

ReplicatorX™ 4.0

Theory of Operation and Planning Guide

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089 USA
Telephone: +1 (408) 822-6000
Fax: +1 (408) 822-4501
Support telephone: +1 (888) 4-NETAPP
Documentation comments: doccomments@netapp.com
Information Web: <http://www.netapp.com>

Part number 210-03658_A0
March 2007

Copyright and trademark information

Copyright information

Copyright © 1994–2007 Network Appliance, Inc. All rights reserved. Printed in the U.S.A.

No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted Network Appliance material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETWORK APPLIANCE “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL NETWORK APPLIANCE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Network Appliance reserves the right to change any products described herein at any time, and without notice. Network Appliance assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by Network Appliance. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of Network Appliance.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark information

NetApp, the Network Appliance logo, the bolt design, NetApp—the Network Appliance Company, DataFabric, Data ONTAP, FAServer, FilerView, Manage ONTAP, MultiStore, NearStore, NetCache, SecureShare, SnapDrive, SnapLock, SnapManager, SnapMirror, SnapMover, SnapRestore, SnapValidator, SnapVault, Spinnaker Networks, SpinCluster, SpinFS, SpinHA, SpinMove, SpinServer, SyncMirror, Topio, VFM, and WAFL are registered trademarks of Network Appliance, Inc. in the U.S.A. and/or other countries. Cryptainer, Cryptoshred, Datafort, and Decru are registered trademarks, and Lifetime Key Management and OpenKey are trademarks, of Decru, a Network Appliance, Inc. company, in the U.S.A. and/or other countries. gFiler, Network Appliance, SnapCopy, Snapshot, and The evolution of storage are trademarks of Network Appliance, Inc. in the U.S.A. and/or other countries and registered trademarks in some other countries. ApplianceWatch, BareMetal, Camera-to-Viewer, ComplianceClock, ComplianceJournal, ContentDirector, ContentFabric, EdgeFiler, FlexClone, FlexShare, FlexVol, FPolicy, HyperSAN, InfoFabric, LockVault, NOW, NOW NetApp on the Web, ONTAPI, RAID-DP, RoboCache, RoboFiler, SecureAdmin, Serving Data by Design, SharedStorage, Simplicore, Simulate ONTAP, Smart SAN, SnapCache, SnapDirector, SnapFilter, SnapMigrator, SnapSuite, SohoFiler, SpinMirror, SpinRestore,

SpinShot, SpinStor, StoreVault, vFiler, Virtual File Manager, VPolicy, and Web Filer are trademarks of Network Appliance, Inc. in the United States and other countries. NetApp Availability Assurance and NetApp ProTech Expert are service marks of Network Appliance, Inc. in the U.S.A.

Apple is a registered trademark and QuickTime is a trademark of Apple Computer, Inc. in the United States and/or other countries. Microsoft is a registered trademark and Windows Media is a trademark of Microsoft Corporation in the United States and/or other countries. RealAudio, RealNetworks, RealPlayer, RealSystem, RealText, and RealVideo are registered trademarks and RealMedia, RealProxy, and SureStream are trademarks of RealNetworks, Inc. in the United States and/or other countries.

All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such.

Network Appliance is a licensee of the CompactFlash and CF Logo trademarks.

Network Appliance NetCache is certified RealSystem compatible.

Table of Contents

	Preface	vii
	About this guide	vii
	Audience	vii
	Special messages	vii
Chapter 1	Introduction	1
	ReplicatorX, a flexible and optimal solution.	2
	The challenge: replication and disaster recovery.	2
	About the ReplicatorX solution	4
	Consistency with ReplicatorX	7
	Key ReplicatorX components	8
	How ReplicatorX works	16
	Recovering from a disaster.	29
Chapter 2	Planning and Configuration	31
	Overview	31
	ReplicatorX requirements	31
	Resources and capacity.	32
	Glossary	51

Preface

About this guide This guide provides an introduction to ReplicatorX™ software and an overview of the system configuration.

Audience This guide is for system administrators with a working knowledge of each supported platform and its operating environment.

Special messages This guide contains special messages that are described as follows:

Note

A note contains important information that helps you install or operate the system efficiently.

Caution

A caution contains instructions that you must follow to avoid damage to the equipment, a system crash, or loss of data.

WARNING

A warning contains instructions that you must follow to avoid personal injury.

This chapter introduces ReplicatorX™ and describes how its features, operating concepts and components work together to provide reliable system-wide data consistency for disaster recovery in your business.

It provides the following information:

- ◆ “[The challenge: replication and disaster recovery](#)” on page 2
- ◆ “[About the ReplicatorX solution](#)” on page 4
- ◆ “[Consistency with ReplicatorX](#)” on page 7
- ◆ “[Key ReplicatorX components](#)” on page 8
- ◆ “[How ReplicatorX works](#)” on page 16
- ◆ “[Recovering from a disaster](#)” on page 29

ReplicatorX, a flexible and optimal solution

NetApp's ReplicatorX is the first enterprise-wide, consistent disaster recovery solution for heterogeneous storage environments, covering large geographical areas, and using low-cost IP networks.

Unlike most other solutions, ReplicatorX supports heterogeneous environments, guaranteeing data consistency across SAN-attached storage devices, direct-attached storage (DAS) devices, storage subsystems, and storage virtualization schemes alike.

ReplicatorX minimizes the performance impact to your applications and network, and supports the most demanding recovery time objective (RTO) and recovery point objective (RPO).

ReplicatorX dramatically reduces total cost of ownership (TCO) by:

- ◆ Providing a single solution for enterprise-wide replication management.
- ◆ Reducing outlays for licensing, maintenance, and training.
- ◆ Easing the work of IT staff, and improving their productivity.
- ◆ Enabling freedom of choice in storage systems and servers used.
- ◆ Using cost-effective IP infrastructures.

With ReplicatorX, managing the replication of your company's growing data is easy and intuitive. ReplicatorX offers outstanding scalability, remote accessibility, and extensive policy-based management.

The challenge: replication and disaster recovery

Your company is dependent on its electronic data. Losing this data – or even temporary loss of access to this data – can have a significant negative impact on business. Today, even periodic, planned outages for backups and maintenance present increasingly challenging demands on the 24/7 availability of your company's data.

Increasingly, replication is the favored solution for protecting and improving the availability of enterprise data. Using this method, data is copied continuously from a (local) Primary Site to a (remote) Secondary Site, in such a way that it can be made available immediately in the event that the Primary Site becomes unavailable. The key requirement for the replication solution is a continuous, consistent replication of the data from a Primary Site – containing many hosts and Volumes with many different data dependency needs – to a set of Volumes on a Secondary Site.

Within the replication solution, there are two main approaches: to replicate the data synchronously or asynchronously:

- ◆ Synchronous Approach

In the synchronous approach, each Primary Site application write is replicated to the Secondary Site before the application can complete the write operation at the Primary Site.

Synchronous replication is relatively straightforward, and inherently maintains full data consistency, since the updates in the Secondary Site are performed synchronously to the updates in the Primary Site. The main disadvantage of this approach is serious performance degradation to the applications running on the Primary Site. This degradation becomes intolerable over longer distances, thus limiting synchronous replication to short distances, and excluding it completely for high performance environments and large-scale disasters.

- ◆ Asynchronous Approach

In the asynchronous approach, Primary Site application writes are replicated to the Secondary Site without requiring the application to wait until each write replication is completed.

Asynchronous replication eliminates performance degradation problems, and enables locating the Secondary Site at any distance from the Primary Site. The main challenge of this approach is in achieving a Consistent image of the data within a reasonably small time gap. Emerging SAN configurations present an even greater challenge for maintaining data consistency during asynchronous replication than that presented by traditional direct attached storage (DAS) configurations.

Many companies use a hybrid of both synchronous and asynchronous approaches. They are willing to endure the performance degradation imposed by the synchronous method for a small, critical portion of their data. However, for most data, the asynchronous approach, together with a reasonable time gap, is often preferred.

Disaster recovery solutions also must be cognizant of today's replication architectures, which may be divided into two main groups:

- ◆ Storage Subsystem-based Architecture

Storage subsystem solutions are based on hardware or firmware embedded in storage devices at Primary and Secondary sites. These solutions are proprietary, operating only among homogeneous storage subsystems of the same family, and limiting company IT operations to a specific vendor. By nature, storage subsystem solutions do not support multi-vendor storage or cross-subsystem consistency.

- ◆ **Server-based Architecture**

Server-based solutions support heterogeneous, multi-vendor storage environments. However, typically, server-based solutions also require lock-in to a specific platform or Volume manager. If the company maintains multiple platforms, then it must implement and operate multiple disaster recovery solutions in parallel, in order to ensure enterprise-wide disaster recover protection. Most of today's server-based solutions do not support cross-server consistency. Finally, server-based solutions are not easily scalable.

Of course, as a company grows, it is impractical to assume that all of its data will be manageable by a single storage subsystem, or single-vendor server software. Even more challenging today is the dramatically increased use of SANs, storage virtualization, and heterogeneous storage subsystems – environments where logical Volume management requirements are inherently different for servers, applications, and end users.

About the ReplicatorX solution

NetApp's ReplicatorX successfully exploits the concurrent nature of SAN and co-exists with Volume manager virtualization schemes. ReplicatorX replicates logical Volumes and guarantees consistent replication across Volumes.

ReplicatorX is foremost a disaster recovery solution, based on asynchronous data replication. In addition, ReplicatorX leverages its replication capabilities to provide additional special features, such as creation of a Consistent backup image from the Secondary Site, and data mining on the Secondary Site. These features are available concurrently with the ongoing replication required for the disaster recovery solution.

Support for existing infrastructures:

- ◆ **Storage Subsystem Agnostic**

ReplicatorX supports any storage subsystem from any vendor. The storage subsystem can be either a simple, low cost, RAID array, or an intelligent and scalable one (for example, EMC Symmetrix). Heterogeneous storage subsystem infrastructures are fully supported.

- ◆ **SAN Agnostic**

ReplicatorX supports any storage interconnect, from basic direct attached storage to complex SAN infrastructure, including fiber channel SAN and future IP-based SAN.

- ◆ **Storage Virtualization Agnostic**

ReplicatorX operates with any SAN virtualization and/or Volume management solutions.

- ◆ **Exploitation of Existing IP Network**
ReplicatorX exploits the existing network infrastructure for the propagation of data to the Secondary Site.
- ◆ **Server Platform/OS Agnostic**
ReplicatorX supports both homogeneous or heterogeneous Primary Site server configurations. This means that application servers can use any of the following:
 - ❖ The same platform or operating system.
 - ❖ Different platforms or operating systems.
 - ❖ Different server models from the same vendors.
 - ❖ Different server models from different vendors.

Minimal impact: Using asynchronous replication, ReplicatorX has a minimal impact upon overall performance of Primary Site servers.

ReplicatorX operates independently of actual Customer Data and does not process this data in any way. ReplicatorX considers only where and when this data is written in order to perform replication.

Disaster recovery: In the case of a disaster or a system-wide failure at a Primary Site, ReplicatorX brings all affected Volumes at the Secondary Site to a fully Consistent state in a timely manner.

Data center-wide consistency: Even in complex enterprise environments that contain multiple servers running multiple operating systems writing to multiple storage subsystem Volumes, ReplicatorX guarantees data consistency across both server and subsystem boundaries.

Off-site processing: Whenever prompted, ReplicatorX can bring Volumes at the Secondary Site to a Consistent, point-in-time image of their associated Volumes at a Primary Site. Then, off-site processing such as traditional backup to tape or data mining can take place on the Secondary Site. During off-site processing, ReplicatorX continues to monitor updates made at the Primary Site. As soon as off-site processing is completed, ReplicatorX re-synchronizes the Volumes, and replication activity continues.

Long distance IP replication: The use of standard IP connectivity means that ReplicatorX performs disaster recovery regardless of the distance between sites. IP is a robust and time-tested technology for replicating large amounts of data, cost effectively.

Data migration: Occasionally, mission-critical application data needs to be migrated to newer disk subsystems at a Primary Site. In some cases, the applications and their data need to be migrated to a Secondary Site. ReplicatorX

enables both local and remote data migration. During the migration process, designated applications continue their work with their original Volumes, as usual. When the migration is completed, work continues transparently on Volumes at the new location.

Fault tolerance: ReplicatorX sustains a wide range of failures and error conditions, including:

- ◆ Network outages
- ◆ A crash of one or more servers
- ◆ Power failures on either the Primary or Secondary sites
- ◆ Volume I/O errors

In each of these cases, ReplicatorX recovery is transparent, does not require full re-synchronization between Volumes at the Primary and Secondary Sites, and does not expose the Secondary Site to an inconsistency window.

Scalability: The ReplicatorX solution is scalable. It does not interfere with the data transfer path from application servers to storage devices. Normally, ReplicatorX does not add more I/O activity to a Primary Site's SAN, so ReplicatorX does not become a bottleneck as the SAN is scaled up.

Flexible replication: ReplicatorX enables the creation of multiple sets of Volumes that enable data to be replicated from a Primary Site to a Secondary Site. In addition, the same Secondary Site may be used to replicate several Primary Sites.

ReplicatorX also supports two synchronization methods to make the contents of Primary Site Volumes identical to the contents of corresponding Secondary Site Volumes:

- ◆ Online Synchronization

The contents of Primary Site Volumes are initially synchronized with their corresponding Secondary Site Volumes using the network, while allowing simultaneous, continuous operation of the Primary Site.

- ◆ Offline Synchronization

When it is more economical to transfer the contents of Primary Site Volumes by external means, ReplicatorX supports a copy-and-transport synchronization method (for example, using a backup tape), while still allowing simultaneous, continuous operation of the Primary Site.

Management tools: ReplicatorX provides a centralized, remotely accessible configuration and monitoring tool that eases management of replication activity. ReplicatorX uses extensive policy-based management features, and other automatic capabilities to eliminate many tedious configuration tasks. Statistical

information reporting provides the feedback necessary to enable system administrators to fine-tune ReplicatorX resource usage and behavior, optimizing overall system performance.

The current software version provides GUI-, CLI-, and SNMP-based management.

Consistency with ReplicatorX

By its very nature, asynchronous replication presents special challenges to the achievement of data consistency. In asynchronous replication, the image constructed on the Secondary Site is not necessarily identical to the image on a Primary Site, but it must be equivalent to an image that existed on the Primary Site at some point in time. This requirement is essential in order to enable applications to subsequently use the data residing on the Secondary Site.

Specifically, asynchronous replication must address two complex challenges to guarantee data consistency:

- ◆ The order of entries applied to the Target Volume.
- ◆ When it is safe to apply a particular entry.

To better illustrate the mathematical nature of these challenges, consider the following examples.

- ◆ Applying Updates to the Same Location

Assume that Data **X** was written to some Primary Site location and then replicated to the Secondary Site. Assume that subsequently, Data **Y** was written to the same Primary Site location and also transferred to the Secondary Site.

In order to create a Consistent image, we must ensure that if Data **Y** is applied first to the Secondary Site, Data **X** is never applied afterwards.

- ◆ Applying Updates to Different Locations

Assume that Data **A**, **B**, and **C** exist at three different locations on the Primary Site. Now assume that three Updates, **X**, **Y**, and **Z** are addressed to these locations respectively.

The order of data images that existed on the Primary Site was as follows:

A, B, C

X, B, C

X, Y, C

X, Y, Z

Any of the above four combinations appearing on the Secondary Site would be Consistent images of the Primary Site. However, the appearance of **A**, **Y**, **C**, for example, would not be a Consistent image.

Finally, assume that at some point in time, **Y** arrived at the Secondary Site, but **X** and **Z** did not arrive. In order to ensure that a Consistent image can be provided at any time, **Y** cannot be applied to the Target Volume.

ReplicatorX handles each of these challenges routinely and transparently, and guarantees that a Consistent image can be achieved at any moment.

Key ReplicatorX components

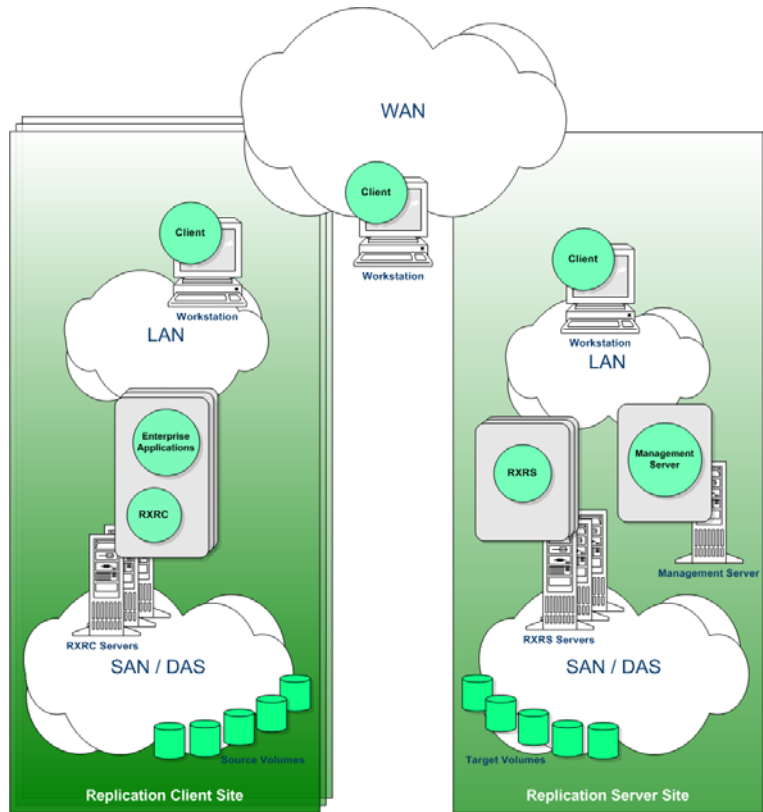
ReplicatorX is comprised of several key components. Each of these components, described below, is critical to the success of ReplicatorX's replication solution.

For the purposes of ReplicatorX and of this document, a local (or production) site is called a Primary Site, and a remote (or backup) site is called a Secondary Site.

Those Volumes located at a Primary Site containing data to be replicated are called Source Volumes. Similarly, Volumes located at the Secondary Site intended to store the replicated data are called Target Volumes.

Some of the key ReplicatorX components are referred to in this document as Services. On Unix platforms, these components are actually daemons, but for reasons of consistency, the term Service will be used for all platforms.

The following figure shows the key components operating in the ReplicatorX system.



Replication Network (RNET): The ReplicatorX replication network (RNET) is comprised of all LANs and WANs that are used to interconnect Primary and Secondary sites for purposes of replication. The primary function of the RNET is to transfer data between these sites with high efficiency, using the existing network bandwidth.

RXMS: The RXMS (ReplicatorX Management Server) is a software module that runs as a service, and resides on a single, standard server located at the Secondary Site.

Its main functions are to:

- ◆ Enable remote User interaction with ReplicatorX software.
- ◆ Maintain all required configuration information about the Primary and the Secondary Sites.

- ◆ Interact with Replication Clients and Replication Servers to configure, control, and monitor all ReplicatorX activities.

Although it is possible to install the RXMS on a server that also hosts a Replication Server, for performance reasons, it is recommended that the RXMS be installed on a standalone server.

Access to the RXMS is provided by the ReplicatorX Clients.

ReplicatorX Client: The Client is a ReplicatorX component that enables a User to interact with the RXMS in order to configure, control and monitor ReplicatorX activities.

The current software version provides graphical user interface (GUI), command line interface (CLI) Clients, and an SNMP client.

A ReplicatorX Client can be located anywhere, enabling universal access. In addition, multiple Clients can be installed on servers throughout a configuration.

Currently there are three available clients.

- ◆ ReplicatorX Management Client (RXMC) - a GUI client
- ◆ ReplicatorX CLI - a command line interface client
- ◆ ReplicatorX SNMP Agent (RXSA) - an SNMP client

ReplicatorX Replication Client (RXRC): The ReplicatorX Replication Client (RXRC) is a software module that resides on each Primary Site server which has access to Source Volumes. The RXRC is comprised of two software components: a device driver and a user-mode daemon/service.

Its main functions are to:

- ◆ Record, store, and supply write I/O and I/O-related information for its associated Source Volumes, required by ReplicatorX in order to achieve and maintain Consistent replication.
- ◆ Be responsible – together with the pertinent Replication Server – for transferring the recorded data from the Primary Site to the Secondary Site.

The set of all RXRCs at a Primary Site is collectively able to intercept all I/O commands addressed to Source Volumes at the Site. When storage virtualization is used, the RXRC is logically situated above the virtual Volume drivers of the Primary Site.

To achieve the above functions, the RXRC maintains two key resources, the Volatile Recording Repository (VRR) and Bitmaps.

Volatile Recording Repository (VRR)

The RXRC uses a volatile recording repository (VRR), a portion of physical memory on its host server, in order to record each intercepted write addressed to a Source Volume. The VRR stores each write until it can be transferred to the appropriate Secondary Site Replication Server (RXRS).

As might be expected, in an active Primary Site, the VRR is subjected to periods during which the stream of inputs is significantly larger than the stream of outputs. This results in increased utilization of the VRR, reducing the space available to record additional writes.

For this reason, the VRR operates in two distinct modes:

- ◆ **Full Recording Mode**

Each VRR entry contains the actual data of a write, called Customer Data, along with its I/O-related information, called Metadata. This is the normal operating mode for the VRR. Full VRR entries can be quite large, depending on the size of I/O generated by the application.

When the RXRC transfers a Full VRR Entry to the RXRS, it transfers all of its contents, both Customer Data and Metadata.

- ◆ **Slim Recording Mode**

Each VRR entry contains only Metadata. This operating mode dramatically increases the number of writes that can be stored in the VRR, since all Slim VRR entries are very small.

When the RXRC transfers a Slim VRR Entry to the RXRS, it first reads its corresponding Customer Data from the designated Source Volume, and then transfers all of its contents, both Customer Data and Metadata.

Slim Recording Mode is used during periods of stress on the VRR, when there is insufficient space in the VRR to accommodate all writes that are addressed to the Source Volumes. This can occur due to I/O peaks, momentary WAN outages, or increased WAN latency. Using Slim recording, ReplicatorX continues replication activity transparently and maintains full data consistency. When stress on the VRR has subsided and enough recording space is available once again, the VRR automatically switches from Slim Recording Mode back to Full Recording Mode.

While Full and Slim recording mechanisms protect the VRR, there can be periods when continuous input streams (or a long network outage) creates prolonged stress. The result of such periods can be that there is not enough free space available in the VRR to record even Slim entries. In such cases, ReplicatorX relies on its Bitmap. Note that if ReplicatorX resources are configured properly, these periods are rare.

Bitmaps

The RXRC maintains a Bitmap for each of its associated Source Volumes. Each bit in a Bitmap maps to a pre-defined extent in its designated Source Volume. When a write occurs within that extent on the Source, its corresponding bit is set in the Bitmap.

ReplicatorX uses this Bitmap to provide Source Volume write information in the event of a VRR overload. In such a scenario, the RXRC uses Bitmap extent information to determine which write I/O commands have not yet been propagated to the RXRS.

When the Bitmap hardening capability is used, Bitmaps are also maintained in a Bitmap Volume, which is used in the event of an RXRC failure. This means that after the RXRC restarts, the data contained in the Bitmap Volume persists, eliminating the need for a full re-synchronization of the Volume.

Bitmaps are handled with the utmost efficiency. ReplicatorX is able to discern situations where it is not necessary to write to a Bitmap Volume (for example, when an extent bit has been set already by a previous write), and when to join multiple set operations to the same write. In addition, ReplicatorX can determine when it is optimal to reset bits, and uses this background mechanism to eliminate unnecessary hardening activity.

Each RXRC can support multiple Bitmap Volumes, and each Bitmap Volume can support multiple Bitmaps for multiple Source Volumes.

Note

ReplicatorX supports Bitmap Volumes which are represented by files (instead of actual volumes). This option is useful for systems with a limited number of volumes. This option is available only for Windows 2003.

ReplicatorX Replication Server (RXRS): The ReplicatorX Replication Server (RXRS) is a software module that runs as a daemon/service, and resides on one or more standard servers located at the Secondary Site. When storage virtualization is used, the RXRS is logically situated above the virtual Volume drivers of the Secondary Site.

The main functions of the RXRS are to:

- ◆ Be responsible – together with the pertinent RXRCs – for transferring recorded data from the Primary Site to the logs.
- ◆ Harden all incoming data to the Log and send acknowledgement of receipt to the associated RXRCs.

- ◆ Ensure that the combination of data already applied to the Target Volumes, together with data residing in the associated logs is sufficient to bring the Target Volumes to a Consistent state, whenever required.
- ◆ Be responsible for transparently moving replicated data from logs to Target Volumes.
- ◆ Manage the configured replication Sessions (described on “Sessions” on page 16), Log resources, communications with the pertinent RXRCs, and responses to commands sent by the RXMS.

A Secondary Site can contain multiple RXRS components. Similarly, a single RXRS can communicate with multiple RXRCs and manage multiple Sessions.

To achieve the above functions, the RXRS maintains a Log.

Log

On each RXRS, the Log is used to accumulate incoming data transferred from its associated RXRCs but that has not yet been applied to the appropriate Target Volumes.

The Log is comprised of three key resources:

- ◆ **Log Cache**

To improve performance, the RXRS uses a portion of the host memory to cache both Customer Data and Metadata that were transferred to the Secondary Site.

As long as the Log Cache can accommodate all of the data that is being propagated, Customer Data is read directly from the Log Cache.

- ◆ **UD-Log Volume**

The UD-Log Volume is used to store Customer Data transferred from an associated RXRC, before it is applied to its Target. Should an RXRS fail, this data persists.

One or more UD-Log Volumes must be assigned to each RXRS in your configuration.

- ◆ **MD-Log Volume**

The MD-Log Volume is used to store Metadata. Should an RXRS fail, this data persists.

One or more MD-Log Volumes must be assigned to each RXRS in your configuration. Unlike Customer Data, stored Metadata is always uniform in size and requires relatively little space.

Log space resource Management

The User Data Log Space (UD-Log Space) is the storage resource available to an RXRS for storing the user-data. UD-Log Space can be managed according to the following management methods:

- ◆ Pure global resource management
- ◆ Pure local resource management
- ◆ A combination of global and local resource management

In **pure global resource management**, the available UD-Log Space is shared by all the Sessions maintained by the RXRS. This means each Session has access to the UD-Log Space that it needs, unless all the UD-Log Space is being used by the other Sessions.

The advantage of pure global resource management is that if one of the Sessions is in trouble, this Session can use much more UD-Log Space than it would have been allocated in pure local resource management. The disadvantage of the pure global method is that a Session in serious trouble log space-wise, will eventually hurt the quality of service of other Sessions.

In **pure local resource management**, a limited UD-Log Space is allocated for each Session. This means that each Session has a pre-defined and limited UD-Log Space, regardless of the current needs of that Session. In this case, there is no sharing of UD-Log Space between Sessions, and what happens to one Session has no affect on the other sessions.

The advantage of local resource management is that it avoids the problem of a single Session using all the available UD-Log Space of the system. However, the local resource management is not the ideal solution in most cases.

To address these issues, ReplicatorX enables RXMCs (ReplicatorX Management Clients) to define Session-level UD-Log Space to provide **a combination of global and local resource management**. For this purpose, RXMCs can define the following parameters for each Session:

- ◆ Log Quota
- ◆ Private Log Space

A Session's **Log Quota** is the upper limit for the Session's UD-Log size. This means that the Session's UD-Log size cannot be larger than the defined Log Quota. By defining a Log Quota for each session, the RXMC can ensure that no single session will grow to an inordinate size and leave the other sessions without enough UD-Log Space.

Note

The sum total of Log Quota for all the Sessions may exceed the total UD-Log Space.

A Session's **Private Log Space** is a quantity of UD-Log Space that is reserved for that Session. The Session is not required to use all of this space, but it is always available to that Session, in case of need. By defining a Private Log Space for a Session, the RXMC ensures that no matter what is happening with the other Sessions, this Session will always have a minimum amount of UD-Log Space available. Note that the sum total of Private Log Spaces for all the Sessions cannot exceed the total UD-Log Space.

A RXRS' **Public Log Space** is the RXRS' UD-Log Space that is not private (i.e., $\text{Public Log Space} = \text{UD-Log Space} - \text{The sum of Private Log Space}$).

For example:

The total UD-Log Space is 200GB, and there are 40 Sessions (S1..S40). All the Sessions are equally important, and each has a Private Log Space of 1GB. In addition, each of the Sessions has a Log Quota of 20GB. The combination of these definitions protects each of the Sessions from being halted by "troubled" Sessions. In addition, the RXRS Public Log Space ($200\text{GB} - 40 * 1\text{GB} = 160\text{GB}$) and the defined Log Quota enables at least eight Sessions to accumulate a significant amount of UD-Log Space before affecting the other Sessions ($8 * 20\text{GB} = 160\text{GB}$).

The Session's Log Quota and Private Log Space can be modified throughout the Session's lifetime, enabling RXMCs to fine-tune performance and adapt to changing conditions and needs.

ReplicatorX Remote Agent (RXRA): The ReplicatorX Remote Agent (RXRA) is a software module that runs as a daemon/service and resides on each of the servers on which an RXRC, RXRS, or RXMS is installed. Only one instance of the RXRA is installed on each server, even if more than one ReplicatorX module is installed on that server. The RXRA interacts with the RXMS and RXMC and executes tasks on their behalf.

The RXRA's main functions are:

- ◆ To perform operations on and query the status of the other ReplicatorX services/daemons.
- ◆ To modify the configuration parameters of the ReplicatorX modules.

Pairs, Sessions, and Consistency Collections: Pairs, Sessions, and Consistency Collections are logical entities, used by ReplicatorX to represent replication relationships.

Pairs and Sessions are created on demand as you configure and use ReplicatorX. They can be changed at any time. Consistency Collections are created and maintained automatically by ReplicatorX as part of the replication process.

Pairs

A Pair represents a replication relationship between a Source Volume and a Target Volume.

You create a Pair using the ReplicatorX Client. A Source or Target Volume can be a member of only one Pair simultaneously. At the time you create a Pair, you assign it to a new or existing Session, and determine how it will be synchronized initially. You also determine whether Bitmap hardening is used for the Pair.

Sessions

Each ReplicatorX Pair is assigned to a replication set, called a Session. ReplicatorX can maintain multiple Sessions between Primary and Secondary sites.

Generally, a Session is comprised of Pairs whose data are mutually dependent. For example, you may define a Session to include all the Volumes that are used by a single database. Each Pair in the Session would contain a portion of the data needed by the database, and ReplicatorX would be able to maintain this database in a Consistent state at all times.

You create a Session using the ReplicatorX Client. Each Session is managed by a single RXRS.

Consistency Collections

Each Session has a sub-group called a Consistency Collection, which is comprised of all the Pairs in the Session that are currently Consistent. When a Session is first defined, its Consistency Collection is empty. Upon activation of a Session, ReplicatorX attempts to include all assigned Pairs in its Consistency Collection.

How ReplicatorX works

ReplicatorX works to maintain data consistency during replication.

This section describes the replication data flow. A thorough understanding of this flow is an important basis for:

- ◆ Capacity planning
- ◆ Site configuration
- ◆ ReplicatorX operation

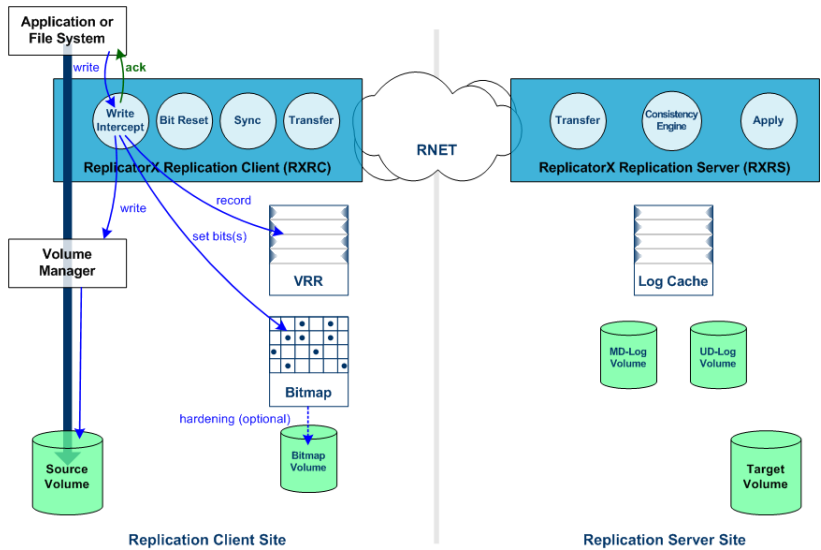
ReplicatorX operation is presented on the following pages in terms of the following key processes and their related data flows:

- ◆ “[Write Intercept](#)” on page 18
- ◆ “[Transfer](#)” on page 18
- ◆ “[Data reduction](#)” on page 20
- ◆ “[Bit reset](#)” on page 22
- ◆ “[Consistency Levels and the Consistency Engine](#)” on page 23
- ◆ “[Apply](#)” on page 25
- ◆ “[Freeze and Unfreeze](#)” on page 25
- ◆ “[Initial Online Synchronization](#)” on page 27
- ◆ “[Re-synchronization](#)” on page 28
- ◆ “[Consistency Standstill](#)” on page 28

For details on operating:

- ◆ The Management GUI (RXMG), see the *ReplicatorX 4.0 Administration Guide*.
- ◆ The ReplicatorX Management CLI (RXMC), see the *ReplicatorX 4.0 Administration Guide*.
- ◆ The ReplicatorX SNMP Messaging (RXSA), see the *ReplicatorX 4.0 Administration Guide*.

Write Intercept: The Write Intercept process occurs when an application or file system writes to a Source Volume on the RXRC.



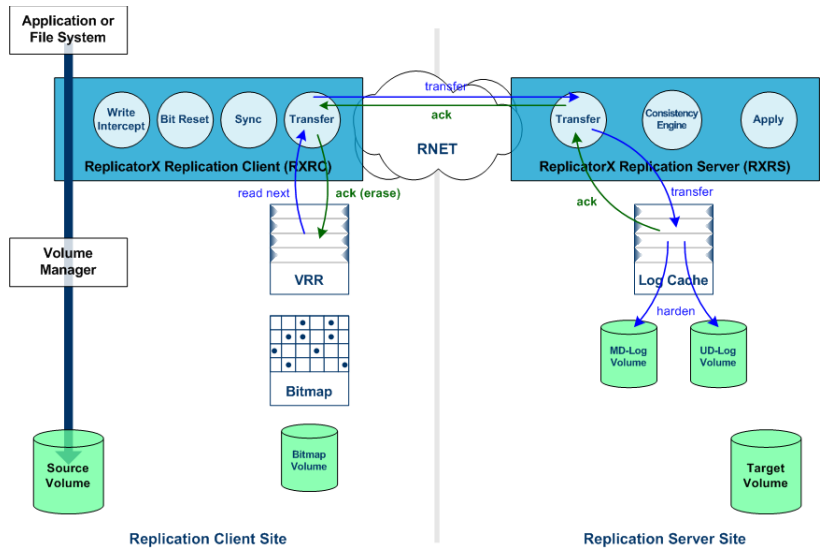
The following occurs during a Write Intercept process:

1. The RXRC intercepts each write addressed to a Source Volume and performs the following:
 - ❖ Sets the bit range in the Bitmap for that Source Volume.
 - ❖ Passes the write I/O on to the Source Volume.
 - ❖ Records Customer Data and Metadata in the VRR. (During Slim Recording, only the Metadata is recorded in the VRR.)
2. If the Bitmap hardening capability is active, the modified Bitmap information is also written to the Bitmap Volume. ReplicatorX is capable of discerning when a bit has already been set in the Bitmap, and therefore will write to the Bitmap Volume only when necessary.
3. When both ReplicatorX activity and the original write activity are completed, the RXRC acknowledges the write. (ReplicatorX does not need to wait for the data to be transferred to the RXRS in order to acknowledge the write.)

Transfer: The Transfer process occurs when data is passed from the VRR to the designated logs on the RXRS. ReplicatorX transfers two types of entries: Full Entries and Slim Entries.

Transfer Full entries

By default, ReplicatorX records and transfers Full entries. A Full entry contains both Customer Data and Metadata for a write.



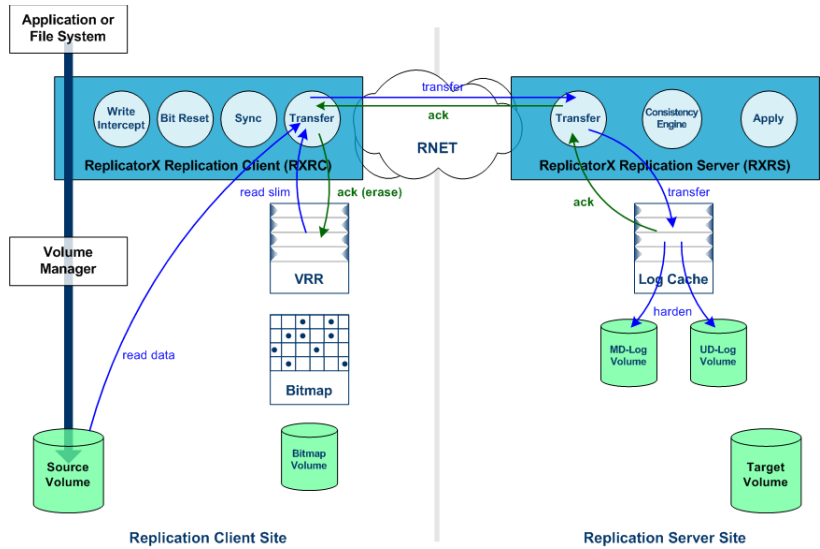
The following occurs during the transfer of Full entries:

1. The RXRS reads each Full entry from the VRR and transfers it to the designated RXRS.
2. The RXRS transfers each entry to its appropriate Log Cache. This includes the following:
 - ❖ Hardening Customer Data to the UD-Log Volume.
 - ❖ Hardening Metadata to the MD-Log Volume.
3. When hardening on the RXRS is complete, acknowledgement is sent to the RXRC and the VRR entry is erased.

To enhance performance, ReplicatorX is capable of transferring multiple entries before an acknowledgement is sent. Similarly, ReplicatorX is capable of acknowledging multiple entries simultaneously.

Transfer Slim entries

ReplicatorX records Slim entries when the space available to record additional writes in the VRR is approaching its capacity. A Slim entry contains only Metadata information about a write.



The following occurs during the transfer of Slim entries:

1. The RXRC reads a Slim entry from the VRR.
2. The RXRC reads the associated Customer Data from the Source Volume to the VRR, which transfers it to the designated RXRS.
3. The RXRS transfers each entry to the appropriate Log Cache. This includes the following:
 - ❖ Hardening Customer Data to the UD-Log Volume.
 - ❖ Hardening Metadata to the MD-Log Volume.
4. When hardening on the RXRS has completed, acknowledgement is sent to the RXRC, and the VRR entry is erased.

Data reduction: Data reduction enables you to reduce the amount of data ReplicatorX sends over the network. Data reduction is configured per Pairs or Sessions. The data reduction features operate on the VRR queues. Data reduction reduces network traffic and log space.

The following are the data reduction features:

- ◆ **Compression.** Compresses the data on the RXRC before it is sent to the RXRS. This is the only time ReplicatorX is concerned with the actual data. On the RXRS, the data is uncompressed before it is applied to the Target Volume.

Note

The compression feature should be used with discretion since it places stress on the CPUs of both the RXRC and RXRS.

- ◆ **Duplicates Removal (DPR).** Scans Source writes on the RXRC within a time interval that are addressed to the same block extents and sends only the last update. You define the time interval length. The RXRC batches data transmissions to the RXRS, and within a batch, eliminates redundant updates to a given LBA. The RXRS does not attempt to create a consistent timeline within this time interval.

However, the RXRC imposes an RPO of at least the time interval. The time interval is defined per Session to enable the RXRS to find a timeline for the entire Session.

For each Pair, you define whether to always enable the DPR feature, to disable it, or to use it only on Slim entries (where the RPO is increasing anyway).

DPR must be turned off before issuing a **FreezeTimeNow** command.

The following is an example of DPR:

If in a time period, the following data is written to the following locations in the Volume being replicated (each number represents a block):

```
11111111111111111111
      22222
                33333333
                    44
                                  555555
```

The end result of these writes is: 4411112223333355555

If DPR is disabled, the following data is passed:

```
entry 1 = <start time><start location>11111111111111111111<end time>
entry 2 = <start time><start location>22222<end time>
entry 3 = <start time><start location>33333333<end time>
entry 4 = <start time><start location>44<end time>
entry 5 = <start time><start location>555555<end time>
```

If DPR is enabled, and the above writes are in the DPR delay period, then the DPR passes only the following data:

entry 1 = <start time><start location>1111

entry 2 = <start location>222

entry 3 = <start location>33333

entry 4 = <start location>44

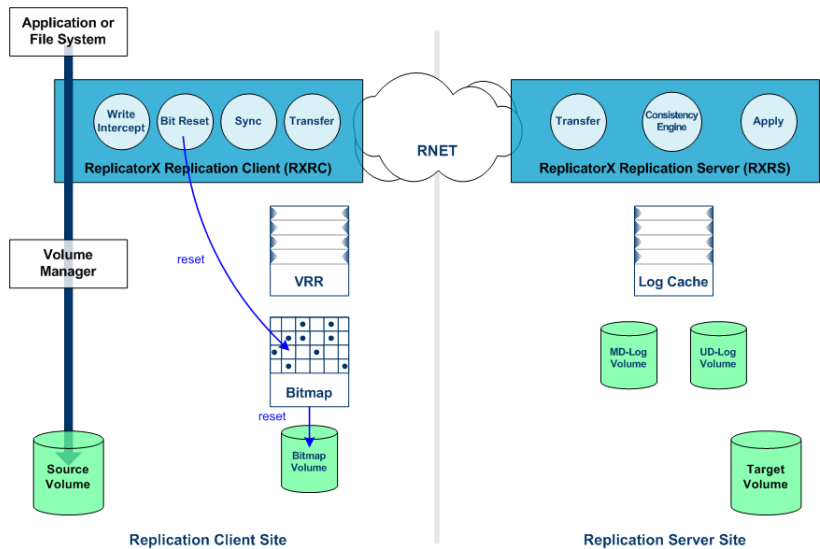
entry 5 = <start location>555555<end time>

Bit reset: Each time an update is addressed to a Source Volume, corresponding bit(s) are set in its Bitmap. If no action is ever taken to reset these bits, eventually, all bits in the Bitmap will be set, thus rendering the Bitmap useless, should ReplicatorX be required to perform a partial re-synchronization.

Consider the following issues before performing a bit reset:

- ◆ Multiple I/O activities may cause the same bit to be set. ReplicatorX must be able to determine when it is possible to reset this bit.
- ◆ Although it may be possible to reset a bit, it may not in fact be optimal to perform the reset. For example, when an application constantly writes to the same disk location, its bit needs to constantly be set. However, if ReplicatorX constantly resets this bit, then the bit will also require constant hardening, and hinder performance. Therefore, ReplicatorX must also be able to determine when it is worthwhile to reset a bit.

ReplicatorX uses a proprietary background mechanism to optimize all bit resets, and is capable of discerning when a reset is most effective and worthwhile



Consistency Levels and the Consistency Engine: At any given moment, a Pair is assigned a Consistency Level that describes its state in terms of consistency. A Pair may be at one of three Consistency Levels:

- ◆ **Inconsistent**

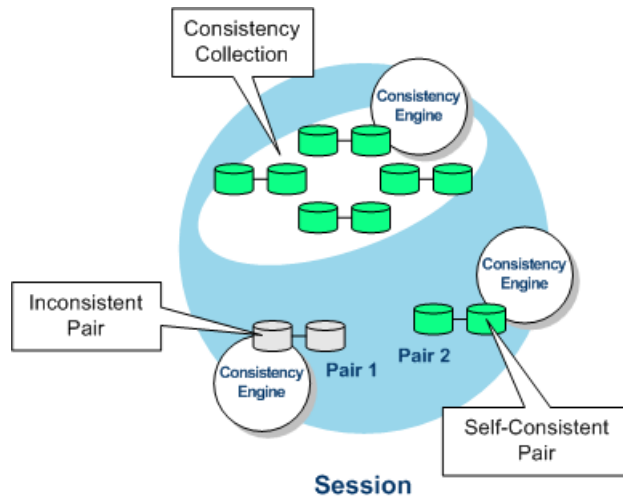
The Pair works in a state that does not guarantee consistency. That is, the combination of the Log and the Target Volume contents will not be sufficient to bring the Target Volume to a Consistent image. When working in this state, the potential to bring the Pair to consistency is still guaranteed, because the correct order of writes is always preserved.
- ◆ **Self-Consistency**

For any Pair working in this state, the combination of the Log and the Target Volume contents is sufficient to bring the Target Volume to a Consistent image.
- ◆ **Consistency Collection**

The Pair belongs to the Consistency Collection for a designated Session. The combination of the Log and the Target Volume contents for all of the Pairs is sufficient to bring all of the members Pairs to a mutual Consistent state.

Not all data that is accumulated in the Log may be applied to a Target Volume. As described in “[Consistency with ReplicatorX](#)” on page 7, there are specific criteria for ensuring that applied data is always Consistent.

ReplicatorX uses a Consistency Engine in order to preserve consistency. As shown in the following figure, each Replication Object has exactly one Consistency Engine. This Engine examines the Metadata associated with each Log entry for an Object, and collectively determines the Object’s Consistency Timeline – a timestamp that tells ReplicatorX which entries can safely be applied to the Target Volume. All entries with a timestamp earlier than the Consistency Timeline can be applied; all entries with a timestamp later than the Consistency Timeline cannot be applied. During routine operation, the Consistency Engine works continuously to advance the Consistency Timeline.

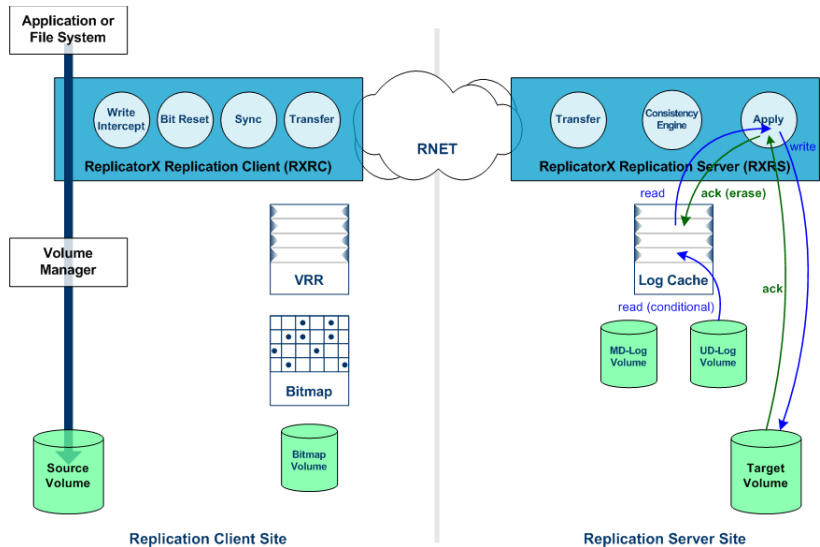


The Apply process (see “Apply” on page 25) is responsible for determining the actual order in which entries are applied. Together, the Consistency Engine and the Apply process guarantee that a Target Volume can always be brought to a Consistent image.

A Pair works at the Inconsistent level during Initial Online Synchronization, or upon explicit User request (typically when resolving a Consistency Standstill (see “Consistency Standstill” on page 28)). A Pair works at the Self-Consistent level as an intermediate step toward joining a Consistency Collection.

Apply: The Apply process occurs when Customer Data is written to a Target Volume on the RXRS.

ReplicatorX writes data to a Target Volume up to the Consistency Timeline calculated by a corresponding Consistency Engine. Following the Apply, the hardened data is erased from the Log.



The following occurs during the Apply process:

1. The RXRS reads data from the Log Cache, up to the given Consistency Timeline for a Consistency Object. If the data is no longer stored in the Cache, the RXRS reads it from the UD-Log Volume.
2. The RXRS writes the data to the appropriate Target Volume.
3. When hardening on the Target Volume is complete, the RXRS erases the hardened entry from the Cache and the UD-Log Volume.

Freeze and Unfreeze: During routine operation, data is continuously copied from the Log to the Target Volume using the Apply process. At any moment, though the contents of a Target Volume may not necessarily be Consistent, the combination of the Log and Target Volume contents is sufficient to bring the Target Volume to a Consistent image upon request.

Other Secondary Site servers, or applications running on the RXRS, are not allowed to access the Target Volume. However, there are times when customer applications require access to a Target, and expect it to contain a Consistent image. Such access is needed when recovering from a disaster and during offsite processing (for example, creating a remote backup to tape).

In order to provide this capability, a User requests a Freeze. When invoked, the Freeze process brings corresponding Target Volumes to a Consistent image, stops the Apply process, and then allows customer applications to access the Target Volumes. The Freeze process attempts to bring a Target to the most recent available Consistency Timeline.

Typically, following offsite processing, a User will want to continue routine replication activity. The User requests an Unfreeze. When invoked, the Unfreeze process resumes the Apply process. Once the designated Target Volumes are unfrozen, customer applications are again not allowed access to them.

Note

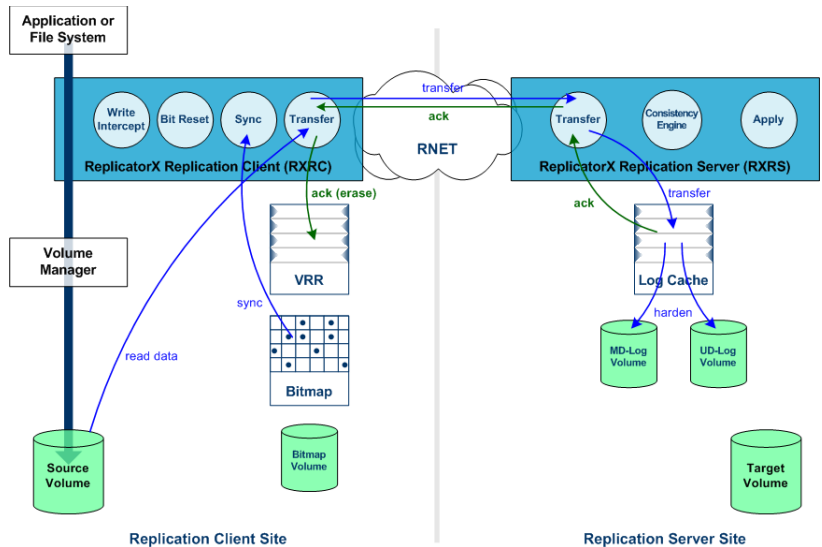
If a Target Volume has been altered in any way during a Freeze, you should not perform an Unfreeze. In order to continue replication in such a scenario, you must remove the affected Pair and re-synchronize it.

For more detailed information on:

- ◆ Freeze, see the *ReplicatorX Administration Guide*.
- ◆ Unfreeze: see the *ReplicatorX Administration Guide*.

Initial Online Synchronization: The Initial Online Synchronization process occurs when a Session is activated with a request to copy all data from Source to Target Volumes.

ReplicatorX synchronizes data concurrently as it replicates new updates from the Source to the Target Volumes.



The following occurs during the Initial Online Synchronization process:

1. Upon activation, the RXRC sets all bits in the Bitmap associated with the Source Volume to be synchronized.
2. The RXRC examines each bit in the Bitmap and reads the appropriate Customer Data extent from the Source Volume.
3. The RXRC transfers the Customer Data to the designated RXRS.
4. The RXRS transfers each entry to the appropriate Log Cache. This includes the following:
 - ❖ Hardening Customer Data to the UD-Log Volume.
 - ❖ Hardening Metadata to the MD-Log Volume.
5. When hardening on the RXRS is complete, acknowledgement is sent to the RXRC, and the affected bits are reset in the Bitmap (using the background mechanism described above in “Bit reset” on page 22).

Re-synchronization: Re-synchronization uses essentially the same data flow as that of “[Initial Online Synchronization](#)” on page 27. However, the following differences exist:

- ◆ Not all bits in the Bitmap are set. ReplicatorX must examine each bit and synchronize only the Customer Data extents that are indicated by set bits. As noted above in “[Bit reset](#)” on page 22, ReplicatorX is capable of optimizing its re-synchronization operation, ensuring that the minimum data will be transferred.
- ◆ In the event of an RXRC failure, after restart, the RXRC reads hardened Bitmap data from the Bitmap Volume. If the Bitmap hardening capability was not activated for an associated Pair during configuration, ReplicatorX performs a full synchronization, exactly as described in “[Initial Online Synchronization](#)” on page 27.

As with “[Initial Online Synchronization](#)” on page 27, ReplicatorX re-synchronizes data concurrently as it replicates new updates from the Source to the Target Volumes.

Consistency Standstill: A Consistency Standstill occurs when ReplicatorX is unable to accumulate sufficient data in the Log in order to advance a Consistency Timeline. A Consistency Standstill occurs very rarely, especially if ReplicatorX resources are configured properly.

In practical terms, a Consistency Standstill means that a Consistency Engine needs more entries in order to reach a Consistency Timeline, but there is not enough space available in the Log to accommodate the required data. This situation temporarily stalls replication activity.

The UD-Log Space quality-of-service definition enables you to optimally configure ReplicatorX resources, reducing the likelihood of a Consistency Standstill. You allocate Log Space per Session, and/or limit the Log Space potentially used by each Session (thereby minimizing the risk of Consistency Standstills in the other Sessions that share the same Log Space).

An inability of the RXRS to advance the Consistency Timeline situation occurs most often during Slim recording or a re-synchronization. For example, assume that a given Pair requires Bitmap re-synchronization following an extended network outage. All updates that occurred in the Source Volume during the outage will be reflected in the bits set. All of these changes will need to be stored in the Log before any data can be applied to its Target Volume. If the total amount of changed data exceeds the space available in the Log, a Consistency Standstill will occur.

To resolve a Consistency Standstill, the User must either:

- ◆ Increase the UD-Log capacity, allowing sufficient data to be stored in the UD-Log in order to achieve consistency; or
- ◆ Transform the affected Session to an Inconsistent state, temporarily enabling data to be applied to a Target Volume without considering its Consistency Timeline.

Even during a Consistency Standstill, a Target Volume always remains Consistent, unless it is transformed manually by the User to an Inconsistent state.

Consistency Standstill Detection

ReplicatorX supports consistency standstill detection on both the GUI and CLI level.

For information on consistency standstill detection, see the *ReplicatorX Administration Guide*.

For more detailed information on Consistency Standstill, see the *ReplicatorX Administration Guide*.

Recovering from a disaster

In case of a disaster or other system-wide failure at a site, ReplicatorX enables complete, scalable, and manageable recovery of your data.

Following a disaster that has affected a Pair, an existing Session or an entire Primary Site, you perform a Freeze on the affected ReplicatorX Objects. The Freeze causes an orderly and effective transition from active replication to a point-in-time Freeze, preserving data consistency on the designated Target Volumes.

After the Freeze is complete, you can access Target Volumes outside of ReplicatorX. A frozen Target Volume contains data that is consistent. The exact time of data consistency is the most current that ReplicatorX is able to achieve, and is dependent upon the data transfer status when the Freeze was invoked. A frozen Session contains a consistent image of all the Pairs that comprise its Consistency Collection.

ReplicatorX supports switching replication direction operation to enable business continuity and data recovery in the event of a temporary failure.

About the Switching Replication Direction Process: Switching replication direction is typically performed when one or more RXRCs become unavailable for a significant period of time, and it is necessary to use the Target Volumes at the recovery site in order to continue working routinely.

During switching replication direction, you create an RXRC on the recovery site in order to record User application write activities on the Target Volume. During this time, ReplicatorX uses Target Volumes as Source Volumes. Later, when the original site becomes available once again, you create an RXRS on the original site and synchronize all the necessary blocks from the recovery site back to the primary site, thus completing a switch of the replication direction.

To return the replication to its original direction, you perform the switching replication direction process a second time.

Throughout this process, ReplicatorX monitors all write activities and maintains the potential to bring all Pairs to consistency.

For more detailed information on disaster recovery, see the *ReplicatorX Administration Guide*.

Overview

This chapter describes how to plan, configure, and optimize the ReplicatorX system. It provides the following information:

- ◆ “[ReplicatorX requirements](#)” on page 31
- ◆ “[Resources and capacity](#)” on page 32

ReplicatorX provides unprecedented flexibility for data replication, and is capable of working transparently across different types of networks, platforms and applications. To support such flexibility, ReplicatorX requires that your replication network (RNET) be carefully planned and configured. Among the key issues to be considered when configuring ReplicatorX are:

- ◆ The needs of your business, including RPO and RTO, and activity growth projections
- ◆ Network bandwidth to be made available to the RNET
- ◆ The workload requirements, and other characteristics, of your business applications
- ◆ The data flow logic and rules between the ReplicatorX components

Each of these issues is complex, and requires a detailed understanding. Combining all of these issues in order to effectively address the problem of company-level replication brings with it new complexities. Such system-level complexities demand expertise and local experience. Therefore, in the following sections, discussion of these issues is limited solely to the context of configuring and optimizing ReplicatorX.

This chapter is intended to assist you, particularly during planning of the initial ReplicatorX installation, by providing details of how ReplicatorX components work together. It will also be helpful to your efforts in capacity planning and network optimization after ReplicatorX is already operational.

ReplicatorX requirements

ReplicatorX operates on multiple servers located at numerous sites, and interacts with various physical storage devices and networks. When optimized, ReplicatorX performs replication elegantly and efficiently, using the minimum required resources.

However, if not configured properly, ReplicatorX may perform poorly and may also cause measurable performance degradation.

ReplicatorX performance is most significantly affected by the following resources:

- ◆ Network bandwidth available, and its characteristics
- ◆ ReplicatorX Replication Client (RXRC) VRR capacity
- ◆ ReplicatorX Replication Server (RXRS) UD-Log capacity
- ◆ Physical storage configuration (disks, RAID, etc.)

ReplicatorX performance is further influenced by the following factors:

- ◆ RXRS Log Cache size
- ◆ Number and placement of RXRC and RXRS components in the configuration

There are additional factors to be considered when planning and configuring ReplicatorX. Some are related to designing an ideal configuration for your company, while others are related to the limitations imposed by ReplicatorX.

One of the strengths of ReplicatorX architecture is that it supports a highly flexible configuration, enabling modifications with minimum disruption to routine operations. However, there are some modifications that will require stopping replication activity, and so should naturally be avoided. Such disruption is often unnecessary and can be avoided with thorough planning prior to ReplicatorX installation.

Resources and capacity

This section details the resources and capacity necessary for planning your installation. It includes the following:

- ◆ [“Allotting sufficient memory for the VRR”](#) on page 33
- ◆ [“Ensuring sufficient bandwidth on the RNET”](#) on page 35
- ◆ [“Allotting optimum sizes for the Log Cache and MD-Log volume”](#) on page 41
- ◆ [“Allotting disk space for the UD-Log Volume”](#) on page 43
- ◆ [“Allotting disk space for Bitmap Volumes”](#) on page 44
- ◆ [“Configuring Target Volumes”](#) on page 46
- ◆ [“Determining the number of servers for RXRS installation”](#) on page 48

Note

Many of the issues in this section are referenced during ReplicatorX site preparations prior to installation. For more information, see the *ReplicatorX Installation and Upgrade Guide*.

Allotting sufficient memory for the VRR: Each RXRC maintains a RAM buffer in which it records all Write activities addressed to its Source Volumes. This buffer is called a Volatile Recording Repository (VRR). After the content of a recorded Write is passed to the RXRS, this data is deleted from the VRR.

In cases where there is limited RAM available for ReplicatorX, the VRR size should be as small as possible. However, too small a VRR size can result in persistent buffer overflow, and seriously impair replication performance. This trade-off makes setting proper VRR size critical.

Ideal VRR operation

Theoretically, during ideal VRR operation, the running bandwidth of all application Writes addressed at Source Volumes remains smaller than the available RNET bandwidth, there are no RNET, RXRS or RXRC availability problems, no Log overflows or other resource problems. In such a scenario, every Write is recorded to the VRR and immediately propagated to the RXRS.

The recommended minimum VRR size is 16 MB. In the theoretically ideal scenario described above, allotting the minimum VRR size will avoid buffer overflow without impairing performance. It is possible to allot a smaller VRR size if a severe physical limitation exists (for example, working with a legacy server with limited physical memory).

Note

The current software version supports a maximum VRR size of 2 GB. On Windows 32-bit, it supports a maximum VRR size of 128 MB.

Routine VRR operation

Ideal operation aside, in practical operation, whenever the stream of inputs to the VRR (that is, application Writes at Source Volumes) is larger than the stream of outputs (that is, recorded Writes that were passed to the RXRS), the VRR utilization grows, resulting in progressively less free space to record additional Writes. The causes for this scenario can include:

- ◆ Peaks in application Write bandwidth – that is, peaks in which bandwidth is higher than the RNET bandwidth.
- ◆ RNET unavailability (planned or unplanned), decreased available RNET bandwidth, or increased RNET latency.
- ◆ Resource problems in the corresponding RXRS (for example, a Consistency Standstill), or unavailability of the RXRS.

The VRR mechanism is designed to continue to operate routinely under such conditions. After the VRR becomes full, the RXRC applies a patented technology called Slim recording: the RXRC frees some (or all) of the recorded

Writes from the VRR, replacing each freed Write with Metadata that describes the Write. Each such Write is called a Slim VRR Entry (as opposed to a Full VRR Entry that contains the Write's actual data). Before the RXRC passes a Slim Write to the RXRS, it reads its contents directly from the Source Volume and propagates the entry to the RXRS. ReplicatorX is capable of handling potential inconsistency problems that may occur, since the data read subsequently from the Source Volume will not always be the data originally written by the Slim Write operation. In any event, during Slim recording, ReplicatorX ensures that the Target Volume remains consistent at all times.

The use of Slim recording increases – by orders of magnitude – the number of VRR entries storable on the RXRC without causing VRR overflow. For example, a VRR size that can accommodate a 30 second RNET outage with Full recording (with average Write sizes of 32 KB), can accommodate an approximately 40-minute RNET outage with Slim recording, without overflow.

While the use of Slim VRR entries dramatically improves performance and reduces the need for Bitmap re-synchronization, Slim VRR entries also exact a price, compared to Full VRR entries. Specifically, the Slim recording mechanism has two disadvantages:

- ◆ Propagation of a Slim VRR Entry requires an additional Read operation that can potentially degrade replication performance. Since this extra Read potentially competes with application I/O activity on a Source Volume, it might also impact application performance. Normally, however, if the underlying storage infrastructure has sufficient bandwidth to deal with these Reads, the impact may be negligible.
- ◆ When receiving a stream of Slim VRR Entries, the RXRS may sometimes need to propagate the Consistency Timeline over large time gaps. (This occurs because ReplicatorX algorithms must recognize and compensate for potential inconsistency.) In extreme cases, this scenario could even result in a Consistency Standstill.

VRR overflow

When the VRR has no free space even for Slim recording, a VRR overflow occurs. As a result, the VRR is erased and Source Volume Writes are recorded only to the Bitmap. When this occurs, ReplicatorX immediately invokes a re-synchronization activity based on the Bitmap recording.

Of course, while ReplicatorX is designed to handle such situations, a VRR overflow is not a desirable state and should be avoided. VRR overflow presents the following disadvantages:

- ◆ Depending on the Bitmap Granularity setting, more data than was actually updated may need to be propagated to the RXRS.

- ◆ The RXRS cannot advance the Consistency Timeline to a time that is between the time of buffer overflow and the time in which the re-synchronization completes. If the RXRS logs are unable to hold the amount of data that needs to be replicated during a re-synchronization, then the corresponding Session(s) may enter a state of Consistency Standstill.

Note

When re-synchronization occurs, it does not affect consistency on the Secondary Site. For more information, see the *ReplicatorX Administration Guide*.

Summary of factors for VRR size

The default VRR size assigned by ReplicatorX can be adjusted to better reflect your requirements. When setting the VRR size, the following should be taken into consideration:

- ◆ **Characteristic RNET Operation.** Momentary RNET outages and momentary application Write peaks can cause the VRR to use Slim recording. If it becomes necessary to reduce instances of Slim recording, increase the VRR size, the RNET bandwidth, or both. For more information on this issue, see “[Workload influences on RPO and VRR performance](#)” on page 36.
- ◆ **RXRC Physical Memory.** Consider the physical memory size on the RXRC. Increasing memory to the VRR reduces the memory available for other processes running on the component.

Changing the VRR size

To change the physical memory being allotted to the VRR on an RXRC, you modify the `vrr_SharedSizeInLbs` parameter on that component. (For change procedure details, see the *ReplicatorX Administration Guide*.)

Ensuring sufficient bandwidth on the RNET: Frequently – especially with replication over long distances – the cost of RNET bandwidth is a major factor in the total cost of data replication. While in synchronous replication the RNET bandwidth should typically be equal to the peak bandwidth of application Writes, the RNET bandwidth requirements of asynchronous replication are usually more relaxed, and RNET bandwidth equal to the average bandwidth of application Writes will often prove sufficient.

A higher RNET bandwidth:

- ◆ Optimizes the Recovery Point Objective (RPO) in periods where application Write bandwidth is higher than the average
- ◆ Improves the ability to transparently withstand RNET unavailability, Secondary Site failures, Primary Site server failures, etc.

- ◆ Ensures that sufficient bandwidth remains available for ReplicatorX, despite dynamic, sudden periods of RNET bandwidth consumption by other applications
- ◆ Permits reductions in size of other resources, such as the VRR and the UD-Log
- ◆ Speeds up initial synchronization (when using the RNET)

Optimizing RPO

For theoretically optimal RPO, the RNET bandwidth should equal the peak application Write bandwidth. In this way, the VRR does not grow, so Writes are immediately propagated to the Secondary Site. Assuming no failures, such a situation might lower RPO to as low as tens of milliseconds, constantly. The VRR and UD-Log sizes could remain relatively small. (Enlarged VRR and Log sizes are still recommended, in order to withstand failures without degrading replication activity.)

Excellent RPO can also be achieved using a more relaxed approach: the RNET bandwidth should be sufficient to prevent the VRR from entering Slim recording mode – even in cases where application Write bandwidth is high. Even if there is occasionally a period of a few seconds of Slim recording, this won't significantly impact overall performance. Depending on the exact configuration, this might result in an average RPO of a few seconds, or possibly even several hundreds of milliseconds.

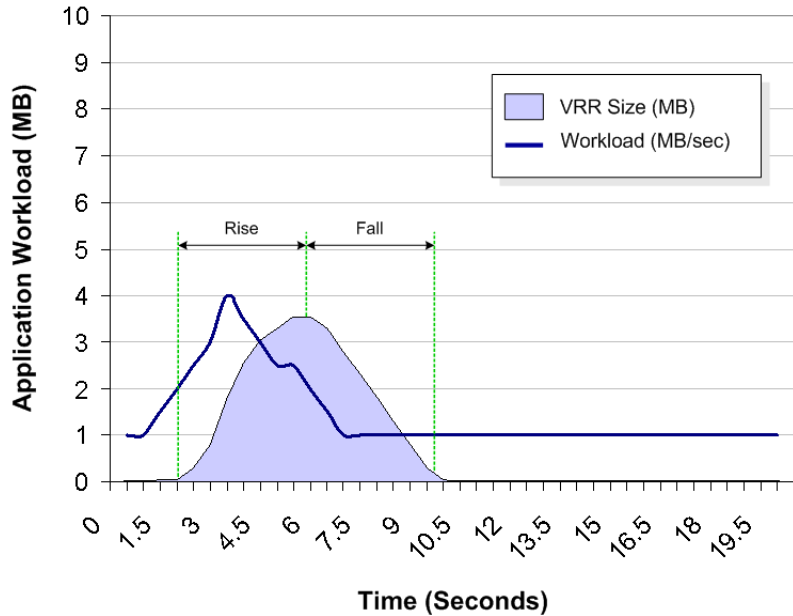
Workload influences on RPO and VRR performance

Assume an RNET connection between Primary and Secondary Sites with a bandwidth of 2 MB/sec, and a configuration comprised of four RXRCs, each with a VRR size of 128 MB. For simplicity, assume that the application workload is distributed uniformly across all RXRCs at all times. Such a configuration could sustain the following application Write throughput, without the need for Slim recording:

Continuous Peak	Duration	Formula
3 MB/sec	8.5 minutes	$512 \text{ MB} / (3 \text{ MB/sec} - 2 \text{ MB/sec})$
4 MB/sec	4 minutes	$512 \text{ MB} / (4 \text{ MB/sec} - 2 \text{ MB/sec})$
6 MB/sec	2 minutes	$512 \text{ MB} / (6 \text{ MB/sec} - 2 \text{ MB/sec})$

In addition, this configuration could easily sustain momentary peaks of tens of megabytes per second.

The following figure shows the effects on RPO when the workload is 1 MB/sec, and the RNET bandwidth is 2 MB/sec. There is a momentary workload peak within a period of 5 seconds that climbs up to 4 MB/sec. Looking at the VRR size in the resulting graph reveals a rise period (from Time 2 to 6 seconds) and a fall period (from Time 6 to 10 seconds).

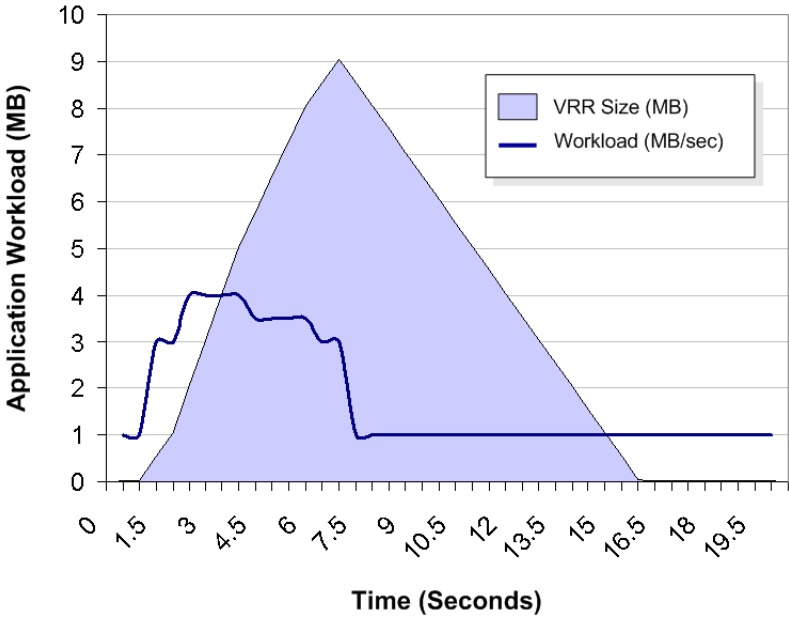


During the rise period, VRR utilization grows, and the RPO grows. At the highest point in the peak (at the end of the rise), the RPO reaches approximately 2 seconds. During the fall period, VRR utilization decreases, as does the RPO. By the end of the fall period, RPO approaches 0.

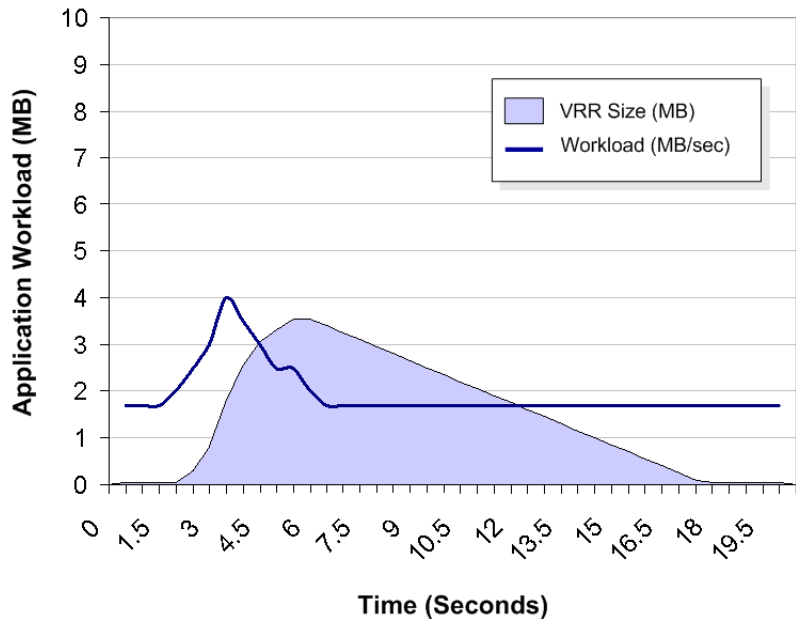
The rise period begins as soon as the workload is higher than the RNET bandwidth. However, the fall period continues even after the workload goes below the RNET bandwidth. The length of the fall period depends on the following factors:

- ◆ The amount of data accumulated in the VRR during the peak
- ◆ The delta between the RNET bandwidth (2 MB/sec) and the workload bandwidth following the peak (1 MB/sec); in this case, a value of 1 MB/sec

The influence of these two factors is more clearly demonstrated in the following figures.



With an extended duration in peak workload, more data must be accommodated, causing the VRR to accumulate more data, resulting in a longer duration of time both for the fall to occur, and for the RPO to approach 0.



During a peak period, a reduced delta between the RNET and workload bandwidth also produces a longer period of time for the fall to occur, but without causing as much data to accumulate in the VRR. The affect of a reduced delta can have dramatic effects on the RPO. If the delta becomes too small, then the period of time until it approaches 0 will greatly increase. In some scenarios, this may result in a constant non-zero RPO, a potentially undesirable state.

Note

As long as the VRR does not enter a Slim recording state, the Consistency Timeline will progress smoothly during both the rise and fall periods.

Note

Even when (during a peak period) Slim recording occurs, the length of the rise and fall periods on the VRR does not change. Because during Slim recording the Consistency Timeline will progress in jumps, the RPO values will remain at their peaks longer. In such a scenario, more UD-Log space should be available to accommodate more entries until the Consistency Timeline progresses. If the UD-Log becomes full during one of the jumps, then a Consistency Standstill might occur.

Handling synchronization requirements

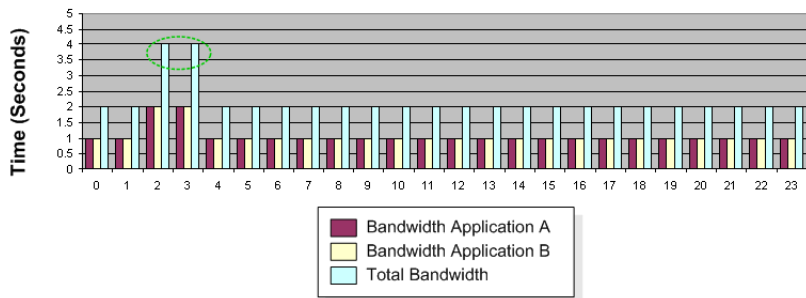
During synchronization – especially during the initial synchronization process – the combination of synchronization activity with new Write activity at the Primary Site can create a peak in bandwidth demand which could potentially reduce the bandwidth available for real-time, ongoing Write propagation. To prevent this, ReplicatorX algorithms are designed to give absolute priority to new I/O activity on a Source Volume, reducing the RNET bandwidth given to synchronization activity bandwidth peaks. As soon as bandwidth allows, ReplicatorX allots increased bandwidth to synchronization.

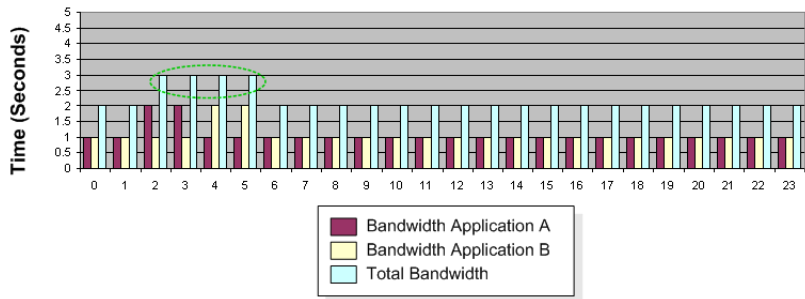
For example, given a RNET bandwidth of 4 MB/sec, and an application workload of 1 MB/sec, synchronization will be allotted bandwidth of 3 MB/sec. If there is a peak in application workload at this time (momentary or continuous) of 3.5 MB/sec, synchronization will be allotted bandwidth of only 0.5 MB/sec for the duration of the peak period.

Handling scheduled jobs

Consideration should also be made for handling scheduled periods of time where application workload is markedly higher for a long period (for example, running daily batch jobs during low usage periods at night). During such periods, significantly more bandwidth may be required to achieve the same quality of RPO that is achieved at any average moment of the day.

One practical approach to this problem is to maintain higher RNET bandwidth at all hours and use the free bandwidth for other applications at different times of the day. Another solution is to spread execution of the batch jobs in order to decrease the RNET bandwidth requirements, as shown in the following figures.





Summary of factors for RNET bandwidth

It is easier to determine the correct RNET bandwidth for a given ReplicatorX configuration once the application workload is carefully studied and after the Company RPO requirements are well defined.

Among key considerations for allocating RNET bandwidth are:

- ◆ Nominal RPO requirements
- ◆ Characteristic application workloads
- ◆ Demands of synchronization process
- ◆ Ability to schedule high bandwidth activities

Allotting optimum sizes for the Log Cache and MD-Log volume: The Log uses several components to perform its operations:

- ◆ One or more UD-Log Volumes (see “[UD-Log size](#)” on page 44)
- ◆ A large portion of RXRS memory, called the Log Cache
- ◆ One or more MD-Log Volumes (see “[MD-Log Volumes size and disk type](#)” on page 42)

The operation of both the Log Cache and the MD-Log Volume is interconnected, so their requirements are presented here together.

When storing a Write that has yet to be applied to a Target Volume, the Log actually handles two entities: a Customer Data entry (the contents of the Write) and a corresponding Metadata entry (consisting of information describing the Write). To enhance performance, the Log Cache is used to store both data types.

In order to allow recovery from an RXRS restart, all Data is hardened to a UD-Log Volume, and all Metadata is hardened to an MD-Log Volume.

To minimize the number of required Reads from a UD-Log Volume and an MD-Log Volume, any available Log Cache space is used to store Customer Data and Metadata. That is, when applying a Write to the Target Volume, if the corresponding Customer Data and Metadata is present in the Log Cache, there is no need to read it from the UD-Log Volume.

Log Cache Size

The size of the Log Cache is configurable. When setting the Log Cache size, the following should be taken into consideration:

- ◆ **Available Physical Memory.** Although the Log Cache is a virtual memory resource, its size should not exceed the available physical memory of the RXRS.

The correct available physical memory may be calculated as follows:

$$\begin{aligned} \text{Max. Cache Size} &= \text{Total Physical Memory} \\ &\quad - \text{Physical Memory Used by OS} \\ &\quad - 100 \text{ MB} \end{aligned}$$

Where **100 MB** is the amount of memory used by the RXRS software for other purposes.

- ◆ **Expected Number of Pairs.** There must be at least 1 GB cache for each 128 pairs that are not clustered. A pair that is configured on a clustered RXRC needs 8MB for each node of the cluster.
- ◆ **32-Bit Address Space Constraint.** The user mode address space of 32-bit operating systems is limited to 2 GB. Therefore, the default Log Cache size set by ReplicatorX for this operating system is 1.5 GB. It is strongly recommended that the Log Cache size not be modified for 32-bit operating systems.

To change the size of the Log Cache on an RXRS, you modify the **LogCacheInLb** parameter on that component. (For change procedure details, see the *ReplicatorX Administration Guide*.) Changes to the Log Cache size will take effect only after restarting the RXRS daemon/service.

MD-Log Volumes size and disk type

When determining the optimum size and disk type for the MD-Log Volumes, the following should be considered:

- ◆ **Separate Disk for MD-Log.** For optimum performance, an MD-Log Volume should be created on a physical disk that is separate from those used for Target Volumes or the UD-Log Volume. Sharing of disk resources may cause serious performance degradation.

- ◆ **General Size Calculation.** The total size of the MD-Log Volumes determines the maximum number of entries allowed in the MD-Log Volumes. Each Metadata entry occupies 64 bytes. Therefore, an MD-Log Volume of 500 MB will store approximately 8 million Metadata entries. The total MD-Log Volumes size requirement may be calculated as follows:

Total MD_Log Volumes Size = Expected Max # of Metadata Entries × **64 Bytes**

Where: Each Metadata entry occupies 64 bytes.

Allotting disk space for the UD-Log Volume: The UD-Log stores data that was passed to the RXRS from its associated RXRCs, but has not yet been applied to Target Volumes.

A data entry is not applied – that is, it remains in the UD-Log – for any of the following reasons:

- ◆ Other data is currently being applied to the Target Volume, so that the stored data entry must wait until it can be handled.
- ◆ The Target Volume is frozen. Until it is unfrozen, the stored data entry must wait.
- ◆ While working in a Consistent state, if other data must arrive in order to advance the Consistency Timeline, any post-Timeline data entries must wait.

When the UD-Log becomes full, additional data is not propagated from the RXRC. This means that the RNET is not utilized, and the VRR may approach its capacity. In severe cases, the VRR may enter Slim recording mode and may even overflow. Additionally, if working in a Consistent state when the UD-Log Volume reaches its capacity, ReplicatorX may be unable to advance the Consistency Timeline. This situation will result in a Consistency Standstill in the following cases:

- ◆ If the VRR is using Slim recording, the RXRS may be awaiting Slim entries from a later time, before the Consistency Timeline can be propagated. When there is no space left in the UD-Log, a Consistency Standstill will result.
- ◆ If the RXRC is performing a re-synchronization based on a Bitmap, the RXRS may be awaiting all required data to arrive before the Consistency Timeline can be propagated. When there is no space left in the UD-Log, a Consistency Standstill will result.

For these reasons, it is recommended that UD-Log Volumes be allotted sufficient resources in order to avoid reaching their capacity.

UD-Log size

The UD-Log must be less than 2 TB. However, the nominal capacity allotted for the sum of UD-Log Volumes should be large enough to store:

- ◆ All incoming data for the duration of any period when its associated Target Volume(s) is frozen.
- ◆ All updates that are required from the associated Source Volume(s) in order to advance the Consistency Timeline.

UD-Log disk type

When determining the UD-Log Volume disk type, the following should be taken into consideration:

- ◆ For best performance, UD-Log Volumes must reside on physical disks separate from the ones used for the MD-Log Volume or any Target Volumes. Sharing these resources may cause serious performance degradation.
- ◆ ReplicatorX writes to UD-Log Volumes in a way that maximizes the amount of sequential I/O. It is recommended that the disks work optimally for sequential writes.
- ◆ The UD-Log uses a cache on the RXRS (see “[Allotting optimum sizes for the Log Cache and MD-Log volume](#)” on page 41) intended to speed up the apply process on the Target Volume. As long as this cache can accommodate all the data that is being propagated, the UD-Log Volume need not be read. However, to improve performance during periods of stress, it is recommended that the UD-Log storage device be capable of providing sufficient bandwidth to write all propagated data and then subsequently read all propagated data.
- ◆ Regardless of the RAID method used to define the UD-Log Volume, it should provide the highest possible speed, so as not to degrade performance.

Allotting disk space for Bitmap Volumes: A Bitmap Volume is used to support persistent Bitmap-level hardening of Source Volume I/O activity on its RXRC.

Each bit in a Bitmap maps to a pre-defined extent in its designated Source Volume. When a Write occurs within that extent on the Source, its corresponding bit is set in the Bitmap. This information is used during a re-synchronization.

Multiple Bitmap Volumes can be assigned to each RXRC. Each Bitmap Volume can hold multiple Bitmaps. The User determines which Source Volumes will be hardened with a Bitmap, and for each defined Pair that uses hardening, ReplicatorX automatically allocates a portion of its corresponding Bitmap, recording there all changes made to the designated Source.

Note

If you have limited disk space, you may consider adding a Bitmap File. For more information, see the *ReplicatorX Administration Guide*.

Bitmap Volume size

For each RXRC, the Bitmap Volume size requirement may be calculated as follows:

$$\text{Recommended Size} = \frac{\left(\frac{\text{Total Source Volume Disk Capacity (bytes)}^*}{\text{Bitmap Granularity (bytes)}} \right)}{2}$$

* When using a volume growing factor, the sum of the size of all volumes multiplied by the growing factor should be used. When not using a volume growing factor, the sum of the size of all volumes should be used. When using a maximum volume size for the volume growing option, the sum of the maximum size of all volumes should be used. For more information, see the *ReplicatorX Reference Guide*.

In the example below, the total disk capacity is 1 TB, the growing factor used is 4. The Bitmap granularity is 128 LBs or 64 KB.

$$\frac{\left(\frac{1 \text{ TB} \times 4}{64 \text{ KB}} \right)}{2} = 32\text{MB}$$

Bitmap Volume disk type

When determining the Bitmap Volume disk type, the following should be taken into consideration:

- ◆ **Separate Disks for Bitmap Volumes.** For optimum performance, a Bitmap Volume should be created on a physical disk that is separate from those used for Source Volumes. Sharing of these disk resources can cause serious performance degradation.
- ◆ **Bitmap-to-Pair Relation.** When handling multiple Pairs, each Pair may be assigned either to the same Bitmap Volume, or to a different Bitmap Volume residing on a different physical disk. Using different physical disks may sometimes improve performance.

- ◆ **Bitmap-to-Source Relation.** If there are many Source Volumes on an RXRC, increase the number of physical disks used for Bitmap Volumes to improve performance.

Bitmap Volume redundancy

Because it stores information that is essential to replication in the event of an RXRC failure, it is recommended that Bitmap Volumes be maintained using RAID, mirroring, or similar safe disk architecture.

Configuring Target Volumes: Each Target Volume must be directly accessible by its corresponding RXRS.

Formatting Target Volumes

Once a Target Volume is defined in ReplicatorX, it must not be altered by any application other than ReplicatorX. Alteration by another application will result in data integrity problems, making the Target Volume unreadable to ReplicatorX.

- ◆ On Unix platforms, a Target Volume disk may be formatted either as raw or as a file system. However, after it is assigned, a file system Volume must not be mounted. If it is mounted, the file system might spontaneously alter the Target Volume, thus resulting in data integrity problems.
- ◆ On Windows platforms, a Target Volume disk may be formatted either as raw volume or as a file system. ReplicatorX will try to unmount the file system and change sector 0 of the volume to make it a raw volume during the Add Pair operation. However, if this operation fails, you must manually ensure that the Target Volume is designated as a raw Volume before activating any associated Sessions.

Accessibility of Target Volumes

Each Target Volume should be accessible by Secondary Site servers following a disaster, or during off-site processing.

- ◆ If the storage environment allows, enable accessibility by both the RXRS and other pertinent Secondary Site servers. However, it is important to prevent other servers from altering Target Volume contents in any way during replication activity.
- ◆ When required, the RXRS may run the applications needed to access a Target Volume. However, it is not recommended that other applications be run simultaneously when the RXRS service is running on the same server.
- ◆ It is also possible to physically disconnect Target disks from one server and reconnect them to a different server, in order to make them accessible.

Matching Source and Target Volume sizes

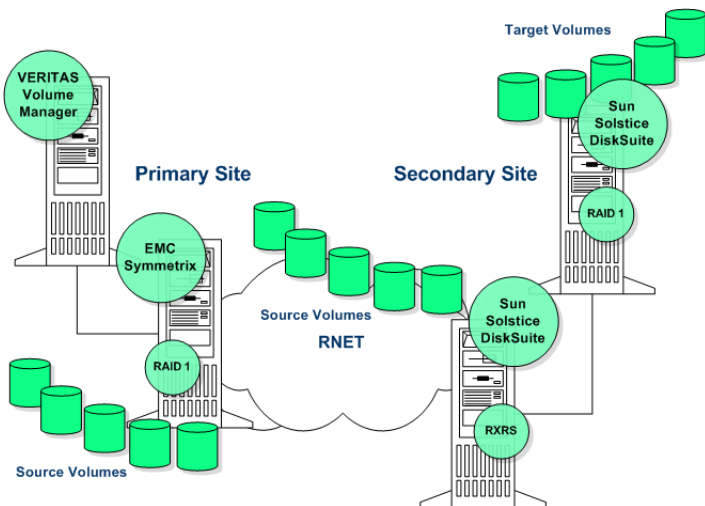
It is best practice to use identical Volume sizes for Source Volumes and their corresponding Target Volumes. In this context, identical means that the two Volumes should have the same number of LBs.

In any event, the Target must not be smaller than the Source. While ReplicatorX supports replication from a smaller Source to a larger Target, not all User applications do. This presents a potential hazard, should these applications be required to read from a nonmatching Target Volume during a recovery operation.

Volume management related issues

ReplicatorX supports the use of mixed Volume management software packages at different sites, and on different servers. However, consideration must be made for how Target Volumes will be accessed during off-site processing or following a disaster.

- ◆ At the Secondary Site, the same (or a compatible) Volume manager must be used to define a Target Volume, both on the server used to access the Target and on the RXRS.
- ◆ Between Primary and Secondary Sites, different Volume managers can be used to define both Source and Target Volumes. For example, as shown in the following figure, Source Volumes at a Primary Site can be defined using the VERITAS Volume Manager, while Target Volumes at the Secondary Site can be defined using Sun Solstice DiskSuite. ReplicatorX operates transparently with both Volume management packages.



Using mirroring for Target Volumes

Defining a Target Volume as mirrored (using techniques such as RAID1, EMC TimeFinder, and the like) can provide the following benefits:

- ◆ **Redundancy.** A mirrored Target Volume becomes a fault-tolerant component, significantly reducing the potential for Target Volume errors.
- ◆ **Enables Altering the Target.** As a rule, following a Freeze, if the Target Volume subsequently needs to be unfrozen, its contents must not be altered. However, if affected Target Volume is mirrored, this limitation can be avoided simply by using the following technique:
 - a. Wait until the Freeze is completed and break the mirror relation.
 - b. Invoke an Unfreeze. ReplicatorX continues replication activity with only one of the mirrored copies, while enabling the second copy to be accessed and modified freely.
 - c. After any offsite processing is completed, rebuild the mirror relation with the affected Target Volume.
- ◆ **Preserves a Consistent Image.** There are scenarios where it is necessary for ReplicatorX to replicate using Inconsistent mode (see the *ReplicatorX Administration Guide*). Typically, this occurs in order to resolve a Consistency Standstill. Mirroring allows a Consistent image of the Target Volume to be preserved, even during the period in which ReplicatorX replicates inconsistently:
 - a. Prior to transforming the Session to Inconsistent mode, break the mirror relation. In this way, one of the Target Volume copies remains untouched and Consistent.
 - b. Resolve the problem that required transformation to inconsistency and toggle the affected Session back to Consistent mode.
 - c. When ReplicatorX resumes replication in Consistent mode, rebuild the mirror relation with the affected Target Volume.

Determining the number of servers for RXRS installation: The minimum ReplicatorX configuration requires a single RXRS at the Secondary Site. However, configurations with multiple operating systems or mixed Volume management packages may require multiple RXRS components. In addition, multiple RXRS components may be required in order to support scalability or security needs.

Volume manager/Target disk layout compatibility

As described in “[Configuring Target Volumes](#)” on page 46, the server that accesses the Target Volume following a disaster (or during offsite processing) must have a Volume manager that is compatible with the one used for that Target in the RXRS. In this context, the native OS drivers that manage the disk must also be considered as Volume managers. This unavoidable requirement can make it impossible to access all Target Volumes from a single RXRS.

For example, consider a configuration that includes both Windows 2000 Servers using basic Volumes (the native Volume manager for Windows NT) and Solaris servers using Solaris DiskSuite Volumes. In such a case, an RXRS would need to maintain both basic Windows NT Target Volumes and Solaris DiskSuite Target Volumes.

Since there is no Windows NT driver that maintains DiskSuite Volumes, and no Solaris driver that maintains Windows NT Volumes, the configuration must use two RXRS components: one running under Windows 2000 and another running under Solaris.

Additional examples:

- ◆ When running different versions of Solaris (that are supported by ReplicatorX) that use the same Volume manager, a single RXRS could be used, running on either Solaris version.
- ◆ In a configuration including several Solaris servers, and using two different Volume managers which cannot co-exist on the same server, the compatibility problem between the two Volume managers would require the use of two RXRS components, with each one running a different Volume manager.

Consult your Volume manager documentation or vendor for more information about compatibility between different product versions and operating systems.

DAS device support

In physical configurations where Target Volumes are directly attached to the Secondary Site servers that are used following disaster (or during off-site processing), and these Target Volumes cannot be physically attached by any other server, it is possible to run an RXRS on each of these Secondary Site servers. In such a scenario, each RXRS maintains only those Sessions associated with the local Target Volumes. Note however, that cross-server consistency could not be achieved using such a configuration.

Isolation for security purposes

Especially when the Secondary Site is used for multiple companies (each maintaining different Primary Sites), multiple RXRS components may be required in order to achieve isolation for security reasons. In these scenarios, each company maintains its own separate RXRS components, connected to that company's Primary Site.

Note

If the security restrictions are less severe, Users with restricted site access can be used. For more information, see the *ReplicatorX Administration Guide*.

Isolation for resource control

As with isolation for security purposes, isolation also provides additional control over resource usage. This can also be beneficial when maintaining a configuration that includes multiple companies or divisions. Using multiple RXRS components, it is possible to separate key resources such as:

- ◆ UD-Log resources
- ◆ RXRS computation power
- ◆ I/O bandwidth

In each case, the replication activity of one company would have no impact on the activity of another company.

Scalability

In configurations designed to support a high number of connections and heavy workloads, multiple RXRS components can provide scalability for current demands as well as future growth. Multiple RXRS components should be considered in the following cases:

- ◆ A single RXRS is not capable of handling all the bandwidth coming from the Primary Site.
- ◆ A large number of RXRCs are used, and a single RXRS cannot be strengthened or expanded to handle all connections efficiently.

Glossary

24/7	24 hours, 7 days a week
API	Application Programming Interface
Bitmap	A fault-tolerant resource residing on the RXRC. The following invariant is always correct: if an extent in the Source Volume has been modified, its corresponding bit is set in the Bitmap. Each bit in a Bitmap maps to a pre-defined extent in its designated Source Volume.
Bitmap Granularity	The number of LBs of data that are represented by a single bit in the Bitmap. The higher the value set for granularity, the greater the amount of data represented per bit.
Clock Group	A ReplicatorX logical object, comprised of a single Clock Master and one or more Slave clocks. Clock Groups are used to implement clock synchronization for Sessions that contain multiple RXRCs.
Clock Loss	The loss of synchronization between a Master and Slave clock pair. Typically, this is a transient condition.
Clock Master	A shared clock, residing on a single designated RXRC, used to distribute the current time to other RXRCs, designated as Slave clocks, for purposes of synchronization.
Consistency Collection	A logical subset of a Session. At any given moment, a Consistency Collection is comprised of those Pairs for which the combination of Customer Data on their Target Volumes, along with the information in their associated logs, is sufficient to bring the Target Volumes to a Consistent state.

Consistency Engine	A proprietary mechanism used by the RXRS to calculate a timeline for Consistency Objects.
Consistency Level	A Pair property that describes its state in terms of consistency. ReplicatorX supports three Consistency Levels: Inconsistent, Self-Consistent, and Consistency Collection.
Consistency Object	For purposes of replication, a Consistency Collection or a Pair that is not in Consistency Collection
Consistency Standstill	A state where ReplicatorX is unable to transfer sufficient data in the Log sites in order to advance the timeline for a Consistency Object.
Consistency Timeline	The most recent time that the data in a Consistency Object is Consistent.
Control Object	An entity, relation or resource maintained by the ReplicatorX RXMS. Once assigned, a Control Object is accessible by any ReplicatorX Client.
Control Object ID	An object property that is used to uniquely identify each assigned Control Object. Each Control Object ID is comprised of an index, tag and name.
CLI	Command Line Interface
CPU	Central Processing Unit
Customer Data	The actual content of a Write addressed to a Source Volume. ReplicatorX uses Customer Data along with Metadata to perform replication.
UD-Log Volume	<i>See Log</i>

DAS	Direct Attached Storage
Freeze	<p>A request that ReplicatorX perform a transition from normal replication activity to create a Consistent image of the data on Target Volumes. Data is no longer applied to Target Volumes.</p> <p>A Freeze is a User-initiated activity, invoked in order to perform routine offsite processing or following a disaster.</p>
Full VRR Entry	An entry in the Volatile Recording Repository that is created during normal operation, containing the full contents of a Write addressed to a Source Volume along with Write I/O-related information (called Metadata).
GB	Gigabyte
Gb	Gigabit
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
IT	Information Technology
KB	Kilobyte
Kb	Kilobit

LAN	Local Area Network
LB	Logical Block. Typically, each logical block occupies 512 bytes.
LBA	Logical Block Address
Log	<p>A Log resides on each RXRS to accumulate incoming data transferred from its associated RXRCs, but not yet been applied to the appropriate Target Volumes. A Log is comprised of three key resources:</p> <p>Log Cache. A portion of host memory used to cache both Customer Data and Metadata.</p> <p>UD-Log Volume. The UD-Log Volume is used to store Customer Data.</p> <p>MD-Log Volume. The MD-Log Volume is used to store Metadata.</p>
Log Cache	<i>See Log</i>
MB	Megabyte
Mb	Megabit
Metadata	<p>Information that describes the I/O-related details of a Write to a Source Volume. Metadata is comprised of Source Volume information, the modified Volume extent, timestamps, and other information. It is used to achieve and maintain Consistent replication between Source and Target Volumes.</p> <p>ReplicatorX uses Metadata along with Customer Data to perform replication.</p>
MD-Log	<i>See Log</i>
NIC	Network Interface Card

Object Property	A dynamic element, typically associated with a specific Control Object. An object property is used to manipulate its Control Object or to reflect its state at any given time.
Pair	A Pair represents a replication relationship between a Source Volume and a Target Volume.
Primary Site	A local production site containing Volumes to be replicated by ReplicatorX.
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
Recovery Point Objective	The maximum age of the data that the company must be able to recover in the event of a disaster. In other words, how old can your data get before it is no longer useful? The older the data, the less demanding your recovery point objective. This parameter is considered together with the Recovery Time Objective, when planning the disaster recovery implementation.
Recovery Time Objective	The length of time required to recover from a disaster. In other words, how long can the company afford to be without access to its data. This parameter is considered together with the Recovery Point Objective, when planning the disaster recovery implementation
Replication Network	The collection of TCP/IP networks used to connect all RXRC and RXRS components. Typically, this network is comprised of Primary Site LANs, the Secondary Site LAN, and the interconnecting WAN.
ReplicatorX Management Server	A ReplicatorX software module that runs as a service, and resides on a single, standard server located at the Secondary Site. Its main functions are: - Enable remote User interaction with ReplicatorX software.

- Maintain all required configuration information about the Primary and the Secondary Sites.

- Interacts with Replication Clients and Replication Servers to configure, control and monitor all ReplicatorX activities.

ReplicatorX Remote Agent

A ReplicatorX software module that runs as a daemon/service and resides on each of the servers on which an RXRC, RXRS, or RXMS is installed. Only one instance of the RXRA is installed on each server, even if more than one ReplicatorX module is installed on that server. The RXRA interacts with the RXMS and RXMC and executes tasks on their behalf.

ReplicatorX Replication Client

A ReplicatorX module that operates on a customer server that records, stores and supplies the Metadata for its associated Source Volumes, and is responsible (together with the pertinent ReplicatorX Replication Server) for transferring both the recorded Metadata and Customer Data from the Primary Site to the Secondary Site.

ReplicatorX Replication Server

A ReplicatorX module that operates on dedicated servers at the Secondary Site, and is responsible for accumulating recorded data from the Primary Site; for ensuring that the Target can be brought to a Consistent state; for moving replicated data from Logs to Targets; and for managing resources, communications, and other aspects of a configuration.

ReplicatorX SNMP Agent

A ReplicatorX service that provides both the interoperability between ReplicatorX and Windows SNMP services, translating RXMS alerts to standard SNMP alerts, and the functionality required for monitoring standard SNMP alerts, and manual control over SNMP operation from the Windows Services manager. For information about SNMP alerts, see the *ReplicatorX Reference Guide*.

RNET

See Replication Network

RPO

See Recovery Point Objective

RTO	<i>See</i> Recovery Time Objective
RXMS	<i>See</i> ReplicatorX Management Server
RXRA	<i>See</i> ReplicatorX Remote Agent
RXRC	<i>See</i> ReplicatorX Replication Client
RXRS	<i>See</i> ReplicatorX Replication Server
RXSA	<i>See</i> ReplicatorX SNMP Agent
SAN	Storage Area Network
SCSI	Small Computer System Interface
Secondary Site	A remote backup site containing Volumes that store data replicated by ReplicatorX, accessible in the event of a disaster or for purposes of offsite processing.
Session	The primary ReplicatorX replication set. Generally, a Session is comprised of Pairs whose data are mutually dependent. For example, you may define a Session to include all the Volumes that are used by a single database. Each Pair in the Session would contain a portion of the data needed by the database, and ReplicatorX would be able to maintain this database in a Consistent state at all times.
Slave Clock	A clock residing on one or more RXRCs, that receives the current time distributed by the Clock Master, for purposes of synchronization.

Slim VRR Entry	An entry in the Volatile Recording Repository that is created during slim recording, containing only Metadata that describes a Write addressed to a Source Volume.
SMIT	System Management Interface Tool (AIX Platform)
SNMP	Simple Network Management Protocol
Source Volume	A Volume located at a Primary Site containing data to be replicated by ReplicatorX.
Target Volume	A Volume located at a Secondary Site intended to store the data replicated by ReplicatorX.
TB	Terabyte
TCP	Transmission Control Protocol
Unfreeze	A request that ReplicatorX resume normal replication activity following a Freeze. Data is once again applied to Target Volumes.
UniVol	A UniVol (or UniVolume) represents a logical connection between a specified RXRC or RXRS object and a Volume object. It is used to support clustered environments, where shared access to Volumes is enabled.
UTC	Coordinated Universal Time
VM	Virtual Memory

Volatile Recording Repository

A ReplicatorX resource on the RXRC, used to record each intercepted Write addressed to a Source Volume. The VRR stores each Write until it can be transferred to the appropriate Secondary Site RXRS.

VRR

See Volatile Recording Repository

WAN

Wide Area Network

