390
  Secondary RLINK 215
  vxrlink keyword 318, 388
restoring bunker
  using command line 283
restoring original Primary
  using command line 282
resume
  resume command 390
  RESUMING state 390
  vxrlink keyword 318
resumerep
  vradmin keyword 163
resuming replication
  after migration 244, 247
  using command line 163
resync
  resync_active flag
    definition 156

resync *(continued)*
  resync_paused flag
    definition 116, 156
  resync_paused flag, definition 116, 156
resync_started flag, definition 116
resyncfromreplica
  recovering failed Secondary data volumes 204
  recovering from logical corruption of data 203
  using snapback 203
RLINK
  concept 20
  configuring 73
  creating 319
  displaying status 114
  dissociating from RVG 318
  flag settings 115
  inactive 387
  STALE 305
  states 386
  status command 98
RPO
  balancing RPO
    using bunker replication 270
RTO
  Recovery Time Objective
    using bunker replication 270
RVG
  associating to clear PASSTHRU 303
  concept 19
  displaying status 104
  flag settings 105
  set up automated failover 285
  start 316

S

sample configuration
  using command line 271
Secondary 80
  See also synchronizing Secondary
  acting 257
  adding 68
  backing up 213
  identifying most up-to-date 132
  removing 177
  removing from RDS 177
  transferring applications to 239
  using data 224
Secondary checkpoint
  backing up the Secondary 213

Secondary checkpoint *(continued)*
- deleting 180
- recovery using 306

Secondary host
- concept 22

Secondary SRL
- header error 308
- volume error 307

secondary_log_err flag 308
secondary_paused flag 307
set
- vradmin keyword 73

setting bandwidth limit
- using command line 79

setting network transport protocol 77
setting packet size 78
shared disk group
- replicating in
  - concepts 32

size of DCM 147
slave nodes
- about 32

SmartMove for VVR 93
- Turn off 93

snapback
- vxrvg keyword 202, 317

snapback operation
- about 202

snapprint
- vxrvg keyword 317

snaprefresh
- vxrvg keyword 317

snaprestore
- vxrvg keyword 317

snapshot
- vxrvg keyword 318

snapshot methods
- overview
  - instant full 47
  - instant plex breakoff 48
  - instant space-optimized 47

snapshot plexes
- creating 200

snapshots
- before DCM replay 257
- before takeover 250
- comparing methods 51
- overview 181
- traditional method 46

snapshots *(continued)*
- understanding 45
- using with fast failback 260
- using with resynchronization 156

snapshots of RVG
- creating
  - cache object 189
  - instant plex breakoff 192
  - instant snapshot 181
  - snapshot plexes 192
  - snapshot volumes 183
  - space-optimized 187
- destroying 199
- displaying information 198
- online data validation 208
- reattaching 202
- refreshing 194
- restoring 196
- resuming replication 187, 191, 194
- taking 201
- unfreezing replication 187

SRL
- associating to a Primary RVG
  - using command line 66
- concept 20
- resizing
  - using command line 160

SRL header error 305
SRL on bunker
- disaster recovery 269

SRL overflow protection
- disable 58
- explanation 77
- modes of 57, 77
- preventing 305
- with DCM
  - flags and definition 156

SRL playback 257
srlprot
- srlprot=dcm
  - override. *See* autodcm

STALE RLINK 387
starting replication 80, 278
startrep
- vradmin keyword 83

statistics
- displaying for VVR 119

stats
- vxrlink keyword 129, 319

stats *(continued)*
    vxrvg keyword 317
status
    vxrlink keyword 98, 319
stopping replication to Secondary
    using command line 164
stoprep
    vradmin keyword 164
synchronizing Secondary
    automatic synchronization
        example 96
    difference-based synchronization
        about 91
        example 92
        using command line 92
    full synchronization
        about 85
        example 86
        using command line 85
    incrementally after SRL overflow
        about 155
        using command line 157
    using bunker replication 267
synchronizing volumes
    about 141
    using difference-based synchronization 142
    using full synchronization 142
synchronous mode
    data flow 29
synchronous replication
    explanation 26, 55
    setting up 75
    synchronous=off 74
    synchronous=override 75
syncrvg
    vradmin keyword
        difference-based synchronization 91
        full synchronization 85
syncvol
    vradmin keyword 141

T

takeover
    vradmin keyword 249
taking over from failed Primary
    example 252–253
    overview 247
    using command line 249
tape backups
    block-level 81, 87
TCP protocol 70
terms
    replication
        in VVR context 22
traditional snapshot method 46
transferring Primary role
    failing back 255
    methods 239
    migrating 240
    taking over 247
trial failover
    example 376
tunables
    VVR
        changing. *See* using vxio.conf file. *See* using vxtune
tunables, VVR
    displaying 123
Turn off
    SmartMove for VVR 93

U

UDP protocol 70
unfreezing replication
    after snapshot 187
    using vxibc unfreeze 371
unrecoverable I/O error 388
up-to-date data
    concept 22
up-to-date Secondary
    determining 132
update ID
    displaying 132
updates
    vxrlink keyword 132, 319
updating Secondary
    using command line 279
usage types
    VVR volumes 66
utility
    vrnotify 134
    vxrlink stats 129
    vxtune 218
utility error messages 341

V

VCS
  bunker replay
    bunker replication 286
VCS agents
  automated failover 284
VCS agents for VVR 32
verify
  vradmin keyword 139
  vxrlink keyword 319
verifydata
  vradmin keyword 208
verifying DR readiness
  Failover 206
  Firedrill 207
  offline data validation 210
  online data validation 208
  overview 206
verifying secondary data 207
verifying volumes
  about 207
  before adding to RDS 140
  using offline data verification 210
  using online data verification 208
viewing
  consolidated statistics 120
  data volume statistics 121
  data volumes in RDS 105
  flag settings 115
  memory tunable parameters
    statistics 123
  network bandwidth used by VVR 124
  replication status 107
  RLINK statistics 120
  RVG and RDS information 103
  SRL volume statistics 122
  status of individual RVG 104
  status of specific RLINK 114
  VVR statistics 119
vol_max_nmpool_sz tunable 131
Volume Manager
  VVR as option 15
volume set
  associating to an RDS 143
volumes
  associating to an RDS
    using command line 137
  removing or dissociating from an RDS
    using command line 152

volumes *(continued)*
  resizing
    about 149
    using command line 151
  synchronizing
    using difference-based
      synchronization 142
    using full synchronization 142
  verifying
    before adding to RDS 140
    using offline data verification 210
    using online data verification 208
volumes, synchronizing
  about 141
vradmin command
  command reference 311
  error messages 343
vradmin utility
  addsec 68
  addvol 138
  changeip 165
  createpri 66–67
  creating RVGs 65
  delpri 178
  delsec 177
  delvol 152
  fbsync 251, 257, 260
  ibc
    compared with vxibc 367
    using off-host processing 223
  makesec 262
  migrate 241
  pauserep 163
  printrvg 103, 350
  printvol 105
  repstatus 107
  resizesrl keyword 161
  resizevol 149
  resumerep 163
  resync keyword 155
  set 73
  set keyword 163
  startrep 83
  stoprep 164
  syncrvg
    difference-based synchronization 91
    full synchronization 85
  syncvol 139, 141
  takeover 249

vradmin utility *(continued)*
- verify 139
- verifydata 208

vradmind daemon
- errors 344
- restarting 120

vrnotify utility 134
vrport command 173
vrstat command 119
VVR
- features 17
- option of VxVM 15

VVR GUIs
- about 63

VVR tunables
- changing 216
- displaying 123

vxassist command
- addlog keyword 146
- growto keyword 162
- make keyword
  - associating DCM 146
  - creating cache volumes 189
  - creating snapshot volumes 186
- maxgrow keyword 342
- mirror keyword 159
- snapback keyword 203
- snapshot keyword 200
- snapstart keyword 200

vxconfigd
- daemon
  - messages 341

vxdg command
- free keyword 161
- split keyword 90, 100
- upgrade keyword 183

vxedit command
- rename keyword
  - for plex-breakoff snapshots 194
  - in Disk Split and Join 90
  - renaming volumes during RLINK recovery 294
- rm keyword 159
- set keyword
  - primary_datavol 392
  - setting synchronous=fail 74
- set remote 394

vxibc utility
- about 367

vxibc utility *(continued)*
- send command 370
- unfreeze command 371

vxmake command
- creating cache objects
  - for space-optimized snapshots 189
  - setting attributes 187
- creating RLINK 319
- creating RVGs 65
- creating volumes during plex recovery 159
- rlink keyword 393

vxmemstat command
- command reference 319
- versus vrstat command 119

vxnetd
- starting 329

vxplex command 158
vxprint command
- displaying primary_datavol field 393
- displaying VVR status 114

vxrlink att 386
vxrlink command
- and clusters 33
- assoc keyword 318
- att keyword 162, 318
- checkdelete keyword 180
- cplist keyword 118
- det keyword 159, 162, 318
- dis keyword 318
- pause keyword
  - creating Secondary checkpoint 179
  - RLINK FAIL state 388
  - using Secondary checkpoints 214
- recover keyword
  - command reference 318
  - two-phase transactions in 390
- restore keyword 215–216, 318, 388
- resume keyword 318
  - ending Secondary checkpoint 180
  - using Secondary checkpoints 214
- stats keyword
  - displaying network statistics 129
  - reference 319
  - versus vrstat 119
- status keyword
  - displaying status of RLINK 117
  - reference 319
  - synchronization after SRL overflow 157
- updates keyword 132, 319

vxrlink command *(continued)*
    verify keyword 319
    versus vrstat 119
vxrlink recover 386
vxrsyncd daemon
    ports 173
vxrvg command
    checkdelete keyword 180
    checkend keyword 180
    checkstart keyword 179
    command reference 316
    cplist keyword 118
    getdatavols keyword 106
    recover keyword 390
    reference 316–317
    snapback keyword
        reattaching instant snapshot plexes 195
        reattaching traditional snapshot plexes 202
        using FastResync 205
    snapdestroy keyword 199
    snapprint keyword 198
    snaprefresh keyword 195
    snaprestore keyword 196
    snapshot keyword
        breaking off mirrors 158
        creating instant snapshots 181
        creating traditional snapshots 200
        using FastResync 205
        using with IBC 374
    start keyword 385
    stop keyword 198
vxsnap command 186
vxstat command
    versus vrstat command 119
vxtune utility 218
VxVM
    VVR as option 15
vxvol command
    aslog keyword
        associating the SRL 162, 303
    assoc keyword
        associating a new volume 294
        during disaster recovery 159
        setting Primary volume name on Secondary volume 393
    dis keyword
        dissociating the SRL 162, 303
        during disaster recovery 159

vxvol command *(continued)*
    set keyword
        enabling FastResync 205
    start keyword
        during disaster recovery 159
        starting data volume 301
        starting SRL 303
vxvset command 144

W

write-order fidelity
    concept 22
writing to Secondary data volumes 239