

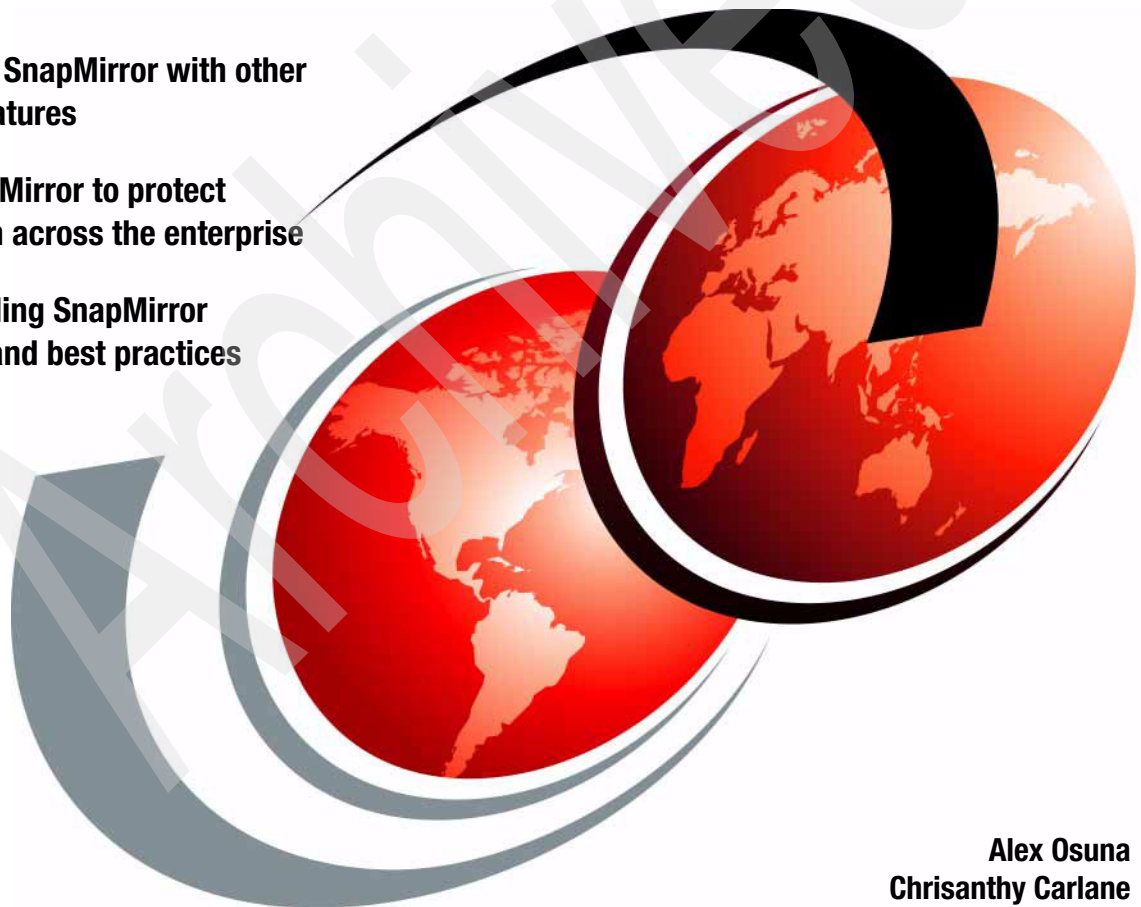


# IBM System Storage N Series SnapMirror

Integrating SnapMirror with other  
N series features

Using SnapMirror to protect  
information across the enterprise

Understanding SnapMirror  
behaviors and best practices



Alex Osuna  
Chrisanthy Carlane

[ibm.com/redbooks](http://ibm.com/redbooks)

**Redbooks**





International Technical Support Organization

## **IBM System Storage N Series SnapMirror**

July 2006

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

## **First Edition (July 2006)**

This edition applies to Version I, Release I, N series N3700 and N5000 series.

© Copyright International Business Machines Corporation 2006. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team that wrote this Redbook .....	ix
Become a published author .....	x
Comments welcome .....	xi
<b>Chapter 1. Basics of SnapMirror</b> .....	1
1.1 Why SnapMirror is needed .....	2
Overall benefits of SnapMirror .....	3
1.2 How SnapMirror works .....	4
1.2.1 SnapMirror configuration .....	5
1.2.2 Snapshot copy behavior in SnapMirror .....	15
1.3 Volume-based versus qtree-based SnapMirror .....	18
1.3.1 Volume-based SnapMirror .....	18
1.3.2 Qtree-based SnapMirror .....	20
1.3.3 Key differences between VSM and QSM .....	21
1.4 Three modes of SnapMirror .....	23
1.4.1 Asynchronous mode .....	24
1.4.2 Synchronous mode .....	24
1.4.3 Semi-synchronous mode .....	26
1.5 NVLOG (NVRAM Log) forwarding .....	26
1.5.1 Data ONTAP local system writes .....	27
1.5.2 Data ONTAP local system writes with NVLOG forwarding active ..	28
1.5.3 Data ONTAP local system writes with consistency point synchronization (CP Sync) active .....	29
1.5.4 How SnapMirror maintains synchronous state .....	30
1.6 SnapMirror advanced features .....	31
1.6.1 Multipath support .....	31
1.6.2 Visibility interval .....	32
1.6.3 Synchronicity level control .....	32
1.7 SnapMirror control files .....	33
1.8 SnapMirror and tape .....	35
1.8.1 Using SnapMirror for image backups .....	35
1.9 SnapMirror with FlexVol and FlexClone volumes .....	36
1.9.1 SnapMirror with FlexVol .....	37
1.9.2 SnapMirror with FlexClone .....	38

1.10 Using SnapMirror .....	40
1.10.1 Deleting source Snapshot copy in a SnapMirror deployment .....	40
1.10.2 Synchronous SnapMirror in an FCP environment. ....	40
1.10.3 Space guarantees in a SnapMirror environment. ....	41
1.10.4 SnapMirror and overcommitting the aggregate. ....	41
1.10.5 SnapMirror supported volume type and clustering .....	41
1.10.6 Logical replication (LREP). ....	42
1.10.7 Maximum number of simultaneous replication operations .....	43
<b>Chapter 2. SnapMirror best practices and behaviors .....</b>	<b>45</b>
2.1 General best practices .....	46
2.2 Best practices for deploying for performance .....	49
2.3 Known SnapMirror behaviors .....	50
2.4 SnapMirror troubleshooting .....	52
2.4.1 Common problem scenarios .....	56
<b>Chapter 3. SnapMirror integration with other N series storage system features .....</b>	<b>61</b>
3.1 DataFabric Manager .....	62
3.1.1 Creating a SnapMirror relationship with DFM .....	64
3.1.2 Managing SnapMirror relationships with DFM .....	65
3.1.3 Managing SnapMirror connections with DFM .....	66
3.1.4 Managing SnapMirror policy with DFM .....	67
3.1.5 Monitoring SnapMirror with DFM .....	69
3.1.6 Reviewing SnapMirror with DFM .....	70
3.2 SnapLock .....	70
3.2.1 Extracting data from SnapLock dump file .....	72
3.2.2 Regular volume to SnapLock compliance volume .....	73
3.2.3 End-to-end SnapLock Compliance VSM relationship .....	73
3.2.4 Synchronous SnapMirror and SnapLock volumes .....	73
3.3 SnapVault .....	73
3.3.1 SnapMirror interlock with SnapVault. ....	74
3.3.2 Using SnapMirror to create DR for SnapVault .....	74
3.3.3 Protecting SnapVault secondaries with volume-based SnapMirror ..	76
3.3.4 Using SnapMirror to migrate SnapVault .....	77
3.3.5 Using DataFabric Manager .....	79
3.3.6 Using SnapVault to protect a VSM destination .....	79
3.3.7 Key differences between SnapVault and SnapMirror (QSM) .....	79
3.4 SnapDrive .....	80
3.4.1 Replication upon SnapDrive Snapshot copy creation .....	81
3.4.2 Replication using rolling Snapshot copies and SnapDrive .....	82
3.4.3 How SnapDrive manages rolling Snapshot copies .....	82
3.4.4 How rolling Snapshot copies are named using SnapDrive .....	83

<b>Chapter 4. Moving SnapMirror sources for volume- and qtree-based SnapMirror</b>	85
4.1 Overview	86
4.2 VSM setup	86
4.2.1 VSM process	86
4.3 Moving the source for QSM	88
4.3.1 QSM setup	88
4.3.2 QSM process	89
<b>Chapter 5. Disaster recovery failover procedures and examples</b>	91
5.1 Procedure for cutting over to destination	92
5.2 Procedure to resync the source with the destination (minor event)	93
5.3 Procedure to rebuild the source from the destination (full disaster recovery)	94
<b>Chapter 6. LREP and SnapMirror</b>	97
6.1 How LREP works	98
6.2 Data center	101
<b>Index</b>	103





# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

*The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## **COPYRIGHT LICENSE:**


This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®

@server®

Redbooks (logo) ™

xSeries®

DFS™

IBM®

Redbooks™

System Storage™

SLC™

The following terms are trademarks of other companies:

Snapshot, SnapDrive, Data ONTAP, WAFL, SnapVault, SnapMirror, NetCache, DataFabric, The Network Appliance logo, the bolt design, Camera-to-Viewer, Center-to-Edge, ContentDirector, ContentFabric, NetApp Availability Assurance, NetApp ProTech Expert, NOW, NOW NetApp on the Web, RoboCache, RoboFiler, SecureAdmin, Serving Data by Design, Smart SAN, The evolution of storage, Virtual File Manager, and Web Filer are trademarks of Network Appliance, Inc. in the U.S. and other countries. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such.

Sun, SLC, VSM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

This IBM® Redbook introduces the features and capabilities of SnapMirror, and describes how SnapMirror technology can be used in an IBM System Storage™ N Series environment to achieve enterprise data protection.

SnapMirror provides a fast and flexible enterprise solution for mirroring or replicating data over local area, wide area, and Fibre Channel networks. By providing a simple solution for replicating data, SnapMirror addresses several critical application areas, including disaster recovery, remote data access, simulating production environments for testing purposes, creating remote archives, and load sharing.

This book is intended for IT professionals who are evaluating SnapMirror technology as well as those who are deploying and architecting a SnapMirror solution.

## The team that wrote this Redbook

This Redbook was produced by a team of specialists working at the International Technical Support Organization, Tucson, Arizona.

**Alex Osuna** is a Project Leader with the San Jose International Technical Support Organization. He has over 27 years in the IT industry and 22 years of experience in the hardware/software storage area dealing with maintenance, development, early ship programs, education, publishing, performance analysis and technical sales support. He holds 10 certifications from IBM, Microsoft®, and Red Hat.

**Chrisanthy Carlane** is an IT Specialist with IBM Information Technology Services in Indonesia. She has 5 years of experience providing enterprise-wide infrastructure implementations, migration and support on IBM Tape and Storage System and xSeries® servers. She has certifications with IBM, Cisco, Microsoft, Red Hat, and McDATA and holds a Bachelor of Economics - Accounting degree from Tarumanagara University, Jakarta, Indonesia.



*Chrisanthy Carlane and Alex Osuna*

Thanks to the following people from for their contributions to this project:

Chad Mitchell, IBM Systems & Technology Group, Development

William B. Wilson, IBM Systems & Technology Group, Development

Lamont V. Harrigan, IBM Systems & Technology Group, Development, RAS Engineer

Darrin Chapman, Network Appliance Corporation

Mike Federwisch, Network Appliance Corporation

Chuck Dufresne, Network Appliance Corporation

## **Become a published author**

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbook@us.ibm.com](mailto:redbook@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400



# Basics of SnapMirror

This chapter discusses the following:

- ▶ Why SnapMirror is needed
- ▶ How SnapMirror works
- ▶ Known behaviors of SnapMirror

## 1.1 Why SnapMirror is needed

There are several approaches to increasing data availability in the face of hardware, software, or even site failures. *Backups* provide a way to recover lost data from an archival medium (tape or disk). *Redundant hardware* technologies also help mitigate the damage caused by hardware issues or failures. *Mirroring* provides a third mechanism to ensure data availability and minimize downtime.

SnapMirror provides a fast and flexible enterprise solution for mirroring or replicating data over local area, wide area, and Fibre Channel (FC) networks. SnapMirror can be a key component in implementing enterprise data protection strategies. If a disaster occurs at a source site, businesses can access mission-critical data from a mirror on a remote N series storage system, ensuring uninterrupted operation.

By providing a simple solution for replicating data across local, wide area, and FC networks, SnapMirror addresses five critical application areas, which are described in the following sections.

### Disaster recovery

Disaster recovery (DR) from a geographically remote location is now possible from a technical perspective. If critical data is mirrored to a different physical location, a serious disaster does not necessarily mean extended periods of data unavailability. The mirrored data can be made available to clients across the network until the damage caused by the disaster is repaired. Recovery may include recovery from corruption, natural disaster at the source site, accidental deletion, sabotage, and so on. SnapMirror is often used for DR planning. Here, data could be mirrored to a destination system at a DR facility. Preferably, application servers would be mirrored to this facility as well. If the DR facility needs to be made operational, applications can be switched over to the servers at the DR site and all application traffic directed to these servers for as long as necessary to recover the source site. When the source site is back online, SnapMirror can be used to transfer the data efficiently back to the production storage systems. After the production site takes over normal application operation again, SnapMirror transfers to the DR facility can resume without requiring a second complete data transfer.

### Remote data access

The data replication capability of SnapMirror allows the distribution of large amounts of data throughout the enterprise, allowing local read-only access to data. Remote data access not only provides faster access to data by local clients, but also results in a more efficient and predictable use of expensive network and server resources. This allows the source data to be replicated or



mirrored at a time chosen by systems administrators, with the intent of minimizing overall network utilization.

### **Application testing**

SnapMirror can make copies of applications for use in test beds, and replication of database environments to be used for testing or simulating production environments. Performance modeling and benchmarking and general development testing can also be done.

### **Remote tape archive**

Some environments require off-site storage or off-site archiving. When a tape device is attached to a SnapMirror destination, data can be moved to tape periodically. SnapMirror can also be used for backup consolidation and for offloading tape backup overhead from production servers. This facilitates centralized backup operations, reducing backup administrative requirements at remote locations. It can also dramatically reduce overhead from stressful backup operations caused by small backup windows on production storage systems. Because backup operations are not occurring on the production systems, small backup windows are not as important.

### **Load sharing**

Load sharing is similar to the remote data access example described previously in both implementation and benefits. The difference in a load-sharing implementation lies in the distance between the source and target volumes of the replication, as well as in the performance goals associated with the replication implementation. In load sharing the goal is to minimize the contention for critical application or database server resources by moving all read-only activities off the critical “transaction” server to a “mirror” or read-only server. The benefit can be twofold:

1. Network access to the same data set is now partitioned and can be further optimized.
2. CPU contention on the source application server is now reduced by having read-only and reporting users access the mirrored data.

## **Overall benefits of SnapMirror**

The benefits provided by implementing SnapMirror include the following:

- ▶ Block-level updates reduce bandwidth and time requirements
- ▶ Data consistency can be maintained at a DR site
- ▶ A DR plan can be tested without affecting production
- ▶ Synchronization between source and destination sites is complete

- ▶ Mission-critical data can be mirrored
- ▶ A DR location can keep many Snapshot copies at once; data can be restored to a point in time before data corruption occurred
- ▶ Data can be replicated between dissimilar N series storage systems
- ▶ A standard IP or FC network can be used for replication
- ▶ SnapMirror supports one-to-one, one-to-many, many-to-one, or many-to-many replication, referred to as cascading and multihop

## 1.2 How SnapMirror works

SnapMirror replicates Snapshot copy images from a *source volume* or *qtree* to a partner destination volume or qtree, thus replicating source object data to destination objects at regular intervals. SnapMirror source volumes and qtrees are writable data objects whose data is to be replicated. The source volumes and qtrees are the objects that are normally visible, accessible, and writable by the storage system's clients.

The SnapMirror destination volumes and qtrees are read-only objects, usually on a separate storage system, to which the source volumes and qtrees are replicated. SnapMirror destination volumes and qtrees are used for:

- ▶ Auditing purposes before the objects are converted to writable objects.
- ▶ Data verification.
- ▶ True mirrors for recovery from a disaster. In this case, a disaster takes down the source volumes or qtrees and the administrator uses SnapMirror commands to make the replicated data at the destination accessible and writable.

SnapMirror uses information in control files to maintain relationships and schedules. One of these control files, the `snapmirror.conf` file, is located on the destination system and allows scheduling to be maintained. The `snapmirror.conf` file, along with information entered via the `snapmirror.access` option or the `snapmirror.allow` file, is used to establish a relationship between a specified source volume or qtree for replication, and the destination volume or qtree where the mirror is kept.

The SnapMirror update process performs the following tasks:

1. Creates a Snapshot copy of the data on the source volume or qtree.
2. Copies the data to the destination, a read-only volume or qtree on the same system or on a remote destination system.

3. Updates the destination file system to reflect incremental changes occurring to the source.

The result of this process is an online, read-only data set that is a point-in-time view of the data on the source at the time of the most recent update.

When `snapmirror.conf` is used, the SnapMirror Snapshot copy creation and updates are controlled by a schedule that is local to the destination N series storage system. In a SAN environment, Snapshot copy creation involving logical unit numbers (LUNs) must be controlled by host systems. Scripts are set up to create Snapshot copies and to initiate the SnapMirror update to the remote storage system. These operations can be scheduled on the host system using the UNIX® **cron** utility, Microsoft Windows® Task Scheduler service, or a third party's scheduler.

## 1.2.1 SnapMirror configuration

This section provides step by step instructions for configuring SnapMirror. In this scenario, `itsotuc1` is the source storage and `itsotuc2` is the destination storage.

### Initializing SnapMirror relationship

Use the following procedure to initialize a SnapMirror relationship:

1. Telnet to `itsotuc1` and `itsotuc2`.
2. Verify that a SnapMirror license is installed on each storage system. If not, install a SnapMirror license, as in Example 1-1.

#### *Example 1-1 Adding SnapMirror license*

---

```
itsotuc2> license add abcdefg  
A snapmirror site license has been installed.  
snapmirror enabled.
```

---

3. In `itsotuc1`:  
Verify that source volume, `vol1`, exists and is online. If `vol1` does not already exist, then create a new volume named `vol1` on your storage and confirm its status, as in Example 1-2.

*Example 1-2 Create source volume in itsotuc1*

---

```
itsotuc1> vol create vol1 aggr1 25m
Creation of volume 'vol1' with size 25m on containing aggregate
'aggr1' has completed.
itsotuc1> vol status vol1
Volume State      Status      Options
vol1  online      raid_dp, flex  create_ucose=on,
convert_ucose=on

Containing aggregate: 'aggr1'
```

---

4. In **itsotuc2**:

Verify that destination volume, vol2, exists and is online. If vol2 does not already exist, then create a new volume named vol2 on your storage and confirm its status, as in Example 1-3.

*Example 1-3 Creating destination volume in itsotuc2*

---

```
itsotuc2> vol create vol2 aggr2 25m
Creation of volume 'vol2' with size 25m on containing aggregate
'aggr2' has completed.
itsotuc2> vol status vol2
Volume State      Status      Options
vol1  online      raid_dp, flex  create_ucose=on,
convert_ucose=on

Containing aggregate: 'aggr2'
```

---

5. Verify that both volumes are in Unicode format. Directories on all SnapMirror source volumes that support CIFS clients *must* be in Unicode format before being replicated to a destination; otherwise, the directories in the read-only destination volume will not be in Unicode format and attempts through CIFS to access directories and open files on the destination volume can receive “access denied” errors. Do this by entering the command in Example 1-4 and make sure convert\_unicode is set to on.

*Example 1-4 Verify both volumes are Unicode format*

---

```
itsotuc2> vol status vol2
Volume State      Status      Options
vol2  online      raid_dp, flex  create_ucose=on,
convert_ucose=on,
fs_size_fixed=on

Containing aggregate: 'aggr2'
```

---

If convert\_ucose is not set to on, enter the following command:

```
vol options vol1 convert_ucose on
```

6. Verify that vol2 is the same size or larger than vol1 on the other filer with the **vol status -b** command, as in Example 1-5.

*Example 1-5 Verify that source and destination volumes are acceptable size*

In itsotuc1:

```
itsotuc1> vol status -b vol1
Volume  Block Size (bytes)  Vol Size (blocks)  FS Size (blocks)
-----
vol1    4096                6400              6400
```

In itsotuc2:

```
itsotuc2> vol status -b vol1
Volume  Block Size (bytes)  Vol Size (blocks)  FS Size (blocks)
-----
vol2    4096                6400              6400
```

7. In itsotuc2, restrict the destination volume for the other filer with the **vol restrict** command and check its status with the **vol status** command. See Example 1-6.

*Example 1-6 Restrict destination volume*

```
itsotuc2> vol restrict vol2
Fri Mar 24 05:51:54 GMT [cifs.terminationNotice:warning]: CIFS: shut
down completed: CIFS is disabled for volume vol2.
Volume 'vol2' is now restricted.
itsotuc2> vol status vol2
Volume  State      Status      Options
vol2    restricted  raid_dp, flex
Containing aggregate: 'aggr2'
```

8. In both storage systems, specify the destination hosts that are allowed to access the source and vice versa. Use the **snapmirror.access** option to verify the current configuration and specify the hosts allowed. See Example 1-7.

In itsotuc1:

```
itsotuc1> options snapmirror.access host=itsotuc2
itsotuc1> options snapmirror.access
snapmirror.access          host=itsotuc2
```

In itsotuc2:

```
itsotuc2> options snapmirror.access host=itsotuc1
itsotuc2> options snapmirror.access
snapmirror.access          host=itsotuc1
```

---

9. In both storage systems, start snapmirror by issuing the **snapmirror on** command.

10. In both storage systems, create CIFS shares for source and destination volumes, as follows:

In itsotuc1:

```
itsotuc1> cifs shares -add srcvol /vol/vol1
```

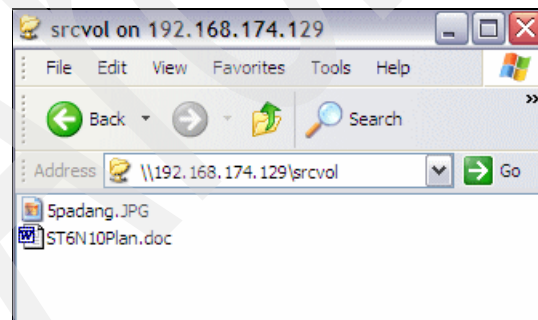
In itsotuc2:

```
itsotuc2> cifs shares -add destvol /vol/vol2
```

11. Access snapmirror source volume srcvol share and create some new files or directories (Figure 1-1) using the following sign on specification:

Share Login: administrator

Password: <CIFS password you configured during setup>



*Figure 1-1 SnapMirror source volume*

12. In itsotuc2, run the **snapmirror initialize** command to start the first SnapMirror baseline transfer from itsotuc1 to itsotuc2 (Example 1-8). This command is case sensitive.

*Example 1-8 snapmirror initialize command*

```
itsotuc2> snapmirror initialize -S itsotuc1:vol1 itsotuc2:vol2
Transfer started.
Monitor progress with 'snapmirror status' or the snapmirror log.
```

13. Enter the **snapmirror status** command to check the transfer status (Example 1-9).

*Example 1-9 snapmirror status command*

```
itsotuc2> snapmirror status
Snapmirror is on.
Source      Destination      State      Lag  Status
itsotuc1:vol1 itsotuc2:vol2    Uninitialized -    Transferring
itsotuc2> snapmirror status
Snapmirror is on.
Source      Destination      State      Lag  Status
itsotuc1:vol1 itsotuc2:vol2    Snapmirrored 00:00:26 Idle
```

14. Access snapmirror destination volume destvol share and try to edit or delete a file (Figure 1-2).

Share Login: administrator

Password: <CIFS password you configured during setup>

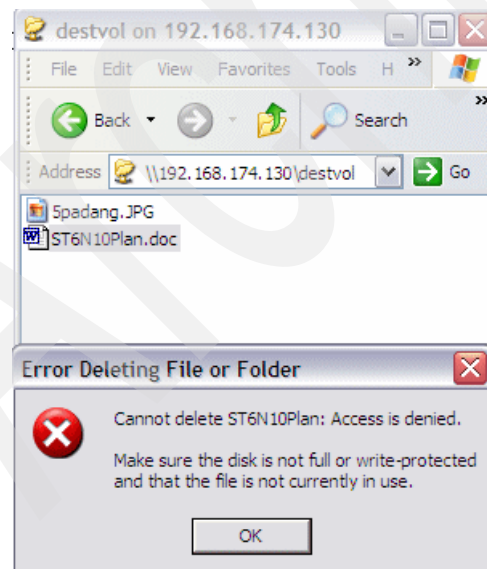


Figure 1-2 Delete file in a snap mirrored volume

**Note:** With snapmirror active, you cannot edit or delete the file or directory in the destination volume.

## Breaking a SnapMirror relationship and making the destination storage active

When catastrophe happens and the source SnapMirror storage fails, you have to break the mirrors and make the destination SnapMirror storage read-write so that the business can continue on the destination storage.

Use the following procedures to break a SnapMirror relationship and make the destination storage active:

1. Turn off SnapMirror in both storage systems with **snapmirror off** and check its status with the **snapmirror status** command. This prevents SnapMirror from trying to perform updates and unintentionally updating the destination qtree and volumes from the source qtrees or volumes. Enter the command in Example 1-10 on both storage systems.

### *Example 1-10 Turn off SnapMirror*

---

```
itsotuc2> snapmirror off
itsotuc2> snapmirror status
Snapmirror is off.
```

Source	Destination	State	Lag	Status
itsotuc1:vol1	itsotuc2:vol2	Snapmirrored	00:04:44	Idle

---

2. Break the mirror of itsotuc1:vol1 on itsotuc2:vol2 as in Example 1-11.

### *Example 1-11 Breaking SnapMirror*

---

```
itsotuc2> snapmirror break itsotuc2:vol2
snapmirror break: Destination vol2 is now writable.
```

Volume size is being retained for potential snapmirror resync. If you would like to grow the volume and do not expect to resync, set vol option fs\_size\_fixed to off.

---

After breaking SnapMirror, the destination SnapMirror is now in read-write status and available to be used.

## Resyncing the mirror

After source storage is back to normal, you need to resynchronize broken mirrors in order to reestablish the original relationship. The resynchronization process



effectively involves reversing the source destination relationship temporarily for one resync. That is, itsotuc2 is the temporary source and itsotuc1 is the temporary destination. Then there is a second resync where itsotuc1 is the source and itsotuc2 is the destination. Use the following procedures to reestablish the original relationship between the two storage systems.

1. Make sure itsotuc1 vol1 and itsotuc2 vol2 are online, as in Example 1-12.

*Example 1-12 Source and destination volume status must be online*

In itsotuc1:

```
itsotuc1> vol status vol1
Volume  State  Status  Options
vol1    online  raid_dp,flex  create_ucose=on,
                        convert_ucose=on
Containing aggregate: 'aggr1'
```

In itsotuc2:

```
itsotuc2> vol status vol2
Volume  State  Status  Options
vol2    online  raid_dp, flex  create_ucose=on,
                        convert_ucose=on
Containing aggregate: 'aggr2'
```

2. Make sure SnapMirror on both storage systems is on (Example 1-13).

*Example 1-13 Make sure Snapmirror on both storage systems is on*

In both storage systems:

```
itsotuc1> snapmirror status
Snapmirror is off.
Source          Destination      State  Lag      Status
itsotuc1:vol1   itsotuc2:vol2    Source  0:10:20  Idle
itsotuc1> snapmirror on
```

3. Start the resync for vol1 on itsotuc1. Notice that itsotuc2:vol2 is the source because it has the most recent updates and itsotuc1:vol1 is the destination. Enter the command shown in Example 1-14.

*Example 1-14 SnapMirror resynchronization from itsotuc2 to itsotuc1*

In itsotuc1:

```
itsotuc1> snapmirror resync -S itsotuc2:vol2 itsotuc1:vol1
The resync base snapshot will be: itsotuc2(0099902332)_vol2.1
Are you sure you want to resync the volume? y
```

```

Fri Mar 24 06:18:00 GMT [snapmirror.dst.resync.info:notice]:
SnapMirror resync of vol1 to itsotuc2:vol2 is using
itsotuc2(0099902332)_vol2.1 as the base snapshot.
Volume vol1 will be briefly unavailable before coming back online.
Share srcvol disabled while volume vol1 is offline.
Fri Mar 24 06:18:01 GMT [cifs.terminationNotice:warning]: CIFS: shut
down completed: CIFS is disabled for volume vol1.
Fri Mar 24 06:18:05 GMT [waf.snaprestore.revert:notice]: Reverting
volume vol1 to a previous snapshot.
Fri Mar 24 06:18:06 GMT [waf.vol.guarantee.replica:info]: Space for
replica volume 'vol1' is not guaranteed.
Share srcvol activated.
Revert to resync base snapshot was successful.
Fri Mar 24 06:18:07 GMT [snapmirror.dst.resync.success:notice]:
SnapMirror resync of vol1 to itsotuc2:vol2 successful.
Transfer started.
Monitor progress with 'snapmirror status' or the snapmirror log.

```

---

4. The **snapmirror resync** command used a baseline snapshot for the resync. Now use the **snapmirror update** command to request any changes made since the baseline snapshot was used. Issue the command shown in Example 1-15 to complete the synchronization of itsotuc2:vol2 with itsotuc1:vol1.

*Example 1-15 snapmirror update command to update changes in itsotuc2:vol2*

---

```

itsotuc1> snapmirror update -S itsotuc2:vol2 itsotuc1:vol1
Transfer started.
Monitor progress with 'snapmirror status' or the snapmirror log.

```

```

itsotuc1> snapmirror status
Snapmirror is on.

```

Source	Destination	State	Lag	Status
itsotuc2:vol2	itsotuc1:vol1	Snapmirrored	06:44:01	Transferring
itsotuc1:vol1	itsotuc2:vol2	Source	06:56:56	Idle

---

5. Check the volume status with the **vol status** command. Notice that itsotuc1:vol1 status is snapmirrored and read-only (Example 1-16).

*Example 1-16 Volume status of itsotuc1:vol1 and itsotuc2:vol2*

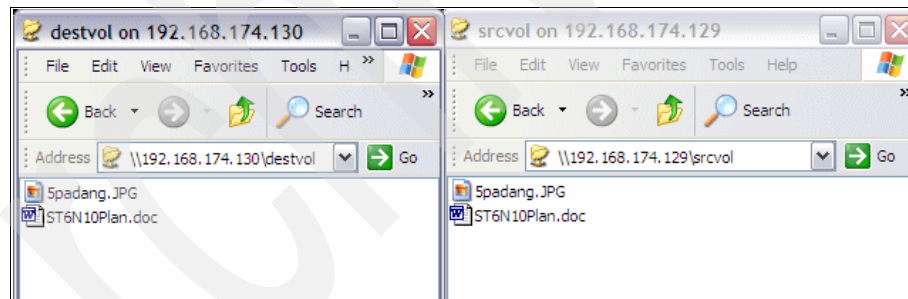
In itsotuc1:

```
itsotuc1> vol status vol1
Volume      State      Status      Options
vol1        online    raid_dp, flex snapmirrored=on,
                        snapmirrored create_ucose=on,
                        read-only    convert_ucose=on,
                        fs_size_fixed=on,
                        guarantee=volume(disabled)
Containing aggregate: 'aggr1'
```

In itsotuc2:

```
itsotuc2> vol status vol2
Volume      State      Status      Options
vol2        online    raid_dp, flex create_ucose=on,
                        convert_ucose=on,
                        fs_size_fixed=on
Containing aggregate: 'aggr2'
```

6. Access the CIFS shared directory srcvol in itsotuc1 and destvol in itsotuc2. Make sure their content is the same (Figure 1-3).



*Figure 1-3 srcvol and destvol share must be identical*

7. You have just completed the first resync and made itsotuc1:vol1 a SnapMirror of itsotuc2:vol2. In Example 1-17, you can see that the source itsotuc1:vol1 and destination itsotuc2:vol2 is broken off and is no longer valid. The current snapmirror status has itsotuc2:vol2 as the source and itsotuc1:vol1 as the destination.

*Example 1-17 Check snapmirror source and destination after snapmirror is broken*

---

In itsotuc2:

```
itsotuc2> snapmirror status
```

Snapmirror is on.

Source	Destination	State	Lag	Status
itsotuc1:vol1	itsotuc2:vol2	Broken-off	07:54:16	Idle
itsotuc2:vol2	itsotuc1:vol1	Source	00:09:38	Idle

---

8. You have to reverse the mirror relationship so that itsotuc1 is once again the source for these objects and itsotuc2 is the destination. In order to do that, you have to break the snapmirror in itsotuc1 with the **snapmirror break** command and start resync from itsotuc2 with itsotuc1 as the source by issuing the **snapmirror resync** command. See Example 1-18.

*Example 1-18 Reversing snapmirror source and destination volume*

---

In itsotuc1:

```
itsotuc1> snapmirror break itsotuc1:vol1
```

snapmirror break: Destination vol1 is now writable.

Volume size is being retained for potential snapmirror resync. If you would like to grow the volume and do not expect to resync, set vol option fs\_size\_fixed to off.

```
itsotuc1> snapmirror status
```

Snapmirror is on.

Source	Destination	State	Lag	Status
itsotuc2:vol2	itsotuc1:vol1	Broken-off	00:21:40	Idle
itsotuc1:vol1	itsotuc2:vol2	Source	08:06:18	Idle

In itsotuc2:

```
itsotuc2> snapmirror resync -S itsotuc1:vol1 itsotuc2:vol2
```

The resync base snapshot will be: itsotuc1(0099922126)\_vol1.4

Are you sure you want to resync the volume? y

Fri Mar 24 14:13:47 GMT [snapmirror.dst.resync.info:notice]:

SnapMirror resync of vol2 to itsotuc1:vol1 is using

itsotuc1(0099922126)\_vol1.4 as the base snapshot.

Volume vol2 will be briefly unavailable before coming back online.

Fri Mar 24 14:13:48 GMT [cifs.terminationNotice:warning]: CIFS: shut down completed: CIFS is disabled for volume vol2.

Share destvol disabled while volume vol2 is offline.

Fri Mar 24 14:13:50 GMT [waf1.snaprestore.revert:notice]: Reverting volume vol2 to a previous snapshot.

Fri Mar 24 14:13:51 GMT [waf1.vol.guarantee.replica:info]: Space for replica volume 'vol2' is not guaranteed.

Share destvol activated.

```

Revert to resync base snapshot was successful.
Fri Mar 24 14:13:52 GMT [snapmirror.dst.resync.success:notice]:
SnapMirror resync of vol2 to itsotuc1:vol1 successful.
Transfer started.
Monitor progress with 'snapmirror status' or the snapmirror log.
itsotuc2> snapmirror status
Snapmirror is on.
Source          Destination  State      Lag      Status
itsotuc1:vol1   itsotuc2:vol2 Snapmirrored 00:23:22 Transferring
itsotuc2:vol2   itsotuc1:vol1 Source      00:23:22 Idle

```

---

9. After resync, issue a **vol status** command in each storage system. Now itsotuc1:vol1 status is read-write while itsotuc2:vol2 is snapmirrored, read-only (Example 1-19).

*Example 1-19 Resynchronized SnapMirror volume status*

```

In itsotuc1:
itsotuc1>vol status vol1
Volume State      Status      Options
vol1  online      raid_dp, flex  create_ucose=on,
convert_ucose=on,
fs_size_fixed=on
Containing aggregate: 'aggr1'

In itsotuc2:
itsotuc2>vol status vol2
Volume State      Status      Options
vol2  online      raid_dp, flex  snapmirrored=on,
snapmirrored      create_ucose=on,
read-only          convert_ucose=on,
fs_size_fixed=on
guarantee=volume(disabled)
Containing aggregate: 'aggr2'

```

---

## 1.2.2 Snapshot copy behavior in SnapMirror

SnapMirror uses a Snapshot copy as a marker for a point in time for the mirroring process. A copy is kept on the source volume as the current point in time that both mirrors are in sync. When an update occurs, a new copy is created and is compared against the previous copy to determine the changes since the last update. SnapMirror marks the copies it needs to keep for a particular destination mirror in such a way that the **snap list** command shows the keyword **snapmirror** next to the necessary Snapshot copies (Example 1-20).

*Example 1-20 snap list command*

---

```
itsotuc1>snap list
Volume vol0
working...
%/used  %/total  date          name
-----
0%(0%)  0%(0%)  Mar 24 12:00 hourly.0
1%(0%)  0%(0%)  Mar 24 08:00 hourly.1

Volume vol1
working...
%/used  %/total  date          name
-----
11%(11%) 0%(0%)  Mar24 14:13 itsotuc2(0099902332)_vol2.2(snapmirror)
21%(12%) 0%(0%)  Mar24 13:50 itsotuc1(0099922126)_vol1.4(snapmirror)
57%(52%) 2%(1%)  Mar24 13:02 itsotuc1(0099922126)_vol1.3
60%(14%) 2%(0%)  Mar24 12:00 hourly.0
62%(12%) 2%(0%)  Mar24 08:00 hourly.1
78%(66%) 4%(2%)  Mar24 06:06 itsotuc2(0099902332)_vol2.1(snapmirror)
```

---

The **snapmirror destinations** command can be used to see what mirror a particular copy is marked as required for at any time. On the source volume, SnapMirror creates the Snapshot copy for a particular destination and immediately marks it for that destination. At this point, both the previous copy and the new copy are marked for this destination. Once a transfer is successfully completed, the mark for the previous copy is removed and deleted. Snapshot copies left for cascade mirrors from the destinations that have the snapmirror tag in the **snap list** command output. (Cascade mirrors are a variation on the basic SnapMirror deployment, involving a writable source volume replicated to multiple read-only destinations, either one-to-one or one-to-many.) See Example 1-21.

*Example 1-21 snapmirror destinations command*

---

```
itsotuc1> snapmirror destinations
Path          Destination
vol1          itsotuc2:vol2->itsotuc1:vol1->itsotuc2:vol2
vol1          itsotuc2:vol2->itsotuc1:vol1
vol1          itsotuc2:vol2
```

---

Use the **snapmirror destinations -s** command to find out why a particular Snapshot copy is marked. This mark is kept mainly for SnapMirror purposes, as a reminder that it should not delete a copy. See Example 1-22.

*Example 1-22 snapmirror destinations -s command*

---

```
itsotuc1> snapmirror destinations -s
Path Snapshot                               Destination
vol1 itsotuc2(0099902332)_vol2.1
itsotuc2:vol2->itsotuc1:vol1->itsotuc2:vol2
vol1 itsotuc1(0099922126)_vol1.4 itsotuc2:vol2->itsotuc1:vol1
vol1 itsotuc2(0099902332)_vol2.2 itsotuc2:vol2
```

---

This mark does not stop a user from deleting a copy marked for a destination that will no longer be a mirror; use the **snapmirror release** command to force a source to forget about a particular destination. This is a safe way to have SnapMirror remove its marks and clean up Snapshot copies that are no longer needed. Deleting a Snapshot copy that is marked as needed by SnapMirror is not advisable and must be done with caution in order not to disallow a mirror from updating. While a transfer is in progress, SnapMirror uses the busy lock on a Snapshot copy. This can be seen in the **snap list** command output. These locks do prevent users from deleting the Snapshot copy. The busy locks are removed when the transfer is complete. See Example 1-23.

*Example 1-23 snapmirror release command*

---

```
itsotuc2> snapmirror release vol2 itsotuc1:vol1
itsotuc2> snapmirror destinations
snapmirror destination: no known destinations
```

---

For volume replication, SnapMirror creates a Snapshot copy of the whole source volume that is copied to the destination volume. For qtree replication, SnapMirror creates Snapshot copies of one or more source volumes that contain qtrees identified for replication. This data is copied to a qtree on the destination volume, and a Snapshot copy of that destination volume is created.

A volume Snapshot copy name has the following format:

dest\_filer (sysid)\_name.number

Where:

- dest\_filer is the host name of the destination storage system.
- sysid is the destination system ID number.
- name is the name of the destination volume.
- number is the number of successful transfers for the Snapshot copy, starting at 1. Data ONTAP increments this number for each transfer.

In our scenario, this is:

itsotuc1(0099922126)\_vol1.4 (snapmirror)

A qtree SnapMirror (QSM) Snapshot copy name has the following format:

```
dest_filer(sysid)_name-src|dst.number
```

Where:

- `dest_filer` is the host name of the destination storage system.
- `sysid` is the destination system ID number.
- `name` is the name of the destination volume or qtree path. 7
- `src|dst` identifies the Snapshot copy location.
- `number` is an arbitrary start point number for the Snapshot copy. Data ONTAP increments this number for each transfer.

In our scenario, this is:

```
filerA(0050409813)_vol1_qtree3-dst.15 (snapmirror)
```

In the output of the **snap list** command, Snapshot copies needed by SnapMirror are followed by the SnapMirror name in parentheses.

**Caution:** Deleting Snapshot copies marked “SnapMirror” can cause SnapMirror updates to fail.

## 1.3 Volume-based versus qtree-based SnapMirror

SnapMirror software provides the ability to replicate individual qtrees as well as whole volumes. The difference is literally *physical* versus *logical*. There are trade-offs, including performance, manageability, configuration, and infrastructure resources. A comparison of the two is necessary to understand the various implications.

### 1.3.1 Volume-based SnapMirror

The *physical* (block-for-block) replication of the layout of data on disk is often referred to as *Volume SnapMirror (VSM™)*. Volume-based replication is a powerful tool in migrating entire data sets and all related copies of those data sets. Furthermore, since the entire volume is being replicated, it operates on raw devices rather than accessing the data through a file system or other logical layer.

This generally enables volume-based SnapMirror to perform at an increased level for replicating an entire volume, since it eliminates an entire layer of abstraction. However, it does not allow you to exclude data in a volume from



replication or to specify different replication settings for different data in a volume. Everything contained in a volume is replicated from one system or location to another, including metadata about the volume itself, such as language translation settings and other volume options, as well as all Snapshot copies of the volume.

With the introduction of FlexVol volumes, volume replication can be as granular as traditional deployments of qtree-based replication. The entire volume is replicated and may be very large, but it can also be very small and used in the same way that a qtree is used in more traditional deployments.

After performing an initial transfer of all data in the volume, VSM sends to the destination only the blocks that have changed since the last successful replication. To find the changed blocks, SnapMirror first creates a new Snapshot copy before performing the initial transfer. This copy is referred to as the baseline Snapshot copy, but it is not really used until the first or next update. When SnapMirror performs an update transfer, it first creates another new Snapshot copy.

All Data ONTAP volumes contain a block map that acts as a lookup table between a physical block in a traditional volume or aggregate and a Write Anywhere File Layout (WAFL) file system block. When a Snapshot copy is created on a volume, the block map is part of this copy. To find changed blocks between the baseline Snapshot copy and the new Snapshot copy, the block map in the new copy is compared to the block map in the baseline copy. Any changed blocks are thus identified and transferred to the destination. Once the transfer is complete, the new Snapshot copy becomes the baseline copy and the old baseline copy is deleted.

SnapMirror volume replication has the following characteristics:

- ▶ SnapMirror volume replication can be synchronous or asynchronous.
- ▶ SnapMirror volume replication can occur only with volumes of the same type; that is, both volumes are traditional volumes or both volumes are flexible volumes.
- ▶ SnapMirror volume replication copies a volume and all of its Snapshot copies to a destination volume.
- ▶ A destination volume set up for SnapMirror volume replication must first be set to restricted, read-only status.
- ▶ The destination volume (entire volume) is read-only.
- ▶ SnapMirror volume replication is a block-for-block replication; it transfers the file system verbatim. Therefore, older releases of Data ONTAP cannot understand file system transfers from a later release of Data ONTAP.

**Note:** You can use volume-based SnapMirror to mirror files to a newer release to assist in migrating to a newer Data ONTAP version, but you cannot do this in the reverse direction. Specifically, the destination of a volume SnapMirror (VSM) relationship must run a version of Data ONTAP that is equal to or more recent than the source. In addition, the source and destination must be on the same Data ONTAP release.

### 1.3.2 Qtree-based SnapMirror

The *logical* (qtree) replication of the layout of data on disk is often referred to as *Qtree SnapMirror* (QSM). Configuring replication based on qtrees rather than on whole traditional volumes is a powerful way to manage data. IT managers can choose replication schedules and destinations based on the content of the data and the business needs of the organization, ignoring the physical layout of the storage. Selective replication is also possible, saving valuable network bandwidth by configuring SnapMirror to transfer only the subsets of data that are needed at the destination.

In order to replicate qtrees, the SnapMirror software first creates a Snapshot copy on the source volume that contains the qtree to be replicated. Note that Snapshot technology always operates on volumes, not on qtrees. This Snapshot copy then contains a point-in-time copy of all of the data on the source volume, including both the data in the qtree to be replicated and also (presumably) other data that is not to be replicated.

Each volume in Data ONTAP contains one inode file, which contains entries for all files in the volume. QSM determines changed data by first looking through the inode file for inodes that have changed and then looking through the changed inodes of the pertinent qtree for changed data blocks. The SnapMirror software then transfers across the network only the new or changed data blocks from this Snapshot copy that are associated with the designated qtree. On the destination volume, a new Snapshot copy is then created that contains a complete point-in-time copy of the entire destination volume, but that is associated specifically with the particular qtree that has been replicated.

SnapMirror qtree replication has the following characteristics:

- ▶ SnapMirror qtree replication is not available with synchronous mode.
- ▶ SnapMirror qtree replication occurs between qtrees regardless of the type of volume (traditional or flexible) in which the qtree resides.
- ▶ SnapMirror qtree replication is asynchronous only.
- ▶ SnapMirror qtree replication copies only the contents of an individual qtree to a destination.

- ▶ If you need to mirror only the data stored on an individual qtree, SnapMirror replication of that individual qtree uses less disk space on the storage system.
- ▶ A destination qtree is read-only, but the volume on which it is located must be online and writable.
- ▶ SnapMirror qtree replication is a logical replication; all of the files and directories in the source file system are created in the destination file system.

### 1.3.3 Key differences between VSM and QSM

The important differences between VSM and QSM are the following:

- ▶ QSM is not sensitive to disk size differences between the source and the destination, nor is it sensitive to differences in checksum technology used on the source and destination volumes. Because of this capability, QSM may be preferred for replicating data between volumes with differing disk technologies in a traditional (before FlexVol) deployment. In a FlexVol deployment there is no disk geometry issue.
- ▶ QSM is sensitive to the number of files in a volume, due to the nature of the qtree replication process. If large numbers of files are being replicated, VSM may be preferable.
- ▶ VSM requires that the size of the destination volume be equal to or greater than the size of the source volume, while QSM requires the destination volume to have less free space than the amount of space actually consumed by the source qtree. The introduction of FlexVol volumes makes this less of an issue for volume-based replication because volume sizes are variable and are more granular than traditional volumes.
- ▶ QSM destinations can be placed on the root volume of the destination appliance. The root volume cannot be used as a destination for VSM. This provides increased flexibility of configuration.
- ▶ VSM replicates all Snapshot copies on the source volume to the destination volume. QSM replicates only one Snapshot copy of the source qtree (the copy created by the SnapMirror software at the time of the transfer) to the destination qtree. To copy all Snapshot copies from the source, VSM is the better option.
- ▶ QSM allows independent Snapshot schedules on the source and destination, and Snapshot copies can be created on the destination. This is not the case with VSM.
- ▶ Volume-based replication allows you to create clean Snapshot copies that are automatically replicated to the mirror. Qtree-based replication does not allow this.

- ▶ A VSM destination volume is always a replica of a single source volume. A QSM destination volume may contain replicated qtrees from multiple source volumes on one or more appliances and may also contain qtrees or non qtree data not managed by SnapMirror software.
- ▶ Multiple relationships would have to be created to replicate all qtrees in a given volume using qtree-based replication. Volume-based replication can take care of this in one relationship (as long as the one volume contains all relevant qtrees).
- ▶ VSM can be initialized using a tape device (SnapMirror to tape) by using the snapmirror store and snapmirror retrieve commands. QSM does not support initialization via tape devices. For large data sets, this may be an important consideration when planning which SnapMirror technology to use.
- ▶ Cascading of mirrors (copying from one destination to another in a series) is supported only for VSM. A multi-hop option is also available. Sometimes referred to as “mirroring from mirrors,” this option can be very useful for data distribution.
- ▶ VSM updates are delayed during a dump of the destination volume to tape using the dump command or NDMP-based backup tools. QSM updates are not affected by backup operations. This allows a strategy called continuous backup, in which traditional backup windows are eliminated and tape library investments are fully utilized. SnapVault software, discussed later, is optimized for continuous backup applications.
- ▶ Qtree-based replication can fail if the destination file system is full; volume-based replication will not encounter this situation unless FlexVol volumes are being used.

Both volume-based SnapMirror and qtree-based SnapMirror are useful tools. The decision of which to use depends on individual site requirements. Volume-based SnapMirror and qtree-based SnapMirror can be freely mixed on both source and destination systems, although any individual destination volume can be a destination for only one or the other. Table 1-1 shows the SnapMirror type recommended for various replication objectives.

Table 1-1 SnapMirror recommendations

Replication objective	SnapMirror type	
	VSM	QSM
Replication of all data on a volume, including Snapshot copies	Yes	
Large number of small files on source	Yes	
Cascading of mirrors	Yes	
Synchronous or semi-synchronous replication	Yes	
Pre-seeding with tape (SM2T)	Yes	
Most up-to-date copy on destination	Yes	
Pre-seeding with LREP (Logical Replication_ portable disk device		Yes
Mirroring destination on root volume		Yes
Different versions of Data ONTAP on source and destination		Yes
Different disk geometry between source and destination		Yes
Replication to another N series storage system		Yes
Need SnapShot copies created on destination		Yes
Multiple replicated qtrees (many to one) to a single destination		Yes
No tolerance for replication disruption during source backup using dump or NDMP to tape		Yes
Distance beyond 150 miles (241 kilometers)		Yes
Application that is sensitive to latency		Yes

## 1.4 Three modes of SnapMirror

SnapMirror can be used in three different modes: *asynchronous*, *synchronous*, and *semi-synchronous*. These modes are explained in the following sections.

## 1.4.1 Asynchronous mode

For both QSM and VSM, SnapMirror can operate in asynchronous mode. In this mode, SnapMirror performs incremental, block-based replication as frequently as once per minute. Performance impact on the source storage system is minimal, as long as the system is configured with sufficient CPU and disk I/O resources.

The first and most important step in asynchronous mode involves the creation of a one-time, baseline transfer of the entire data set. This is required before incremental updates can be performed. This operation proceeds as follows:

1. The source storage system creates a Snapshot copy (a read-only, point-in-time image of the file system). This copy is called the baseline copy.
2. All data blocks referenced by this Snapshot copy and any previous copies are transferred and written to the destination file system.
3. After the initialization is complete, the source and destination file systems have at least one Snapshot copy in common.

After the initialization is complete, scheduled or manually triggered updates can occur. Each update transfers only the new and changed blocks from the source to the destination file system. This operation proceeds as follows:

1. The source storage system creates a Snapshot copy.
2. The new copy is compared to the baseline copy to determine which blocks have changed.
3. The changed blocks are sent to the destination and written to the file system.
4. After the update is complete, both file systems have the new Snapshot copy, which becomes the baseline copy for the next update.

Because asynchronous replication is periodic, SnapMirror is able to consolidate writes and to conserve network bandwidth. There is minimal impact on write throughput and write latency.

## 1.4.2 Synchronous mode

Certain environments may have very strict uptime requirements. All data that is written to one site must be mirrored to a remote site or system synchronously. SnapMirror in synchronous mode is a mode of replication that sends updates from the source to the destination as they occur, rather than according to a predetermined schedule. This guarantees that data written on the source system is protected on the destination even if the entire source system fails. A semi-synchronous mode (discussed in the next section), which minimizes data loss in a disaster while also minimizing the extent to which replication affects the performance of the source system, is also provided.

**Note:** Synchronous and semi-synchronous modes are supported only with VSM.

No additional license fees need to be paid to use this feature; all that is required is appropriate hardware, the correct version of Data ONTAP software, and a normal SnapMirror license for each storage system. Unlike asynchronous mode, which can replicate either volumes or qtrees, synchronous and semi-synchronous modes work only with volumes. Synchronous SnapMirror can have a significant performance impact and is not necessary or appropriate for all applications.

The first step in synchronous replication is a one-time, baseline transfer of the entire data set. This occurs exactly as it does for asynchronous mode described previously. When the baseline transfer is complete, SnapMirror can transition into synchronous mode, as follows:

1. Perform an asynchronous update (at the next minute).
2. Turn on consistency point forwarding.
3. Perform another asynchronous update.
4. When asynchronous update is finished, turn on NVLOG forwarding.

Once SnapMirror has determined that all data acknowledged by the source has been safely stored on the destination, the system is in synchronous mode. At this point, the output of a SnapMirror status query shows that the relationship is in sync. Figure 1-4 illustrates this mode.

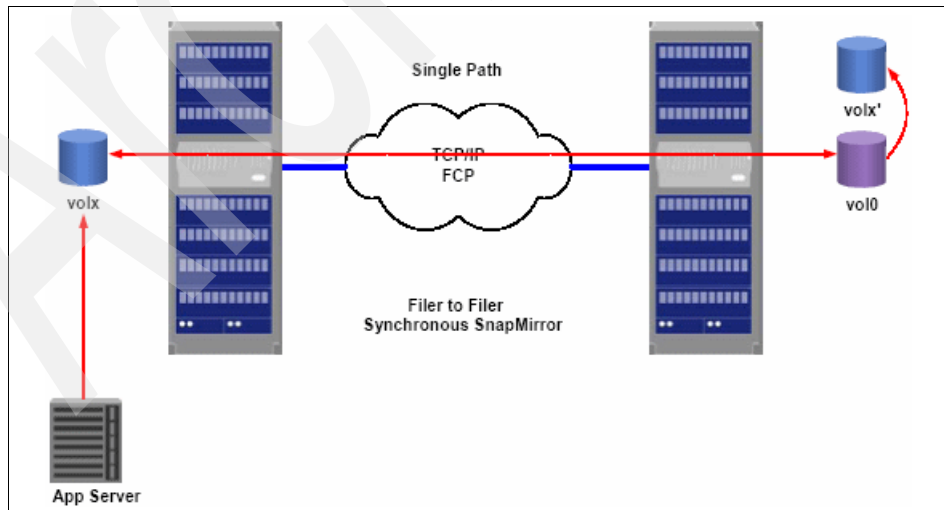


Figure 1-4 Single path synchronous SnapMirror

### 1.4.3 Semi-synchronous mode

Semi-synchronous mode provides a middle ground that keeps the source and destination file systems more closely synchronized than asynchronous mode, but with less impact on application performance than synchronous mode causes. Configuration of semi-synchronous mode is identical to configuration of synchronous mode, with the addition of an option that specifies how many writes/seconds/operations can be outstanding (unacknowledged by the destination system) before the source system delays acknowledging writes from clients. Internally, semi-synchronous mode works identically to synchronous mode in most cases. The only difference lies in how quickly client writes are acknowledged; the replication methods used are the same.

However, it is possible to configure semi-synchronous mode in a way that changes the replication strategy. As discussed in 1.5, “NVLOG (NVRAM Log) forwarding”, a Consistency Point (CP) is triggered when NVRAM is half full, or every 10 seconds, whichever occurs sooner. If semi-synchronous mode is configured to allow unacknowledged transactions older than 10 seconds, SnapMirror falls back to performing CP synchronization only. NVLOG forwarding is halted because CP synchronization is sufficiently frequent to meet the service level requested. When CP synchronization occurs under such circumstances, the data sent to the destination storage system includes not just the list of data blocks to be written, but also the content of those data blocks. This is because with NVLOG forwarding disabled, the destination system does not have a copy of the data until the CP synchronization occurs. Choosing the level of synchronicity is up to the customer. N series storage system provides a dial of performance versus synchronicity, and the user can choose how far to turn that dial. For instance, configuring SnapMirror to allow more than 10 seconds of outstanding data is not recommended for customers who want higher synchronicity levels. However, if NVLOG forwarding is not required, specifying a large time value for outstanding data may reduce the overall CPU usage on the source storage system. This can allow significant increases in overall throughput if CPU usage is a limiting factor.

## 1.5 NVLOG (NVRAM Log) forwarding

NVLOG forwarding is the method used to take write operations submitted from clients against the source file systems to be replicated to the destination. It is a critical component of how synchronous mode works. To completely understand NVLOG forwarding requires a basic knowledge of how Data ONTAP performs local file system writes.



## 1.5.1 Data ONTAP local system writes

The following description assumes an N series storage system running Data ONTAP, ignores interaction with SnapMirror, and for simplicity describes a stand-alone storage system rather than a cluster. It also assumes a functional operating environment where all writes succeed; descriptions of how to handle each possible error condition are outside the scope of this document.

1. The storage system receives a write request. This request could be a file-oriented request from an NFS, or CIFS client, or it could be a block-oriented request via FCP or iSCSI.
2. The request is journaled in battery backed-up, nonvolatile memory (NVRAM). It is also recorded in cache memory, which has faster access times than NVRAM.
3. Once the request is safely stored in NVRAM and cache memory, Data ONTAP acknowledges the write to the client system, and the application that requested the write is free to continue processing. At this point the data has not been written to disk, but is protected from power failure and hardware problems by the NVRAM.
4. Under certain conditions, a CP is triggered. Typically this occurs when the NVRAM journal is half full or when 10 seconds have passed since the most recent CP, whichever comes first (Figure 1-5).

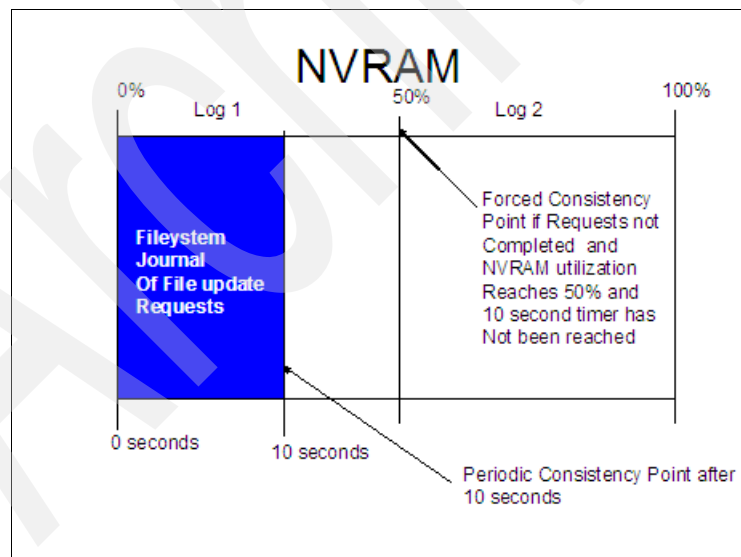


Figure 1-5 NVRAM Log and Consistency Point

5. When a CP is triggered, Data ONTAP uses the transaction data in cache memory to build a list of data block changes that need to be written to disk. It also computes parity information at this time. Once the required disk modifications have been determined, WAFL sends a list of data blocks to be written to the disks.
6. The RAID software writes the data blocks out to disk. When it is complete, it returns an acknowledgment to the WAFL software, and Data ONTAP considers the CP complete.

### 1.5.2 Data ONTAP local system writes with NVLOG forwarding active

When NVLOG forwarding is active in synchronous mode, the steps are modified as follows:

1. The storage system receives a write request. This request could be a file-oriented request from an NFS or CIFS, or it could be a block-oriented request via FCP or iSCSI.
2. The request is journaled in NVRAM. It is also recorded in cache memory and forwarded over the network to the SnapMirror destination system, where it is journaled in NVRAM and cache memory.
3. Once the request is safely stored in NVRAM and cache memory on both the source and destination systems, Data ONTAP acknowledges the write to the client system, and the application that requested the write is free to continue processing.
4. Under certain conditions, a consistency point (CP) is triggered. Typically this occurs when the NVRAM journal is half full or when 10 seconds have passed since the most recent CP, whichever comes first.
5. When a CP is triggered, Data ONTAP uses the transaction data in cache memory to build a list of data block changes that need to be written to disk. It also computes parity information at this time. Once the required disk modifications have been determined, WAFL sends a list of data blocks to be written to the disks.
6. The RAID software writes the data blocks out to disk. When it is complete, it returns an acknowledgment to the WAFL software, and Data ONTAP considers the CP complete.

NVLOG forwarding is the source mechanism by which data is synchronously protected.

One important detail is the way in which the NVLOG data is stored on the destination storage system. Because this system has its own storage as well as the mirrored data from the source system (at the very least, every storage system has its own root volume), and because CPs need to be kept

synchronized on any mirrored volumes, the NVLOG data cannot be stored in NVRAM on the destination system in the same way as normal file system writes. Instead, the NVLOG data is treated as a stream of writes to a pair of special files (replication log files) in the root volume on the destination system. The writes being sent to these files are logged in the destination system's NVRAM just like any other write.

**Note:** Because the NVLOG data needs to be written to these files on the root volume, the performance of the root volume on the destination system has a direct impact on the overall performance of synchronous or semi-synchronous mode SnapMirror.

### 1.5.3 Data ONTAP local system writes with consistency point synchronization (CP Sync) active

Although NVLOG forwarding protects the data by synchronously replicating it from source to destination storage, it functions independently from file system writes to disk on each appliance. CP sync is the underlying mechanism that is used to ensure that the actual on-disk file system images, independent of NVRAM or NVLOG forwarding, are kept synchronized.

When CP synchronization is active (as it always is when in synchronous or semi-synchronous mode), the steps are modified as follows:

1. The storage system receives a write request. This request could be a file-oriented request from an NFS or CIFS client, or it could be a block-oriented request via FCP or iSCSI.
2. The request is journaled in NVRAM. It is also recorded in cache memory and forwarded over the network to the SnapMirror destination system, where it is journaled in NVRAM and cache memory.
3. Once the request is safely stored in NVRAM and cache memory on both the source and destination systems, Data ONTAP acknowledges the write to the client system, and the application that requested the write is free to continue processing.
4. Under certain conditions, a consistency point (CP) is triggered. Typically this occurs when the NVRAM journal is half full or when 10 seconds have passed since the most recent CP, whichever comes first (Figure 1-5 on page 27).
5. When a CP is triggered, Data ONTAP uses the transaction data in cache memory to build a list of data block changes that need to be written to disk. It also computes parity information at this time. Once the required disk modifications are ready, WAFL sends a list of data blocks to be written to the disk. This list of data blocks is also sent across the network to the destination system, which initiates its own write to disk (Figure 1-6).

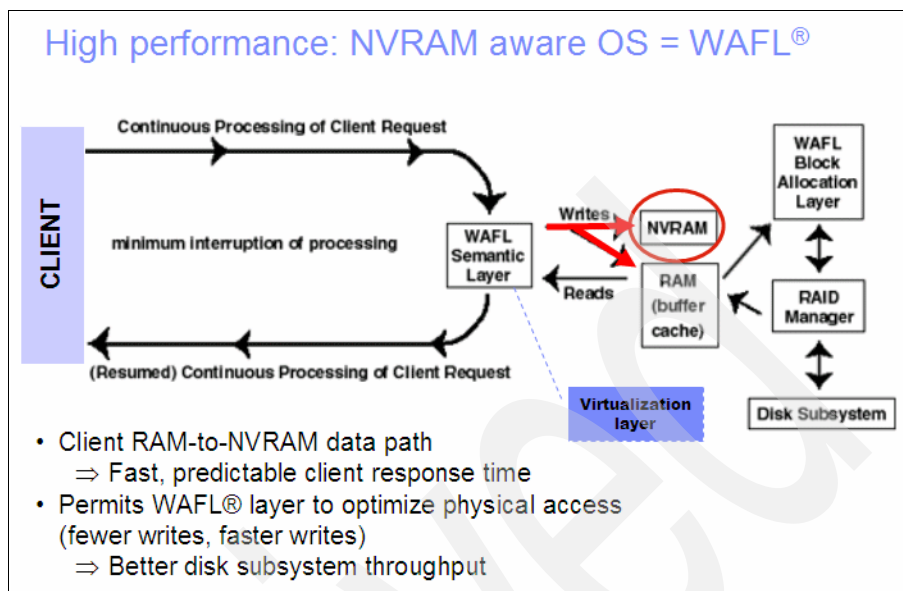


Figure 1-6 WAFL file system

- The local RAID software writes the data blocks out to disk. The destination storage also writes the data blocks out to disk. When each system completes this process, it returns an acknowledgment to the WAFL file system on the source storage system, and Data ONTAP considers the CP complete.

**Note:** If the environment is not able to maintain synchronous mode because of networking or destination issues, SnapMirror drops to asynchronous mode. However, if the environment is unstable because of an unsupported configuration, there is no guarantee that SnapMirror will successfully fall back to asynchronous mode. When the connection is re-established, the source storage system asynchronously replicates data to the destination once each minute until synchronous replication is re-established. When this occurs, a message is logged of the change of state (into or out of synchronous status). This "safety net" is known as fail-safe synchronous.

### 1.5.4 How SnapMirror maintains synchronous state

Synchronous SnapMirror keeps the source and destinations in sync as much as possible, but the following situations can cause it to fall out of sync:

- ▶ The NVLOG channel requests (per operation) time out
- ▶ The CP on the source takes more than one minute

- ▶ The source volume becomes bigger than the destination volume
- ▶ Any network errors persist even after three retransmissions
- ▶ Source or destination reboots, fails, and so on
- ▶ ACK is not received in a sufficient amount of time
- ▶ snapmirror.conf is edited
- ▶ IP address changes

Sync SnapMirror gets in-sync status back by:

- ▶ Performing an asynchronous update (at the next minute)
- ▶ Turning on consistency point forwarding
- ▶ Performing another asynchronous update
- ▶ When asynchronous update is finished, turning on NVLOG forwarding

## 1.6 SnapMirror advanced features

This section provides details about the advanced features of SnapMirror.

### 1.6.1 Multipath support

More than one physical path between a source and a destination system may be desired for a mirror relationship. SnapMirror supports up to two paths for a particular relationship. These paths can be Ethernet, Fibre Channel, or a combination of the two. Multipath support allows traffic to be load-balanced between these paths and provides for failover in the event of a network outage (Figure 1-7). There are two modes of multipath operation:

- ▶ **Multiplexing mode:** Both paths are used simultaneously, spreading data transmissions across the two. When a failure occurs, the load moves to the remaining path.
- ▶ **Failover mode:** One path is specified as the primary path in the configuration file. This path is the desired path and is used until a failure occurs. In the event of a failure, the second path is used. If the primary path becomes available again, it is again used exclusively.

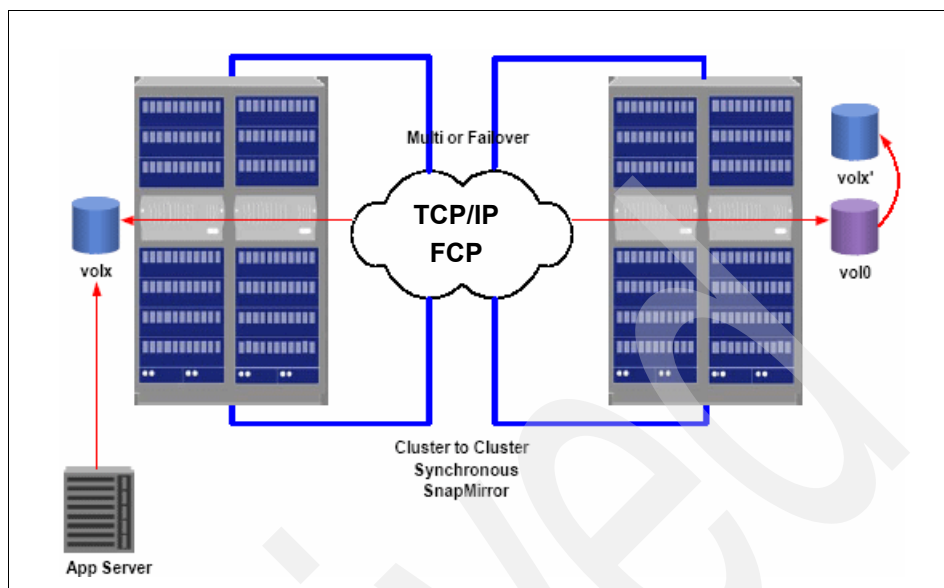


Figure 1-7 Dual path synchronous SnapMirror

## 1.6.2 Visibility interval

With synchronous SnapMirror, there would be an undesirable performance impact in presenting every update that happens on the destination volume to the client as it happens. Updates are shown only after the source storage system creates a Snapshot copy of the source volume. Any Snapshot copy created on the source volume causes the destination volume to present that copy as the visible data. The interval at which these Snapshot copies are created can be adjusted. Visibility interval is a value that can be set by the user and that adjusts the visibility of the updates on the destination. A performance impact may be seen on the source volume if this value is small. It is recommended that, unless a different value is completely necessary, the default value of three minutes be used.

## 1.6.3 Synchronicity level control

This feature controls the semi-synchronous mode of synchronous SnapMirror. The field in the configuration file that controls the synchronicity level is the outstanding argument. By default, if no outstanding value is present, synchronous SnapMirror operates in a fully synchronous manner (the source waits for each transaction to be acknowledged from the destination before moving forward). If there are a large number and frequency of write operations from clients, the systems involved in the relationship can see a performance

impact. You can modify the amount of time (seconds or milliseconds) or the number of operations a destination waits before sending a write acknowledgment. This does mean that the source and destination are out of synchronization between write acknowledgments.

## 1.7 SnapMirror control files

This section provides information about the SnapMirror control files.

### **snapmirror.access**

This option specifies which SnapMirror destination storage systems can initiate transfers and which network interfaces they can use. This is the preferred method for controlling SnapMirror access on a SnapMirror source storage system.

On the source storage system console, use the **options snapmirror.access** command to specify the host names of storage systems that are allowed to copy data directly from the source storage system (Example 1-24).

*Example 1-24 Options snapmirror.access example*

---

```
itsotuc1> options snapmirror.access host=itsotuc2
itsotuc1> options snapmirror.access
snapmirror.access          host=itsotuc2
```

---

The syntax for specifying which storage systems are allowed access to the server is the same for SNMP, telnet, and rsh and is described in the Data ONTAP man pages and in the N series System Storage documentation.

**Note:** If you set the `snapmirror.access` option to `legacy`, the `snapmirror.allow` file is used instead.

### **snapmirror.conf**

This is the core configuration file for all SnapMirror operations. The `/etc/snapmirror.conf` file defines the relationship between the source and the destination, the schedule used by the destination to copy data, and the arguments that control SnapMirror when copying data. This file resides on the destination N series storage system. See Example 1-25 for a sample `/etc/snapmirror.conf` file.

*Example 1-25 /etc/snapmirror.conf file*

---

```
itsotuc1:vol1 itsotuc2:vol2 - 45 10,11,12,13,14,15,16 * 1,2,3,4,5
```

---

In this example the source volume /vol/vol1 on itsotuc1 is replicated to destination volume /vol/vol2 on itsotuc2. The data is replicated at the fastest rate possible; itsotuc2 updates /vol/vol2 at 10:45 a.m., 11:45 a.m., 12:45 p.m., 1:45 p.m., 2:45 p.m., 3:45 p.m., and 4:45 p.m., Monday through Friday. The asterisk (\*) in this example means that the mirror update schedule is not affected by the day of month.

### ***/etc/snapmirror.conf distribution***

You can create a single /etc/snapmirror.conf file for your site and copy it to all the storage systems that use SnapMirror. The /etc/snapmirror.conf file can contain entries pertaining to other storage systems. For example, the /etc/snapmirror.conf file on filerB can contain an entry for copying a volume from filerC to filerD. When filerB reads the /etc/snapmirror.conf file, it ignores the entries for other storage systems. However, each time the file is read, a warning message is displayed on the system console for each line that is ignored.

### ***/etc/snapmirror.conf parameters***

There is no limit on the total number of entries in the /etc/snapmirror.conf file; however, there is a limit of 600 entries for each individual storage system in the /etc/snapmirror.conf file. Entries beyond the entry limit for each storage system are ignored, and a warning message is displayed on the system console.

If a cluster is in place, the limit on the maximum number of entries applies to the storage system pair combination. If one head in a cluster fails, the maximum number is cut in half.

**Note:** This limitation is different from the maximum number of simultaneous replications you can have on a storage system. For details, see 1.10.7, “Maximum number of simultaneous replication operations” on page 43.

### ***Configuration changes***

If SnapMirror is enabled, changes to the /etc/snapmirror.conf file take effect within two minutes. If SnapMirror is not enabled, changes to the /etc/snapmirror.conf file take effect immediately after you enter the **snpmirror on** command to enable SnapMirror.



## 1.8 SnapMirror and tape

SnapMirror to tape (SM2T) gives users the ability to perform the initial full-volume transfer of a SnapMirror volume using tapes. The user can transfer the contents of a source volume to tapes. When the tapes have been created, remote volumes can be “seeded” with the source volume data by shipping these tapes to the remote locations and unloading their contents to the target volumes. After the initial data has been transferred to the target volumes from tapes, subsequent incremental transfers are performed over the network. This feature can significantly reduce the data transfer over the WAN and the amount of time required to initialize a SnapMirror target volume. It can also be used for long-term storage, off-site storage, or potential tertiary protection.

**Note:** SM2T should not be used as a mechanism for backup and recovery for a primary customer environment. It is not meant to be a backup and recovery tool and has no built-in backup features and no enhanced recovery mechanisms.

### 1.8.1 Using SnapMirror for image backups

Mirroring data to tape using SnapMirror is sometimes referred to as image backup. You may want to accomplish this in certain extreme situations. In reality, image backup uses exactly the same algorithm as performing a VSM initialization to a SnapMirror destination, except that the destination is a tape device. The raw volume is opened, ignoring WAFL (Write Anywhere File Layout) file system, and all the blocks in the volume are sent to tape in data block number order. There is no index of what files or Snapshot copies are in the volume because you are looking only at the raw blocks in the volume, not at the file system contained on them. As a result, read-ahead works extremely well because you always know what blocks you are about to ask to read. File size and directory structure have no impact on performance because you are not going through the file system at all.

The end result is performance that is theoretically equivalent to the raw read performance of the spindles in the RAID group. However, blocks are read in order, which means that for traditional volumes, you effectively start with block zero on the first drive in the RAID group, read all of the blocks on that drive, and then move to the next drive. So with traditional volumes, you get the maximum read performance (burst rate) of a single spindle. With flexible volumes, reads are spread across all physical spindles in the aggregate, so maximum read performance is significantly higher. Aggregate throughput approaches linearity, and backup performance is enhanced.

Because you are not going through the actual file system, however, any file system fragmentation or corruption is replicated to the destination system when restored. Also, only a baseline transfer to tape is currently supported, not incrementals. There is no way to get just the changed blocks since the last backup, as you can with SnapMirror update. You must run the entire baseline again. You also get all Snapshot copies in the volume and all live data. There is no way to select a single Snapshot copy where you know that an application or file system was consistent (unlike with an NDMP-based application). Figure 1-8 illustrates image backup.

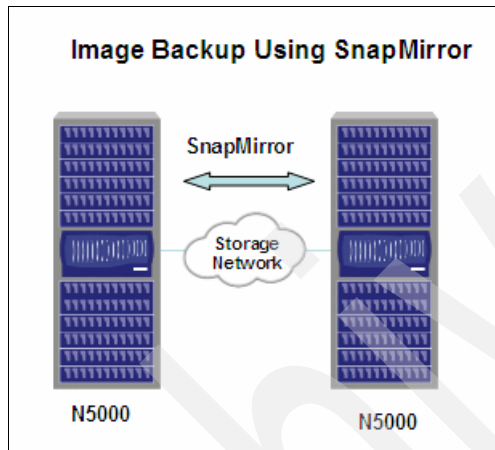


Figure 1-8 Image backup using SnapMirror

Most importantly, it depends entirely on the destination system being able to interpret the RAID or WAFL version written on the tapes. This means that in theory an earlier version of Data ONTAP could be restored to a newer version (just as with SnapMirror) but never to an older version. You also cannot restore a transfer from a traditional volume to a flexible volume or vice versa; the restoration must be to a volume of identical type. If backups are not well documented, this creates issues with long-term retention.

## 1.9 SnapMirror with FlexVol and FlexClone volumes

Flexible volumes are a ground breaking new technology. These volumes are logical data containers that can be sized, resized, managed, and moved independently and non-disruptively from the underlying physical storage. A FlexVol volume can be shrunk or increased on the fly without any downtime. Flexible volumes have all the spindles in the aggregate available to them at all times. For I/O-bound applications, flexible volumes can run much faster than equivalent-sized traditional volumes.

A FlexClone volume is a writable point-in-time image of a FlexVol volume or another FlexClone volume. FlexClone volumes add a new level of agility and efficiency to storage operations. They take only a few seconds to create and are created without interrupting access to the parent FlexVol volume. FlexClone volumes use space very efficiently, leveraging the Data ONTAP architecture to store only data that changes between the parent and clone.

SnapMirror is enhanced and in some instances changed by the introduction of FlexVol and FlexClone volumes.

Only QSM can be used to migrate from traditional volumes to FlexVol volumes. VSM does not work in this scenario. Essentially, the rule that volume replication can occur only with volumes of the same type (that is, both traditional volumes or both flexible volumes) applies here.

### 1.9.1 SnapMirror with FlexVol

When using Flexvol, VSM requires that source and destination volumes to be the same type (traditional to traditional or FlexVol to FlexVol, with the same revision of Data ONTAP). Table 1-2 identifies the supported SnapMirror source and destination volumes.

Table 1-2 Supported SnapMirror source and destination volumes

Source	Destination		
	Traditional VSM	Traditional QSM	FlexVol
Traditional VSM	Yes	No	No
Traditional QSM	No	Yes	Yes
FlexVol	No	No	Yes

Because VSM is block-based, the source and destination need to be the same type: for example, traditional volume to traditional volume or flexible volume to flexible volume. While QSM uses logical replication, it does not have these restrictions.

The size of a FlexVol volume can be changed dynamically. It can also act as a hard quota for a group or project assigned to it. In each volume, user- and group-level quotas as well as qtrees can be used to obtain finer granularity of quota management.

Flexible volumes make disk geometry less of an issue. Destination volumes no longer have to contain the same number of disks or the same size disks as the source volumes, allowing more efficient deployment of resources. With FlexVol

volumes, SnapMirror is no longer bound by physical limitations of copying physical disks block for block. The physical nature of VSM has been virtualized (Figure 1-9).

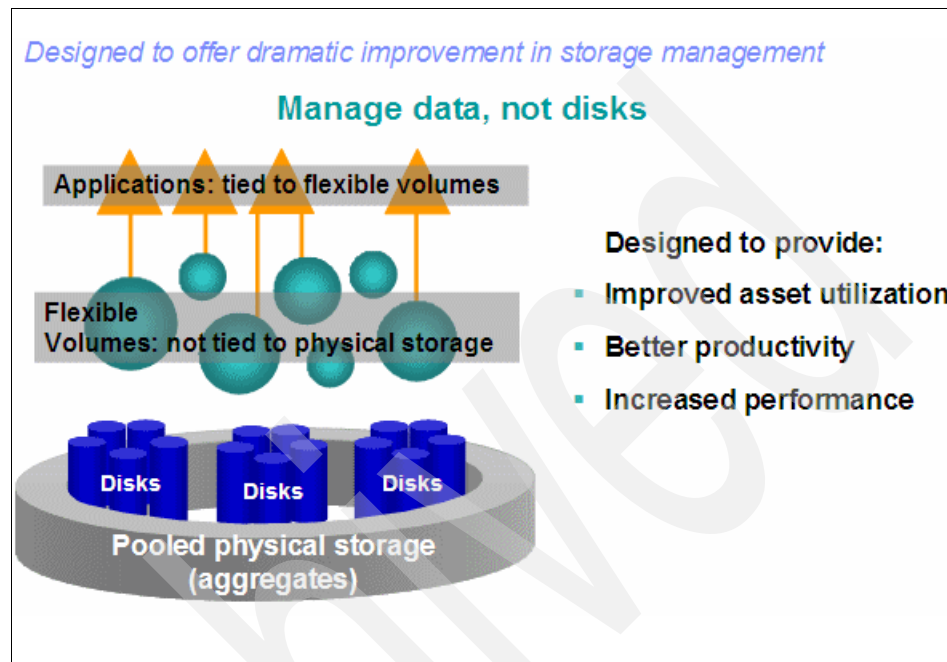


Figure 1-9 FlexVol dynamic virtualization

## 1.9.2 SnapMirror with FlexClone

Cloning offers a nearly instantaneous replica of a volume within the same aggregate. It also offers substantial space savings for work environments that require multiple copies of the same data, such as source trees, chip simulations, and weather simulations, without causing any performance bottlenecks.

A FlexClone volume can be split from its parent to create a new standalone volume. It is also possible to create a writable volume from a read-only SnapMirror destination. Cloning is available only with flexible volumes, not with traditional volumes. Cloning does not require any special hardware (Figure 1-10).

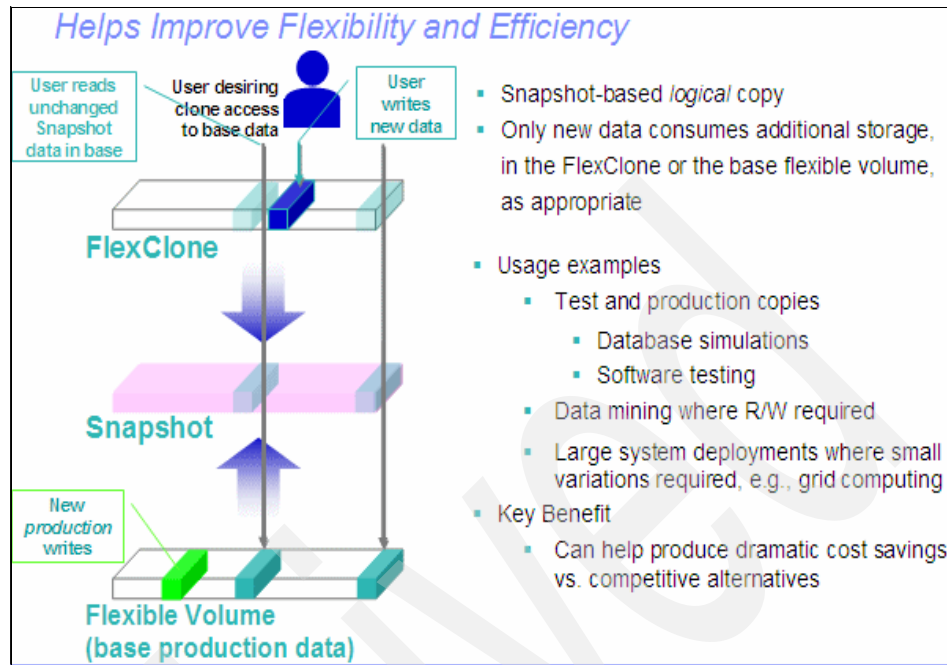


Figure 1-10 FlexClone

Users can clone SnapMirror volumes. Users must be aware that a clone of a SnapMirror destination locks down the Snapshot copy from which the clone was created. In doing so, it also locks down that copy in the source volume, and in every volume in the cascade, if the volume is part of a SnapMirror cascade.

Also, if a FlexClone volume is created from a Snapshot copy in the destination volume that is not the most recent copy, and hence has locked down the Snapshot copy, if that Snapshot copy no longer exists on the source volume, every update needs to delete the copy on the destination. In this case, all SnapMirror updates to the destination volume will fail until the clone is destroyed or split. This does not occur if the clone is created from the most recent Snapshot copy in the SnapMirror destination, because that copy still exists in the source volume.

Splitting a FlexClone volume from its parent removes the connection between the clone and its parent. The administrator can split a FlexClone volume in a SnapMirror environment without affecting the SnapMirror transfer because once the FlexClone volume is split, it becomes an independent entity. In fact, it is a good idea to split clones that have been used for an extended period of time in order to avoid any impact on SnapMirror, especially if the source Snapshot copy could be deleted.

FlexClone data resides in the parent Snapshot copy, so operations that would destroy the parent copy are not allowed. The following operations are not allowed:

- ▶ Destroy parent volume (but it can be taken offline)
- ▶ Destroy parent shared Snapshot copy
- ▶ Use **vol-copy** over a parent volume
- ▶ Use **snapmirror initialize** over a parent volume
- ▶ Use **snaprestore** with a parent volume
- ▶ Use **snapmirror resync** with a parent volume

**Note:** A FlexClone volume cannot be used as a SnapMirror destination.

## 1.10 Using SnapMirror

There are several different uses for SnapMirror depending on environment and requirements.

### 1.10.1 Deleting source Snapshot copy in a SnapMirror deployment

SnapMirror creates Snapshot copies at the beginning of an update and deletes Snapshot copies at the end of the update. Customers often set their SnapMirror schedules so that many updates are initiated at the same time. As a result, SnapMirror may create many simultaneous Snapshot copies on the source volume. If one of these source copies for a SnapMirror relationship is deleted, SnapMirror fails on subsequent update requests. As stated previously, when a FlexClone volume is created, either at the source or at the destination, it is bound to the source copy. Even if the FlexClone volume was created remotely on the destination system, deleting the source copy on the source system results in a failed SnapMirror update when SnapMirror attempts to propagate the deletion to the destination.

### 1.10.2 Synchronous SnapMirror in an FCP environment

The administrator can create a clone volume that already attains writable attributes.

Administrators can use either QSM (LUN cloning) or VSM (FlexClone) if they want writable LUNs on the destination or the ability for clients to map and use LUNs on the destination

### 1.10.3 Space guarantees in a SnapMirror environment

*Guarantees* determine how the aggregate volume pre-allocates space to the flexible volume. SnapMirror never enforces guarantees, regardless of how the source volume is configured. As long as the destination volume is a SnapMirror destination (replica), the guarantee is volume-disabled. Subsequently, when the destination is broken, the guarantee mode is the same as the volume mode.

### 1.10.4 SnapMirror and overcommitting the aggregate

When users require additional space, the administrator can increase the size of an aggregate volume by assigning additional disks to it. In a SnapMirror configuration, overcommitting the aggregate volume allows more efficient use of disk space on the destination. Only the data that is used on the SnapMirror source is used in the FlexVol volume on the SnapMirror destination. If that SnapMirror destination is broken, the disk usage is deducted from the overall aggregate. Unless mirrors are broken, you can have many source volumes of varying sizes all mapped to destination flexible volumes.

To overcommit an aggregate volume, create flexible volumes with a guarantee of none or file so that the volume size is not limited by the aggregate size. The total size of the flexible volumes can be larger than the containing aggregate.

It is recommended that the size of the destination volume be as big as the `vol autosize maxgrowth` setting. This allows uninterrupted mirror transfers. The volume can be grown manually as well.

### 1.10.5 SnapMirror supported volume type and clustering

Table 1-3 displays the supported volume type matrix for asynchronous VSM. Synchronous and semi-synchronous SnapMirror follow the same volume type matrix as VSM, with two exceptions involving FlexVol volumes (traditional volumes do not have any additional configuration issues).

Synchronous and semi-synchronous SnapMirror unsupported configurations are the following:

- ▶ FlexVol → FlexVol when source and destination FlexVol volumes are both on the same clustered system
- ▶ Bidirectional synchronous SnapMirror between FlexVol volumes between two different clustered systems; that is, some relationships are  $A \rightarrow C$ , and some are  $D \rightarrow B$

Table 1-3 General volume type support with asynchronous VSM

Volume type	Support
Traditional volume to traditional volume	Yes
FlexVol to FlexVol	Yes
Traditional volume to FlexVol	No
FlexVol to traditional volume	No

**Note:** A deadlock can be caused when a bidirectional configuration is deployed where a source or destination system is both the SnapMirror source and the SnapMirror destination. This may happen when a cluster failover occurs and one head assumes both roles.

In addition, a deadlock may be encountered if one storage system is sending a CP to another, which causes a CP on the other storage system. If the two storage systems are set up as a bidirectional pair, they would be frozen waiting for each other to finish the CPs.

This is illustrated in Figure 1-11.




Configuration	SnapMirror Relationships	Support
Unidirectional Clustered Pair		Yes
Bidirectional Clustered Pair		No
Same cluster		No

Figure 1-11 SnapMirror relationships supported configuration

### 1.10.6 Logical replication (LREP)

LREP is a set of tools that allows the source site to mimic a SnapMirror LREP stream to a portable device (tape or external drive) and to populate the destination site with the source data set without traversing the network. This is commonly referred to as *seeding the baseline* or *initial transfer*. Thus, it is a good solution for low-bandwidth networks.



Consider a scenario in which you are going to establish a SnapMirror relationship (specifically the initial level-0 transfer) between a source storage system and a destination storage system over a low-bandwidth connection. This scenario assumes the following configuration:

- ▶ A low-bandwidth connection between the source and destination storage systems
- ▶ A local tape drive attached to the source storage system
- ▶ A local tape drive attached to the destination storage system

Incremental Snapshot copy mirroring from the source to the destination over the low-bandwidth connection is feasible, but the initial base Snapshot copy mirroring is not, due to the data size and low-bandwidth connection between the source and destination storage systems. In such a case, it is faster to first transfer the initial base Snapshot copy image from source to destination via tape, and then set up incremental SnapMirror updates to the destination storage system via the low-bandwidth connection. Logical replication (LREP) is the mechanism for accomplishing this task.

**Note:** LREP applies only to QSM.

### 1.10.7 Maximum number of simultaneous replication operations

A VSM, QSM, or SnapVault replication consists of two operations, one on the source side of the transfer and the other on the destination side of the transfer. Therefore, if a storage system is the source of one replication and the destination of another replication, it uses two replication operations. Likewise, if a storage system is the source and the destination of the same replication, it uses two replication operations. (Migrating from traditional volumes to FlexVol volumes with QSM is an example of the same storage system being both a source and a destination.) Table 1-4 shows the maximum number of simultaneous replication operations that each storage system model can support.

*Table 1-4 Maximum simultaneous replication operations per storage system*

	Simultaneous transfers allowed for FC drives	Simultaneous transfers allowed for ATA drives
IBM System Storage N3700	8	8
IBM System Storage N5200	16	8
IBM System Storage N5500	16	8

### ***Factors that might reduce the maximum number of operations***

A storage system might not reach the maximum number of simultaneous replication operations for the following reasons:

- ▶ Storage system resources, such as CPU usage, memory, disk bandwidth, or network bandwidth, are taken away from SnapMirror or SnapVault operations.
- ▶ Each storage system in a cluster has the maximum number of simultaneous replication operations listed in Table 1-4. If a failover occurs, the surviving storage system cannot process more than the maximum number of simultaneous replication operations specified for that storage system. These can be operations that were scheduled for the surviving storage system, the failed over storage system, or both. For example, each N5500 in a cluster can run a maximum of 16 simultaneous replication operations. If one N5500 fails over to the other, it still has a maximum of 16 operations, which can be operations that were scheduled by the surviving N5500, the failed N5500, or both.

**Note:** Take this limitation into consideration when you are planning SnapMirror or SnapVault replications using clusters.

# SnapMirror best practices and behaviors

This chapter discusses the following:

- ▶ SnapMirror best practices and recommendations
- ▶ SnapMirror common behaviors
- ▶ SnapMirror basic troubleshooting guide

## 2.1 General best practices

Taking into account all the details of how SnapMirror works, and assuming that synchronous or asynchronous mode has been selected (including the appropriate level of replication, volume, or qtree), the following best practices should be applied when deploying a SnapMirror replication solution. Although it is possible in most cases to deploy a fully supported configuration that violates one or more of these guidelines, following all of them is strongly recommended to provide the best levels of performance and data protection.

### Adjusting the TCP window size

The *TCP window* is the amount of data that a source can send on a connection before it requires acknowledgment from the destination that the data was received. The default window size for SnapMirror operations is 1,994,752 bytes. This is a large window; in some networks, particularly WANs, it can result in the termination of the SnapMirror transfer or in very low throughput. Decrease the TCP window size to fit your network better and to avoid slow link problems.

### Destination volume requirement

In VSM, the mirrored volume must be as large as or larger than the source volume. Although it is possible to configure a mirrored volume with a different configuration than that of the source volume (for example, different size RAID groups or disk geometries), this can result in suboptimal mirroring performance.

In addition to VSM, you must create a restricted volume to be used as the destination volume. SnapMirror does not automatically create a volume.

In QSM, the destination volume must contain 5% more free space than the source qtree consumes. In addition to that, destination qtree names cannot consist of certain wildcards and special characters. There is also a 64-character limit on the length of names. A qtree name must be present and appended to the full volume path. The full qtree path must also be preceded with /vol before the volume name that the qtree resides in. For non qtree data, a dash (-) is specified in place of the qtree name. This replicates data not in a qtree but in the volume.

A destination volume in VSM and QSM must not consist of the root volume.

### Source volume requirement

The source volume must be in online status and cannot be a mirrored volume, with the exception of a cascaded VSM mirror.

In QSM, the SnapMirror source volume can be the root volume.

## Network configuration

If a dedicated network cannot be used, SnapMirror transfers must mix on the network with other network traffic. Using a private network for SnapMirror is recommended because it can relieve the pressure that SnapMirror puts on a network. It is also recommended to use the throttle option, which can be configured.

## Firewall configuration

SnapMirror uses the typical socket/bind/listen/accept sequence on a TCP socket. SnapMirror source binds on port 10566. The destination storage system contacts the SnapMirror source storage system at port 10566 using any of the permissible, available ports assigned by the system. A firewall set in the customer scenario must allow requests to this port of the SnapMirror source storage system. Allowing a range of TCP ports from 10565 to 10569 is recommended.

## Memory configuration

SnapMirror can run into problems if a destination storage system has much less memory than the source machine. The problem comes when the transfer has completed and the destination storage system attempts to bring the volume online. Make sure that the memory requirements for the destination storage system are sufficient for the volume you will bring online on the destination storage system.

## Semi-synchronous mode

Use semi-synchronous mode if at all possible, to reduce the latency impact to applications.

## Write throughput

Because synchronous mode (and semi-synchronous mode with outstanding values of nine seconds or less) requires the source storage system to do approximately twice as much work as a configuration without mirroring, destination systems should be configured to support approximately twice the raw write throughput that applications are expected to need.

## Platforms

In synchronous modes, the destination storage system should be the same storage system model as the source, or a higher-end storage system.

## **Root volume**

In synchronous modes, the destination storage system should have at least four data disks (the number varies by platform) in the root volume unless using an outstanding interval of greater than 10 seconds.

## **Disk size**

In synchronous modes, the disk sizes must be the same on both the source and destination systems. This is a requirement to have a supported configuration; unequal disk sizes can cause serious performance degradation.

## **Synchronous mode and applications**

In synchronous replication environments, the destination storage system should not be used for applications; it should be dedicated to providing service as a synchronous or semi-synchronous SnapMirror destination.

## **Many-to-one configuration**

Replication of two or more source systems to a single destination system should be done only if the destination storage system can service the combined write throughput requirements of all source systems.

## **Outstanding interval**

In semi-synchronous mode the tolerance for outstanding writes should be set to nine seconds unless there are specific reasons to use another value.

- ▶ One possible exception would be changing the value to 11 seconds to reduce CPU impact on the source storage system, potentially increasing overall throughput.
- ▶ Another possible exception would be to reduce the value to a much smaller one, based on business requirements with low tolerance for lost data. This results in increased application-visible latency.

## **Synchronous SnapMirror and storage system performance**

The synchronous mode of SnapMirror impacts the performance of a storage system. This impact depends on the power of your storage system, its write load, and the number of simultaneous synchronous SnapMirror relationships. To keep this impact under control, synchronous SnapMirror functionality requires that the user not have more than eight simultaneous synchronous SnapMirror operations.

## **Snapshot copies**

For smoothest performance, schedule SnapMirror Snapshot copy and update times (in the etc/snapmirror.conf file) to be different from the time of any regular

system Snapshot copies (set through the **snap\_sched** command) on the source storage system.

### **Scheduling and Snapshot copies**

Avoid scheduling a Snapshot copy at the same time that SnapMirror is transferring the data in the copy. SnapMirror locks Snapshot copies before they are transferred; therefore, if SnapMirror is transferring data in a Snapshot copy when an automatic copy is scheduled, the automatic copy is not created. Overlapping VSM transfers can cause Snapshot copies to not get deleted at the usual times because one transfer has a Snapshot copy locked while the other is trying to delete it.

### **Unicode**

Directories on all SnapMirror source volumes that support CIFS clients must be in Unicode format before being replicated to a destination; otherwise, the directories in the read-only destination volume will not be in Unicode format and attempts through CIFS to access directories and open files on the destination volume may receive “access denied” errors. This is true in QSM deployments only.

## **2.2 Best practices for deploying for performance**

Follow these best practices to achieve the highest performance:

- ▶ Stagger SnapMirror schedules for transfers to reduce impact to resources on the target. For example, if four transfers are required every hour, space the start times 15 minutes apart.
- ▶ Throttle bandwidth on transfers in the `/etc/snapmirror.conf` file with the `Kbs` argument:
  - Default settings result in no throttling of transfers.
  - Throttling is especially important in a high-speed LAN with many mirror relationships.
- ▶ When possible, schedule transfers at different times than scheduled Snapshot copies.

## 2.3 Known SnapMirror behaviors

When evaluating your SnapMirror implementation, it is important to consider the following common SnapMirror behaviors and to understand when and why they may occur.

### Clustered failover interaction

SnapMirror complements N series storage system clustered failover (CF) technology by providing an additional level of recoverability. If a catastrophe disables access to a clustered pair of storage systems, one or more SnapMirror volumes can immediately be accessed in read-only mode while recovery takes place. If read-write access is required, the mirrored volume can be converted to a writable volume while the recovery takes place. If SnapMirror is actively updating data when a takeover or giveback operation is initiated, the update aborts. Following completion of the takeover or giveback operation, SnapMirror continues as before. No specific additional steps are required for the implementation of SnapMirror in a clustered failover environment. For detailed information on N series storage system clustered failover technology, and takeover and giveback scenarios, see the Data ONTAP System Administrator's Guide.

### Open files

If SnapMirror is in the middle of a transfer and encounters an incomplete file (a file that an FTP server is still transferring into that volume or qtree), it transfers the partial file to the destination. Snapshot copies behave in the same way. A Snapshot copy of the source would show the transferring file and would show the partial file on the destination.

A workaround for this situation is to copy a file to the source. When the file is complete on the source, rename the source file to the correct name. This way the partial file has an incorrect name, and the complete file has the correct name.

### Adding disks to SnapMirror environments

When adding disks to volumes in a SnapMirror environment, always complete the addition of disks to the destination storage system or volume before attempting to add disks to the source volume.

**Note:** The `df` command does not immediately reflect the disk or disks added to the SnapMirror volume until after the first SnapMirror update following the disk additions.



## Cascading SnapMirror

A variation on the basic SnapMirror deployment and function involves mirroring from established mirrors to more SnapMirror destinations. The function of this deployment is to make a uniform set of data available on a read-only basis to users from various locations throughout a network and to allow updating that data uniformly at regular intervals. However, cascading a SnapMirror replication from A to B to C and so on is allowed only via VSM. QSM does not allow this behavior. In order to perform a QSM, mapping takes place between source inodes and destination inodes. For example, suppose that the `/vol/vol1/qt7/home/users/darrin/email.txt` file has an inode number 456. When this qtree is transferred to the destination using QSM (such as `vol1_rpl`), the `/vol/vol1_rpl/qt7_rpl/home/users/darrin/email.txt` file may have an inode number 5987432. To be able to apply a modification on number 456 to number 5987432, QSM needs to keep a map of the inodes. Mapping the inodes is necessary because QSM is taking qtrees from different volumes and mirroring them into one common volume. Files from those qtrees may have the same inode number (because they come from different volumes or storage systems). Therefore, QSM reallocates the inodes, so that it does not have problems with conflicts in numbering. In addition, this inode mapping would cause problems because mapping the state can become confusing in a cascade, so this configuration is not allowed in a cascade configuration. Going from A to B to C is straightforward. Simply map A to B and B to C. With a VSM cascade, you can then map A to C. See Figure 2-1 for cascading SnapMirror supported configuration.

























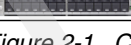
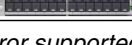
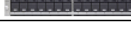
Configuration of SnapMirror Relationships					Support
	→ Sync VSM →		→ Async VSM →		Yes
	→ Sync VSM →		→ Sync VSM →		No
	→ Sync VSM →		→ Async QSM →		No
	→ Async VSM →		→ Async VSM →		Yes
	→ Async VSM →		→ Sync VSM →		No
	→ Async VSM →		→ Async QSM →		Yes
	→ Async QSM →		→ Async VSM →		Yes
	→ Async QSM →		→ Sync VSM →		No
	→ Async QSM →		→ Async QSM →		No

Figure 2-1 Cascading SnapMirror supported configuration

## SnapMirror logging

The SnapMirror log file (located in /etc/logs/snapmirror.log) records the start and end of an update as well as other significant SnapMirror events. It records whether the transfer finished successfully or whether it failed for some reason. If there is a problem with updates, it is often useful to look at the log file to see what happened since the last successful update. Because the log file is kept on the source and destination storage systems, quite often the source or the destination system may log the failure, and the other partner knows only that there was a failure. For this reason, you should look at both the source and the destination log file to get the most information about a failure. The log file contains the start and end time of each transfer, along with the amount of data transferred. It can be useful to look back and see the amount of data needed to make the update and the amount of time the updates take.

**Note:** The time versus data sent is not an accurate measure of the network bandwidth because the transfer is not constantly sending data.

## 2.4 SnapMirror troubleshooting

When you are troubleshooting any scenario that involves SnapMirror, it is essential to obtain the following information and to contact IBM technical support immediately (**autosupport** does not automatically send SnapMirror log files):

- ▶ Which SnapMirror type: VSM or QSM
- ▶ Which mode: asynchronous, synchronous, or semi-synchronous
- ▶ Source storage system information:
  - Data ONTAP version
  - Model
  - System ID
- ▶ Destination storage system information:
  - Data ONTAP version
  - Model
  - System ID (to get information from autosupport)
- ▶ SnapMirror log files on the source and destination storage systems
- ▶ snapmirror.conf file on the destination
- ▶ Intent of the deployment
- ▶ Expected outcome
- ▶ Commands entered and their results

- ▶ **snapmirror status** output (source and destination)
- ▶ **snap list** on source and destination
- ▶ For performance issues, you can run these commands from advanced privilege level:
  - **perf report**  
Obtain output from source and destination (this adds statit and sysstat output as well). See Example 2-1.

---

*Example 2-1 perf report output*

---

```
itsotuc1*> perf report -t
Perf Report Version 1
Samples: 424
Frequency: 1
cdnum1: 359 36 4 0 0 0 0 0 0 0
cdnum2: 4 1 3 0 0 0 0 0 0 0
cdnum3: 0 1 0 0 0 1 0 0 0 0
cdnum4: 1 1 0 0 0 0 0 0 0 0
disk KB: 13008 4224 17052
disk ops: 1990 244 1515
```

---

- **statit**  
This command provides detailed performance statistics about the system, both on the source and on the destination (Example 2-2).

---

*Example 2-2 Statit command*

---

```
To begin statit command:
itsotuc1*> statit -b
To end statit command and capture the performance:
itsotuc1*> statit -e
Hostname:itsotuc1 ID:0099922126 Memory:128 MB
Data ONTAP Release 7.1X17: Mon Aug 8 02:50:45 PDT 2005(IBM)
Start time: Sun Mar 26 09:21:23 GMT 2006
CPU Statistics
3.929652 time (seconds)          100%
0.105077 system time             3%
0.000393 rupt time               0%(393 rupts x1 usec/rupt)
0.104684 non-rupt system time    3%
3.824575 idle time              97%
0.190066 time in CP              5%
0.000019 rupt time in CP         0%(19 rupts x 1 usec/rupt)
Miscellaneous Statistics (per second)
764.70 hard context switches    0.00 NFS operations
0.00 CIFS operations            0.00 HTTP operations
```

0.00 NetCache URLs	0.00 streaming packets						
0.25 network KB received	0.25 network KB transmitted						
20.36 disk KB read	32.57 disk KB written						
0.25 NVRAM KB written	0.00 nolog KB written						
0.00 WAFL bufs given to clients	0.00 checksum cache hits(0%)						
0.00 no checksum - partial buffer	0.00 DAFS operations						
0.00 FCP operations	0.00 iSCSI operations						
WAFL Statistics (per second)							
0.00 name cache hits (0%)	0.00 name cache misses(0%)						
0.25 inode cache hits(100%)	0.00 inode cache misses(0%)						
10.69 buf cache hits(100%)	0.00 buf cache misses(0%)						
0.00 blocks speculative read-ahead	7.38 blocks written						
1.02 stripes written	0.00 blocks over-written						
0.25 waf_l_timer generated CP	0.00 snapshot generated CP						
0.00 low datavecs generated CP	37.92 non-restart messages						
0.00 IOWAIT suspends	17062 buffers						
RAID Statistics (per second)							
5.09 xors	0.00 long dispatches [0]						
0.00 long consumed [0]	0.00 long consumed hipri [0]						
0.00 long low priority [0]	0.00 long high priority [0]						
0.00 long monitor tics [0]	0.00 long monitor clears [0]						
0.00 long dispatches [1]	0.00 long consumed [1]						
0.00 long consumed hipri [1]	0.00 long low priority [1]						
0.00 long high priority [1]	0.00 long monitor tics [1]						
0.00 long monitor clears [1]	13 max batch						
1.53 blocked mode xor	0.25 timed mode xor						
0.00 fast adjustments	0.00 slow adjustments						
0 avg batch start	0 avg stripe/msec						
1.53 tetrises written	0.00 master tetrises						
0.00 slave tetrises	5.09 stripes written						
4.33 partial stripes	0.76 full stripes						
7.89 blocks written	0.00 blocks read						
3.05 1 blocks per stripe size							
1.27 2 blocks per stripe size							
0.76 3 blocks per stripe size							
Network Interface Statistics (per second)							
iface	side	bytes	packets	mcasts	errors	collisions	pkt drops
ns0	recv	354.23	5.60	0.00	0.00	0.00	
xmit		261.09	3.56	0.00	0.00	0.00	
ns1	recv	0.00	0.00	0.00	0.00	0.00	
xmit		0.00	0.00	0.00	0.00	0.00	
vh	recv	0.00	0.00	0.00	0.00	0.00	
xmit		0.00	0.00	0.00	0.00	0.00	

```

Disk Statistics (per second)
ut% is the percent of time the disk was busy.
xfers is the number of data-transfer commands issued per second.
xfers = ureads + writes + cpreads + greads + gwrites
chain is the average number of 4K blocks per command.
usecs is the average disk round-trip time per 4K block.
disk    ut%  xfers  ureads--chain-usecs  writes--chain-usecs
cpreads-chain-usecs  greads--chain-usecs  gwrites-chain-usecs
/aggr0/plex0/rg0:
v0.16   5   2.54   0.51   1.00 55000   1.78   1.71 6667   0.25
4.00 10000   0.00   ....   .   0.00   ....   .
v0.17   2   0.76   0.00   ....   .   0.51   5.50 4545   0.25
7.00 2857   0.00   ....   .   0.00   ....   .
v0.18   3   1.27   0.00   ....   .   1.02   2.25 15556  0.25
4.00 5000   0.00   ....   .   0.00   ....   .
/aggr1/plex0/rg0:
v1.16   0   0.00   0.00   ....   .   0.00   ....   .   0.00
....   .   0.00   ....   .   0.00   ....   .
v1.17   0   0.00   0.00   ....   .   0.00   ....   .   0.00
....   .   0.00   ....   .   0.00   ....   .
v1.18   0   0.00   0.00   ....   .   0.00   ....   .   0.00
....   .   0.00   ....   .   0.00   ....   .
v1.19   0   0.00   0.00   ....   .   0.00   ....   .   0.00
....   .   0.00   ....   .   0.00   ....   .

Aggregate statistics:
Minimum  0   0.00  0.00   0.00  0.00  0.00  0.00  0.00
Mean     1   0.51  0.00   0.25  0.00  0.00  0.00
Maximum  5   2.54  0.51   1.78  0.25  0.00  0.00

Spares and other disks:
v1.20   0   0.00   0.00   ....   .   0.00   ....   .
0.00   ....   .   0.00   ....   .   0.00   ....   .

Spares and other disks:
v1.21   0   0.00   0.00   ....   .   0.00   ....   .
0.00   ....   .   0.00   ....   .   0.00   ....   .

FCP Statistics (per second)
309746870.68 FCP Bytes rec

```

#### – **sysstat**

Use this command to obtain output while the transfer is going on, both on the source and on the destination.

Example 2-3 sysstat command

---

itsotuc1*> sysstat										
CPU	NFS	CIFS	HTTP	Net kB/s		Disk kB/s		Tape kB/s		Cache
				in	out	read	write	read	write	age
1%	0	0	0	0	0	11	17	0	0	8
1%	0	0	0	0	0	7	9	0	0	8
4%	0	3	0	7	1	27	64	0	0	8
1%	0	0	0	0	0	8	10	0	0	9
0%	0	0	0	0	0	22	17	0	0	0

---

- Network details such as other jobs, bandwidth, failures, expected throughput, throttling in place, and so forth

## 2.4.1 Common problem scenarios

### Appears to be a problem with the volume

During an initialization, the destination volume must be in restricted state. The **snapmirror update** command and scheduled updates will not proceed until the **snapmirror initialize** process has completed.

For VSM, the destination volume must be the same size as or larger than the source. Use the **vol status -b** command on the source and destination to determine the relative sizes.

### Problems and errors on the source storage system

This section identifies some common problems originating from the source storage system. Solutions are provided for some of these issues.

#### ***Destination request while SnapMirror is off on the source filer***

Issue **snapmirror on** command on the source.

#### ***Destination request while SnapMirror not licensed on the source filer***

SnapMirror is not licensed on the source storage system.

#### ***Destination request with incorrect version***

The Data ONTAP version is different between the source and destination storage systems. Both the source and destination storage systems must be running the same version of Data ONTAP.

The destination of a VSM relationship must run a version of Data ONTAP that is equal to or more recent than the source. In addition, the source and destination must be on the same Data ONTAP release.

### ***Request from X denied, not listed in /etc/snapmirror.allow***

The destination machine is not mentioned (by hostname) in the snapmirror.allow file. Make sure that it is listed and that it matches the output of the hostname command on the destination storage system.

**Note:** The hostname in the snapmirror.allow file is required even when the mirror source and destination volumes are on the same storage system.

### ***Source volume requested offline or nonexistent volume***

The destination storage system requested a transfer from a volume that does not exist on the source storage system. Check the volume list on the source and the name of the volume on the destination volume.

**Note:** /vol is not part of the volume name. The `vol status` command gives the volume names.

### ***Cannot create incremental Snapshot copy***

A Snapshot copy cannot be created on the source storage system. This can happen if a Snapshot copy is locked due to another transfer in progress. Determine the reason for not being able to create a Snapshot copy on the source storage system.

## **Problems and errors on the destination storage system**

This section identifies some common problems originating from the destination storage system. Solutions are provided for some of these issues.

### ***Unknown filer***

The destination storage system is attempting to find the source machine but is unable to determine the machine based on the name. Check the /etc/hosts file to make sure it is listed there.

### ***Cannot contact source filer***

SnapMirror attempted to but could not create a connection to the source storage system. Check the network connection. Ping the source storage system (by the name listed in the snapmirror.conf file) from the destination. If that doesn't work, fix that problem before proceeding.

### ***Denied by source filer***

The source machine must approve the transfer before the transfer can occur. This message comes out on the destination storage system when the source denied the request for a transfer. The source storage system console gives a more specific message that describes the reason in greater detail. Most often this situation results because the destination storage system hostname is not listed in the source storage system's `snapmirror.allow` file or in the options of `snapmirror.access`. Check the message on the source storage system and investigate that.

### ***Destination volume too small***

The destination volume must be the same size as or larger than the source volume. Unfortunately, the `df` command does not display the exact size of the volume; the output of `df` takes the snap reserve into account. The `vol status -r` command gives a more accurate size of the volume (although it is not completely accurate).

### ***Transfer aborted from source***

The source storage system aborted the transfer for some reason. Check the error messages displayed on the source storage system.

### ***There is insufficient memory on this system***

Trouble can occur if a destination storage system has much less memory than the source machine. The problem comes when the transfer has completed, the destination storage system attempts to bring the volume online, and there is insufficient memory to bring the volume online. Storage systems have memory requirements, and this applies to the destination storage system as well. To work around this problem, make sure that the memory requirements for the destination storage system are sufficient for the volume you are bringing online on the destination storage system.

## **Failed updates**

If a transfer fails for any reason, SnapMirror attempts a retry of the transfer immediately, not waiting for the next scheduled mirror time. These retry attempts continue until they are successful, until the appropriate entry in the `/etc/snapmirror.conf` file is commented out, or until SnapMirror is turned off. Some events that can cause failed transfers include:

- ▶ Loss of network connectivity
- ▶ Source storage system is unavailable
- ▶ Source volume is offline



## **SnapMirror timeouts**

There are three situations that can cause a SnapMirror timeout:

### ***Write socket timeout***

If the TCP buffers are full and the writing application cannot hand off data to TCP within 10 minutes, a write socket timeout occurs. Following the timeout, SnapMirror resumes at the next scheduled update.

### ***Read socket timeout***

If the TCP socket that is receiving data has not received any data from the application within 30 minutes, it generates a timeout. Following the timeout, SnapMirror resumes at the next scheduled update. By providing a larger timeout value for the read socket timeout, you can be assured that SnapMirror will not time out while waiting for the source file to create Snapshot copies, even when dealing with extremely large volumes. Socket timeout values are not tunable in the Data ONTAP and SnapMirror environment.

### ***Sync timeouts***

These timeouts occur in synchronous deployments only. If an event occurs that causes a synchronous deployment to revert to asynchronous mode, such as a network outage, no ACK is received from the destination system.



# **SnapMirror integration with other N series storage system features**

This chapter discusses the following:

- ▶ DataFabric Manager
- ▶ SnapLock
- ▶ SnapVault
- ▶ SnapDrive

## 3.1 DataFabric Manager

The DataFabric Manager (DFM) application delivers comprehensive monitoring and management for IBM System Storage N series. From a central point of control, DFM provides alerts, reports, and configuration tools to keep your storage and content delivery infrastructure in line with your business requirements for maximum availability and reduced total cost of ownership (TCO). DataFabric Manager allows an organization to rapidly deploy, provision, and manage a complete enterprise storage and content delivery network.

The following capabilities are currently available in DFM for managing a SnapMirror environment:

- ▶ Create and manage asynchronous and synchronous SnapMirror relationships
- ▶ Manage policy for replication and failover
- ▶ Report on relationships and lag times
- ▶ Alert on replication state changes
- ▶ Schedule replica updates
- ▶ Visualize relationships
- ▶ Simplify recovery of data services after failure

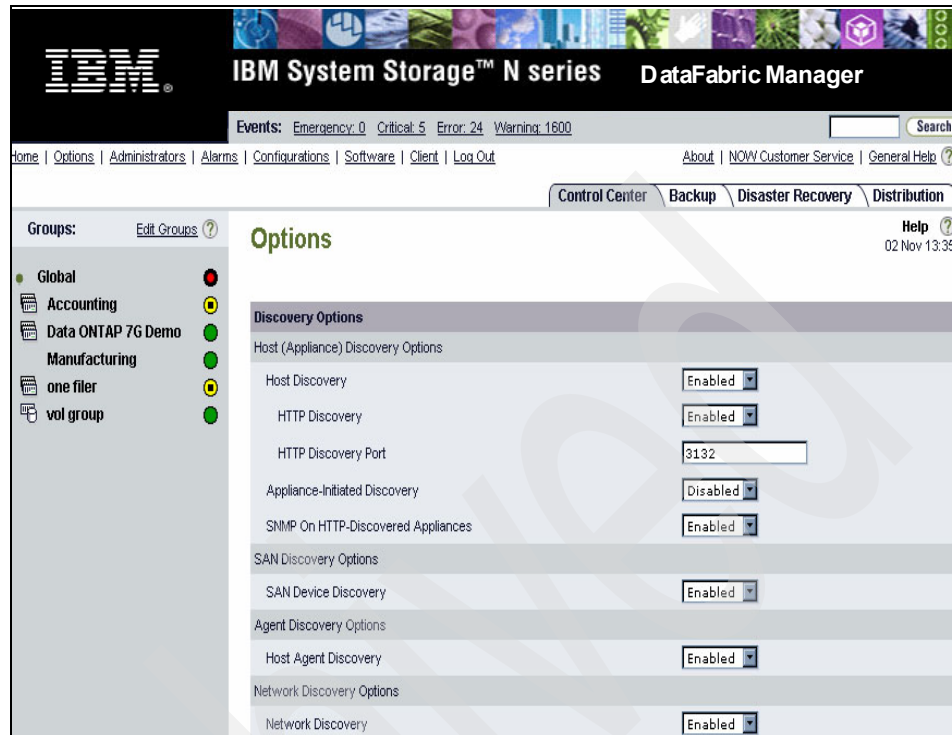


Figure 3-1 The DFM control center

There are two main items on the Options menu that relate to the DFM Disaster Recovery Manager. First, the Business Continuity Option must be licensed, as shown under Licensed Features. The only other option that can be customized to a user's needs is SnapMirror Monitoring Interval, which specifies how often DFM checks to make sure that the SnapMirror relationship is working.

### 3.1.1 Creating a SnapMirror relationship with DFM

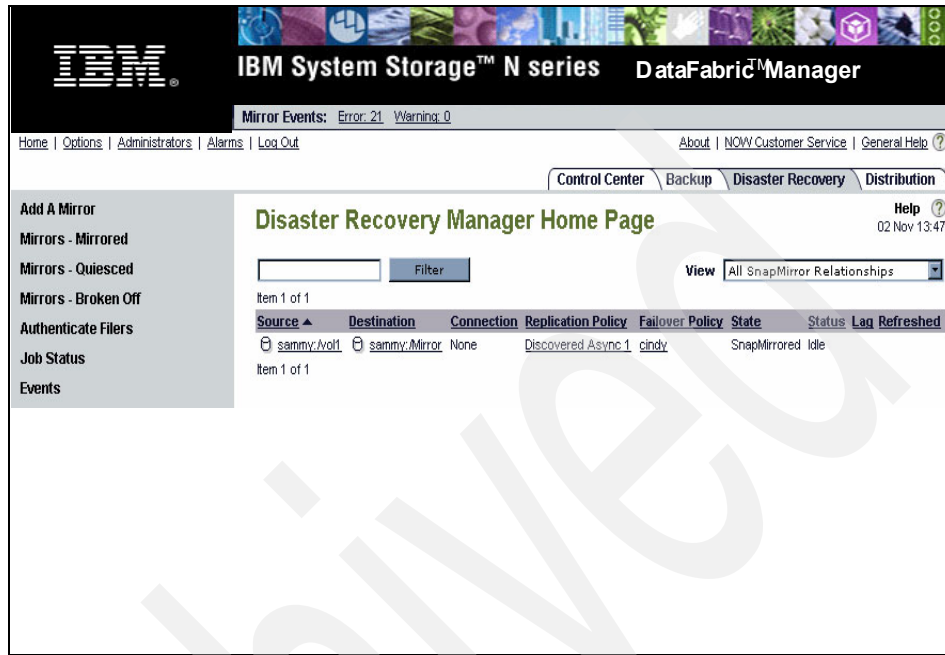


Figure 3-2 Creating a SnapMirror relationship

Before using asynchronous or synchronous replication, you must first create a SnapMirror relationship that identifying the source, the destination, and the policies governing the relationship. Use the following procedures to do this:

1. Select the Disaster Recovery tab and click the **Add a Mirror** option at the left. On the panel that is returned, define the characteristics of the new relationship as follows:
  - At the top, select whether this is a qtrees or a VSM relationship.
  - Under the Source section, either select an appliance from the drop-down list or click the icon at the far right to add a new source. If you select the new source icon, enter the new source appliance name and the NDMP username and password for authentication in the panel returned.
2. After you have selected or entered the pertinent information, click **Add**.
3. Return to the SnapMirror Relationships screen and specify the destination in the same way that you specified the source.

**Note:** Before setting up the source and destination, you must have NDMP enabled and configured on each of the affected appliances.

### 3.1.2 Managing SnapMirror relationships with DFM

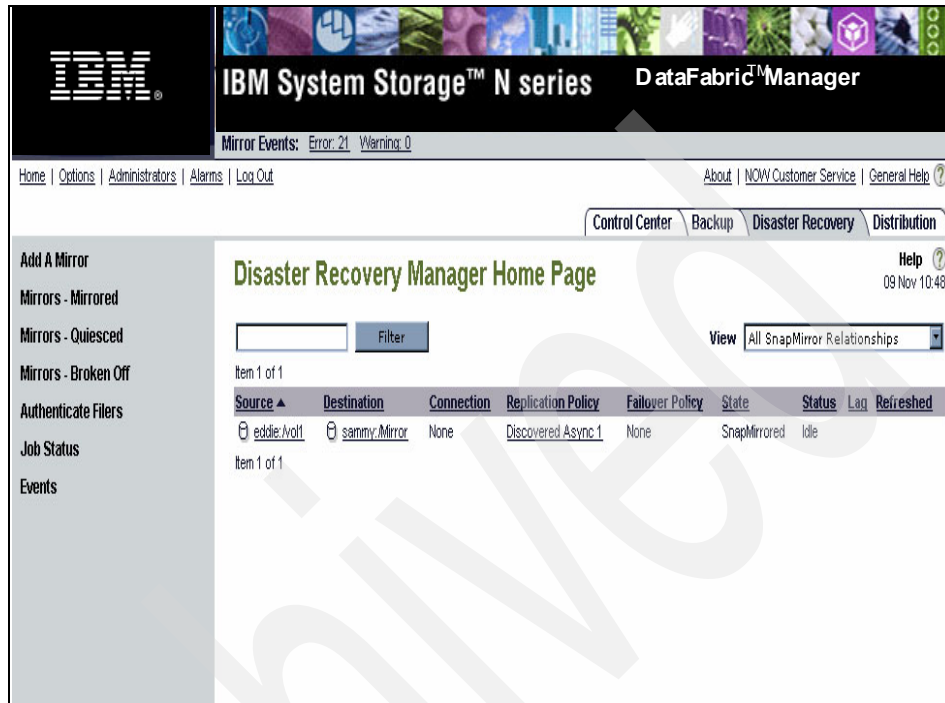


Figure 3-3 Managing SnapMirror relationship

After you have created a SnapMirror relationship and have learned about monitoring and event management for the jobs, you can manage these relationships from a central console.

From the Disaster Recovery manager home page, you can make the following selections from the left-hand navigation pane:

- ▶ **Mirrors - Mirrored** - This option lets you view all currently running relationships and select one or more to be quiesced, updated, or broken off.
- ▶ **Mirrors - Quiesced** - This option displays all relationships that have been quiesced. From here you can either resume or break off chosen mirrors.
- ▶ **Mirrors - Broken Off** - This option displays all broken mirrors. From here you can either resync or delete mirrors completely.
- ▶ **Authenticate Filers** - This option lets you ensure that all affected storage systems are properly set up for authentication.

### 3.1.3 Managing SnapMirror connections with DFM

**Connections** Help ? 02 Nov 14:40

**Add a New Connection**

Connection Name: new

Connection Mode: Multi

Filers:

Source: eddie.PMLAB.ibm.com

Destination: sammy.PMLAB.ibm.com

Address Pair 1:

Source: 1.1.1.1

Destination: 1.1.1.1

Address Pair 2:

Source: 1.1.1.2

Destination: 1.1.1.2

Add

Figure 3-4 Managing connections

In a SnapMirror relationship, you can optionally establish a predefined connection that allows customers to specify one or two network paths between a source and a destination appliance. This provides more bandwidth and some networking failover capability.

A *connection* defines an alternate name for the source storage system that can be used as the source storage system in a relationship. Connections are used to describe the parameters for the connections between the source and destination storage systems. SnapMirror supports the multipath specification for synchronous mirrors only.

Each connection specifies a connection name, one or two pairs of IP addresses, and a mode of operation for the connection.

- Name is the name of the connection to be defined. This name is used as the source storage system in relationship definitions.



- ▶ Mode is optional and specifies the mode in which two IP address pairs are used:
  - The multiplexing mode causes SnapMirror to use both paths at the same time. If one fails, it switches to use the remaining path only. When the failing path is repaired, both paths are used again.
  - Failover mode causes SnapMirror to use the first path as the desired path and to use the second path only if problems arise with the first path.
- ▶ The addresses are resolvable network names or IP addresses that define a path through the network between the source and the destination. The source addresses are the IP addresses of interfaces to use on the source and respectively for the destination. The pairing denotes a path from source to destination.

### 3.1.4 Managing SnapMirror policy with DFM

**SnapMirror Relationships** Help ? 02 Nov 14:40

**Add a New SnapMirror Relationship**

Relationship Type: ☐ Volume ☒ Qtrees

Source:

Filer: eddie.PMLAB.ibm.com Manage... Refresh...

Volume: eddie:/Source Manage... Refresh...

Qtrees: None Available

Destination:

Filer: sammy.PMLAB.ibm.com Manage... Refresh...

Volume: sammy:/Backup Manage... Refresh...

Qtrees:

Connection:

Connection Name: None Available Manage...

Policies:

Replication Policy: Discovered Async 1 Manage... Refresh...

Failover Policy: cindy Manage... Refresh...

**Add**

Figure 3-5 Add a new SnapMirror relationship

## Policies

Policies are central to the SnapMirror management service. A *policy* is a collection of configuration settings that customers can apply to multiple SnapMirror relationships. When a policy is updated, all SnapMirror relationships

sharing the policy are updated at once. A policy can be shared by multiple relationships, so policies are important tools to help customers manage large numbers of SnapMirror relationships.

## Replication policies

A replication policy affects how data is replicated from source to destination. There are two types of replication policies:

- ▶ **Asynchronous:** Data changed at the source is transferred in batches, at specific time intervals. When the timer for a relationship on an asynchronous replication policy goes off, SnapMirror packs up all data that has changed since the last transfer and brings the destination up to date with it. The destination falls furthest behind the source just before the next update. If the source is down, data loss is limited to that changed since the last update. An asynchronous replication policy can perform updates as frequently as once a minute.
- ▶ **Synchronous:** Data changed at the source is continuously propagated to the destination. With this setting, the destination is in sync with the source at any point in time. A disaster at the source causes no loss of data because the destination has an exact copy of it. There is also a semi-synchronous mode, in which a destination can be allowed to lag the source between 0 and 60 seconds.

An asynchronous replication policy has the following parameters:

- ▶ **Schedule:** Specifies when automatic update is triggered.
- ▶ **Throttle:** Specifies the bandwidth limit for any data transfer in kilobits per second.
- ▶ **Restart:** Specifies how SnapMirror attempts to restart if a previous transfer was aborted in the middle and is restartable. The values for the argument are Always, Never, and Default.

A synchronous replication policy has the following parameters:

- ▶ **Semi-synchronous:** The amount of data or time the destination is allowed to fall behind the source. The value is between 0 and 60000 ms (60s), or between 0 and 500 write operations.
- ▶ **Visibility interval:** A time interval at the end of which the accumulated data throughout the period is made visible to storage system clients via CIFS or NFS.

## Failover policies

A failover policy affects how SnapMirror breaks and resyncs should behave. There is only one setting in a failover policy: the path to a user-installed script

that is called before and after a SnapMirror break, and before and after a SnapMirror resync.

**Note:** See the `dfdm` man page for more details.

### 3.1.5 Monitoring SnapMirror with DFM

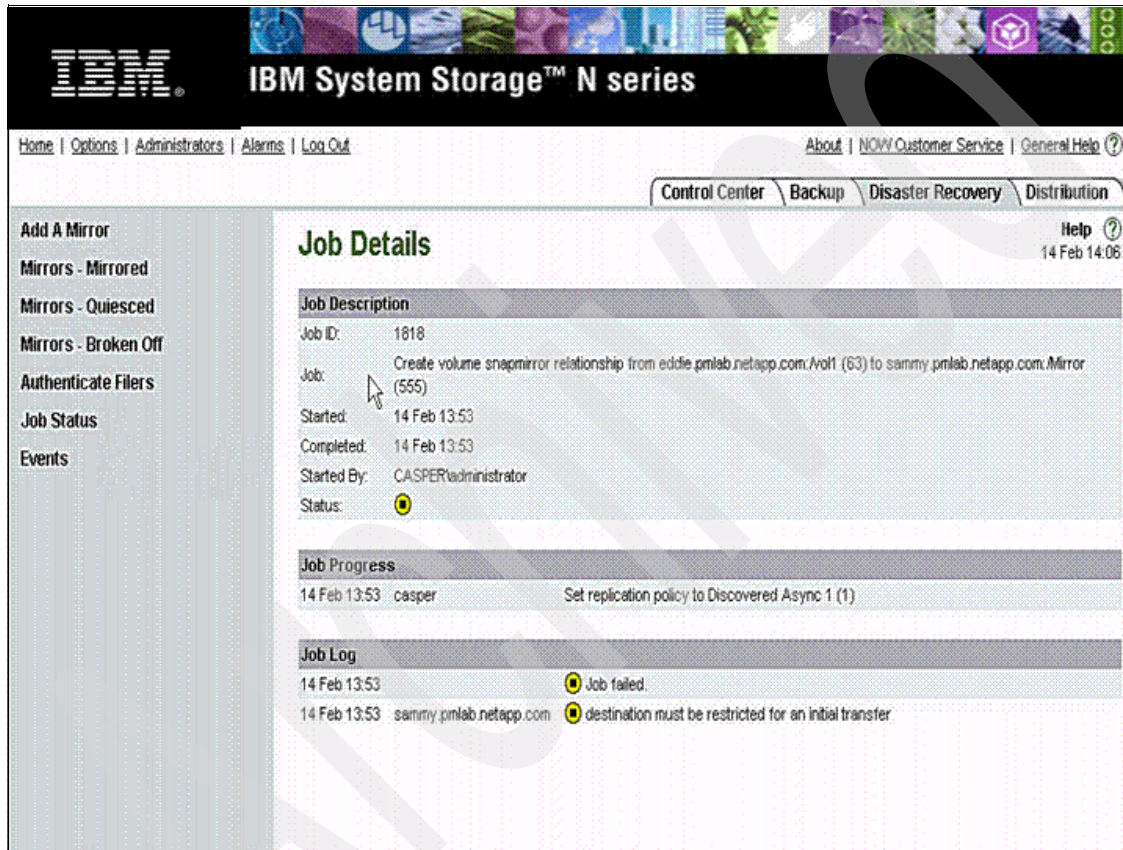


Figure 3-6 Monitoring DFM

Enabling the creation and management of replication relationships is only part of the DataFabric Manager capabilities. It can also monitor the jobs, report on both failed and successful replication relationships, and provide detailed information (Figure 3-6).

The Disaster Recovery Manager also creates events that can be used to notify the appropriate personnel for corrective action. To view events created with the same filtering capability for jobs, click **Events** on the left side of the screen.

Along with the other features of DFM, the capability it gives you to monitor jobs and create events for notification produces a much more efficient and proactive replication environment, with the result that your data will be current and available when you need it during a disaster.

### 3.1.6 Reviewing SnapMirror with DFM

Central control of the creation and management of asynchronous and synchronous SnapMirror relationships improves efficiency and availability. IT policies for replication and failover can be established and enforced based on needs and business requirements. Reporting on relationships, including visualization and lag times, allows the documentation of all replication jobs and ensures that data is kept up to date. Proactively using DFM to manage, configure, report, and troubleshoot assists in problem resolution before these problems become critical.

## 3.2 SnapLock

SnapLock is a software driver secure mechanism for data archiving. It helps ensure the permanence, accuracy, integrity, and security of data by enabling business records to be both unalterable and rapidly accessible online for long periods of time. SnapLock is an integrated feature in Data ONTAP that uses Write Once, Read Many (WORM) volumes, which you create for data that you want to archive permanently.

A WORM volume possesses the property that data can only be written once to any area of the media and can never be overwritten or erased. In some cases this is a property of the physical media (such as with WORM optical platters) and in other cases the physical media is rewritable but the integrated hardware and software codes controlling access to the media prevent such overwrites (such as with WORM magnetic tape and disk-based SnapLock).

In regulated environments the regulating body, such as the SEC, normally requires that all business records that fall under the umbrella of the regulations be archived on WORM media in order to maintain a non-erasable and non-rewritable electronic paper trail that can be used for the purposes of discovery or investigation (Figure 3-7).

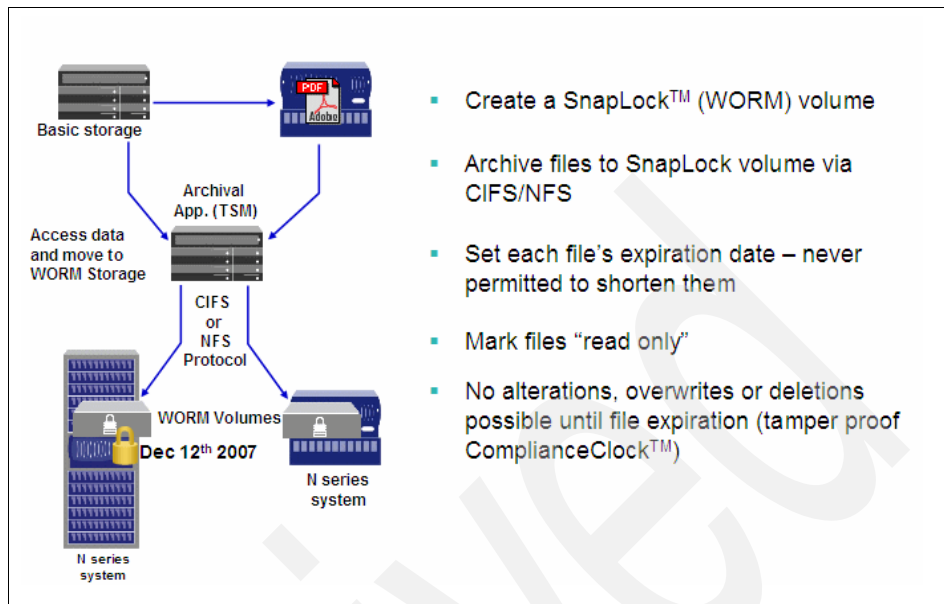


Figure 3-7 WORM Data Protection with SnapLock

There are two types of SnapLock volumes:

- ▶ SnapLock Compliance (SLC™) volume: Software-driven secure clock mechanism that is intended for strict regulatory environments, such as SEC 17a-4 compliant environments.

A SnapLock Compliance volume cannot be destroyed and has default, minimum, and maximum retention periods. WORM file retention dates can be extended and expired WORM files can be deleted.

- ▶ SnapLock Enterprise volume: For environments without regulatory restrictions.

Data ONTAP supports both SnapLock and non-SnapLock destination volumes or qtrees.

One very critical restriction is that you cannot use the SnapMirror resync feature to reestablish a VSM relationship to a SnapLock Compliance destination volume because this operation would result in the destruction of WORM data on the destination volume and would make SnapLock non-compliant with SEC regulations regarding non-erasability of records. This important consideration must be kept in mind while planning DR scenarios for SnapLock Compliance volumes.

Also note that there is no such restriction on resync in the case of SnapLock Enterprise volumes because the administrator is trusted.

The other main difference in behavior for SnapLock Compliance volumes is that to establish a VSM relationship to a SnapLock Compliance destination volume, the destination volume must initially *not* be a SnapLock volume. An extra option to the mirror initialization command causes the destination volume to be converted to a SnapLock volume once the initial level 0 transfer has completed and the mirror relationship is established. This extra footwork is not required in the case of SnapLock Compliance qtree destinations or SnapLock Enterprise destination volumes.

### 3.2.1 Extracting data from SnapLock dump file

After a QSM resync, any new data created on the destination SLC qtree is archived into a dump file and stored in the /etc directory in the SnapLock volume (Example 3-1).

*Example 3-1 SnapLock Volume archive*

---

```
/mnt/etc/log/snapmirror_resync_archive/slcsec_1374c60e-44ba-11d9-9991-0050560669 e7_qd
```

---

To later extract the data from the dump file, perform the following steps:

1. Using a UNIX or Windows client, create a directory called temp in the SnapLock volume; or create a new directory called temp and copy the dump file into this directory, giving the file the new name dump file. Although this is not necessary, it makes running the restore command much easier because the leading path information from the dump file's original location is long.
2. To view files contained in the dump file, run the following command:  

```
restore -tf /vol/volume_name/temp/dumpfile
```
3. To restore files contained in the dump file to their original location, run the following command:  

```
restore -rfQ /vol/volume_name/temp/dumpfile
```
4. To restore files contained in the dump file to a different location, such as the temp directory where the dump file resides, run this command:  

```
Restore -rfQD /vol/volume_name/temp/dumpfile /vol/volume_name/temp
```

Extracted files will be in their original SnapLock state, regardless of the approach used.
5. If it is desirable to migrate the dump file to a different appliance, use two UNIX or Windows clients to transfer the file with a utility such as ftp.

### 3.2.2 Regular volume to SnapLock compliance volume

You cannot mirror a regular volume to a SnapLock Compliance volume, and you cannot mirror a SnapLock Enterprise volume to a SnapLock Compliance volume. You can mirror a regular volume to a SnapLock Enterprise volume. However, SnapMirror does not automatically do WORM commits for files transferred in such a configuration. To do that, you must break the mirror and perform the WORM commit on all transferred files, either manually or by using a script. The reverse situations have no restrictions—you can mirror a SnapLock volume (Compliance or Enterprise) to a volume other than SnapLock by using SnapMirror.

### 3.2.3 End-to-end SnapLock Compliance VSM relationship

In order to create an end-to-end SnapLock Compliance VSM relationship, the destination volume must initially be a regular volume other than SnapLock. Use the `snapmirror initialize` command to initialize the mirror.

This does not apply to SnapLock Compliance QSM relationships, which are initialized just as would be done with regular volumes.

### 3.2.4 Synchronous SnapMirror and SnapLock volumes

SnapLock Compliance does not yet support synchronous SnapMirror, but it will in a future release. However, SnapLock Enterprise does support synchronous SnapMirror.

## 3.3 SnapVault

SnapVault provides a centralized disk-based backup solution for heterogeneous storage environments. Storing backup data in multiple Snapshot copies on the SnapVault secondary storage system lets enterprises keep weeks of backups online for faster restoration. SnapVault also gives users the power to choose which data gets backed up, the frequency of backup, and how long the backup copies are retained.

SnapVault and SnapMirror often work hand in hand. Following are some examples of this tight integration, and Figure 3-8 illustrates it graphically.

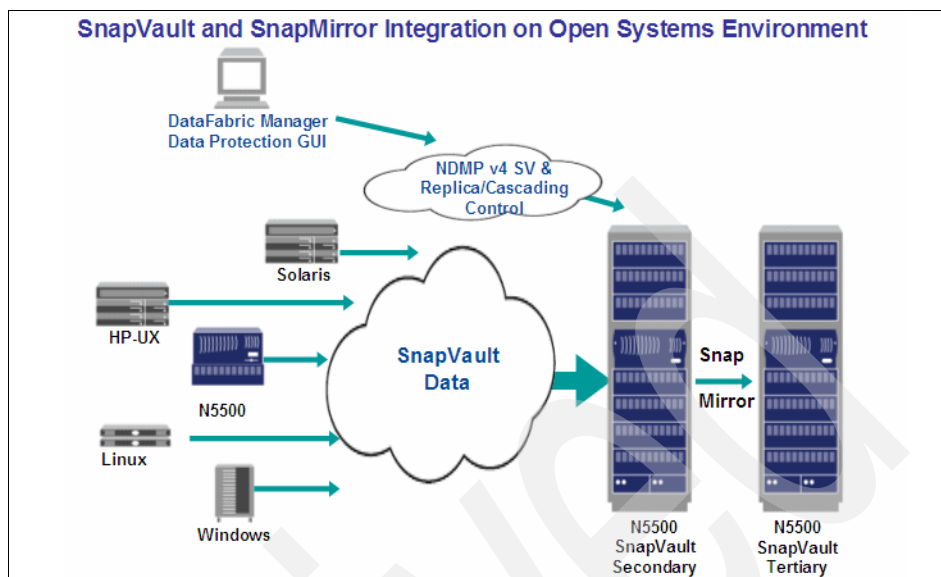


Figure 3-8 SnapVault and SnapMirror integration

### 3.3.1 SnapMirror interlock with SnapVault

When using SnapMirror in combination with SnapVault, it is important to understand their relationships with Snapshot. Because SnapMirror and SnapVault both use the same Snapshot copies, they cannot run simultaneously. Schedules must be managed to accommodate the interlock that keeps SnapMirror and SnapVault from stepping on each other.

### 3.3.2 Using SnapMirror to create DR for SnapVault

SnapVault does not currently have the ability to create a writable destination on the secondary system. However, you can use SnapMirror to convert the SnapVault destination to a SnapMirror destination, making it a typical SnapMirror destination that can be quiesced and broken.

#### Licensing

Both SnapVault primary and SnapVault secondary systems must be licensed on the secondary system. SnapMirror must be licensed on the secondary system for open systems failover and to make the secondary volume writable.



## Converting and making secondary Read/Write

Perform the following steps to convert an Open Systems SnapVault (SSV) or SnapVault secondary backup destination to a usable/writable destination (typically for DR situations). All of these steps are done *on the secondary*:

1. Turn SnapMirror and SnapVault off.
2. Switch to privileged mode with this command:  
`priv set diag`
3. Convert SnapVault qtree to SnapMirror qtree using the command:  
`snapmirror convert <sec_qtree_path>`
4. Turn SnapMirror on.
5. Quiesce the qtree.
6. Break the mirror, making it writable.
7. Turn SnapVault on.

## Re-establishing the relationship

The following steps apply only to storage system-to-storage system SnapVault. Because OSSV doesn't consist of a primary running Data ONTAP, these steps are not used in an OSSV relationship.

In an OSSV situation, if the primary is brought online later or restored from the secondary system, a baseline would be required. OSSV resync is a future consideration. To re-establish the storage system-to-storage system SnapVault relationship, there are two scenarios.

### Scenario 1

Use the following steps to preserve all the changes made to the secondary during the DR period.

1. Primary: Resync the primary qtree with command:  
`snapmirror resync <pri_qtree_path>`
2. Primary: Quiesce the qtree with command:  
`quiesce <pri_qtree_path>`
3. Primary: Break the mirror, making it writable.
4. Secondary: Resync the secondary qtree with command:  
`snapmirror resync <sec_qtree_path>`
5. Secondary: Turn SnapMirror and SnapVault off.

6. Secondary: Convert SnapMirror qtree to SnapVault qtree with command:  
`snapvault convert <sec_qtree_path>`
7. Secondary: Turn SnapVault and SnapMirror on.

### **Scenario 2**

Use the following steps to discard all the changes made to the secondary during this period (the period of time that you are in DR mode).

1. Secondary: Resync the secondary qtree with command:  
`snapmirror resync <sec_qtree_path>`
2. Secondary: Turn SnapMirror and SnapVault off.
3. Secondary: Convert SnapMirror qtree to SnapVault qtree with command:  
`snapvault convert <sec_qtree_path>`
4. Secondary: Turn SnapVault and SnapMirror on.

Storage system-to-storage system SnapVault can now update the qtree as if no changes had occurred.

## **3.3.3 Protecting SnapVault secondaries with volume-based SnapMirror**

You can protect against disasters on the SnapVault secondary system using SnapMirror software. The configuration involves setting up SnapMirror relationships from the volumes on your SnapVault secondary system to volumes on a remote (tertiary) Data ONTAP system. SnapMirror thus provides an exact replica of the SnapVault secondary data on the tertiary system.

Additionally, *softlock* support enables you to continue SnapVault replication relationships between the original SnapVault primary system and the tertiary system, without any initial baseline transfers. For instance, if your SnapVault secondary system becomes unusable because of a disaster, you can manually redirect your next SnapVault transfers to the tertiary system instead of to the old SnapVault secondary system. Effectively, your tertiary system becomes the new SnapVault secondary system, and your SnapVault transfers continue using the latest Snapshot copy that is common to both the primary and the tertiary systems.

### 3.3.4 Using SnapMirror to migrate SnapVault

To migrate a volume that contains SnapVault destination qtrees from one secondary system to another secondary system without having to perform another baseline transfer, complete the following steps:

1. Ensure that you have SnapVault baselines of the qtrees or directories that you are migrating.

Example: In this procedure, assume that a baseline of the bno:C:\500MB directory was backed up to N3700-old:/vol/old\_vol/bno\_C\_500MB.

2. Using SnapMirror, replicate the volume from the present secondary system to a volume on the new secondary system.

**Note:** This is a SnapMirror replication of a volume, not a qtree.

To replicate the old\_vol volume from the N3700-old secondary system to the new\_vol volume on the N3700-new secondary system, complete the following steps on the *new* secondary system (N3700-new):

- a. Create the new\_vol volume by issuing the following command:

```
N3700-new> vol create new_vol 3
```

- b. Mark the new\_vol volume restricted by issuing this command:

```
N3700-new> vol restrict new_vol
```

- c. Transfer the old\_vol volume to the new\_vol volume by issuing the following command:

```
N3700-new> Snapmirror initialize -S r200-old:old_vol new_vol
```

3. Quiesce and break the SnapMirror relationship between the old secondary system and the new secondary system.

Example: To quiesce and break the SnapMirror relationship between N3700-old and N3700-new, issue the following commands on N3700-new:

```
snapmirror quiesce new_vol  
snapmirror break new_vol
```

4. Run **snapmirror status** and **snapvault status** on the new secondary (Example 3-2). SnapMirror status should be Broken-off. SnapVault status should be snapvaulted to the new volume on the new secondary system.

*Example 3-2 Check SnapMirror and SnapVault status*

---

```
N3700-new> snapmirror status  
Source Destination State  
N3700-old:old_vol N3700-new:new_vol Broken-off
```

```
N3700-new> snapvault status
Source Destination State
bno:C:\500MB N3700-new:/vol/new_vol/bno_C_500MB Snapvaulted
```

---

5. Confirm that SnapVault configuration information is not present on the new secondary system by using the **snapvault status -c** command from N3700-new as follows:  

```
N3700-new> snapvault status -c
Snapvault secondary is ON.
```
6. Add SnapVault configuration information to the registry on the new secondary system using the **snapvault start** command from N3700-new (Example 3-3).

**Note:** This does not start a new baseline; it updates the registry.

---

*Example 3-3 Start SnapVault*

---

```
N3700-new> snapvault start -S bno:C:\500MB
N3700-new:/vol/new_vol/bno_C_500MB
Snapvault configuration for the qtree has been set.
Qtree /vol/new_vol/bno_C_500MB is already a replica.
```

---

7. Confirm that SnapVault configuration information is present on the new secondary system by using the **snapvault status -c** command from N3700-new (Example 3-4).

---

*Example 3-4 Check SnapVault status*

---

```
N3700-new> snapvault status -c
Snapvault secondary is ON.
/vol/new_vol/bno_C_500MB source=bno:C:\500MB
```

---

8. Test the new SnapVault relationship by manually updating N3700-new. If you are using the command line to manage your environment, continue to the next step; otherwise, you are finished. Perform the step in Example 3-5 from N3700-new.

---

*Example 3-5 Test new SnapVault relationship*

---

```
N3700-new> snapvault update N3700-new:/vol/new_vol/bno_C_500MB
Transfer started.
```

---

Monitor progress with **snapvault status** or the **snapmirror log** command.

9. Re-create any schedules used on the old secondary system to the new secondary system and ensure that access permissions are in place.

### 3.3.5 Using DataFabric Manager

To migrate a volume from one secondary to another secondary using DataFabric Manager, complete the following steps:

1. Complete the command line procedure before using the DataFabric Manager interface for SnapVault management of the new secondary system.
2. Ensure that NDMP is configured properly on the new secondary system, that is, ndmpd is turned on and the ndmpd password is set up.
3. Import the new relationships to the new volume and secondary system and reestablish schedules and retention policies to the new secondary system.
4. Remove the old relationships to the old volume and old secondary system.

### 3.3.6 Using SnapVault to protect a VSM destination

SnapVault is often used to protect a VSM destination. Note that this is supported, you are not cascading QSM at this point. Also note that while SnapVault is updating, the VSM cannot update (this applies only to dump).

Also, if VSM needs to delete a Snapshot copy on the destination volume (because that Snapshot copy was deleted on the VSM source volume), and SnapVault still has that Snapshot copy locked (because a backup relationship depends on it), the normal expected behavior is for the VSM transfer to fail.

However, this should never happen if SnapVault is set up correctly. Any softlocks that SnapVault sets should be propagated to the VSM source, so those Snapshot copies should never get deleted on the VSM source volume until SnapVault no longer needs them. Therefore, the VSM destination would never need to delete them while they're locked, and would never fail.

Otherwise, the configuration is actually simpler than typical SnapVault. You configure only the schedule on the SnapVault secondary, and it simply takes the most recent Snapshot copy of the VSM as the Snapshot copy to transfer (similar to a VSM cascade).

### 3.3.7 Key differences between SnapVault and SnapMirror (QSM)

Some of the key differences between SnapVault software and the qtree-based SnapMirror feature are identified in Table 3-1.

Table 3-1 SnapMirror (QSM) and SnapVault comparison

SnapMirror (QSM)	SnapVault
Required in immediate failover capable environment.	Does not require immediate failover capability.
Mainly used to protect local offices.	Mainly used to protect remote offices.
Can run in the same N series storage system.	Cannot run in the same N series storage system.
Use the same software and licensing on the source and destination system.	Use different software and licensing on the source and destination system, which is client and server.
Transfers can be scheduled as frequently as once per minute.	Transfers can be scheduled once per hour.
Snapshot copies are deleted when they are no longer used for replication purposes.	Snapshot copies are retained and deleted on a specified schedule.
Direction between source and destination can be reversed.	Direction between server and client cannot be reversed.
Can only be used on storage system running Data ONTAP.	Can be used on both N series or Open Systems source storage.

## 3.4 SnapDrive

SnapDrive supports the use of SnapMirror at the volume level only, it does not support qtree-level SnapMirror operations. To use SnapDrive in conjunction with SnapMirror, your system must meet the following requirements:

- ▶ SnapMirror must be licensed on the source and destination storage systems.
- ▶ Depending on the virtual disk protocols you are using, you must enable the iSCSI and FCP licenses on both the source and destination storage systems.
- ▶ You must manually create and initialize a mirror between the source and destination volumes, but you must not create a SnapMirror replication schedule.

When setting up SnapMirror on your storage system, avoid conflicts with SnapDrive by setting the replication schedule on the storage system to “- - -”, which disables any scheduled transfers. When you set the replication schedule, make sure that the destination volume is in a restricted state.

- ▶ You must create your SnapMirror relationship using storage system names, not IP addresses.
- ▶ The system must contain one or more SnapMirror source volumes hosting virtual disks.
- ▶ The system must contain one or more SnapMirror destination volumes for each source volume.
- ▶ The destination volume must be at least as large as the source volume.
- ▶ The destination volume must have CIFS shares identical to those on the source volume.
- ▶ You can create CIFS shares manually for the volumes on the destination storage system. The data paths for these shares must match exactly those on the source volumes. See your Data ONTAP Data Protection Online Backup and Recovery Guide for details.
- ▶ The Windows domain account used by the SnapDrive service must be a member of the local BUILTIN\administrators group on both the source and destination storage systems.
- ▶ The Windows domain account used to administer SnapDrive must have full access to the Windows domain to which both the source and destination storage systems belong.
- ▶ The source and destination storage systems must be configured to grant root access to the Windows domain account used by the SnapDrive service. That is, the `waf1_map_nt_admin_to_root` option must be set to on. For information about enabling storage system options, see your Data ONTAP documentation.
- ▶ If you want to use a Windows host to access the replicated LUNs on the destination volume, the destination storage system must have at least one LUN access protocol licensed (iSCSI or FCP).
- ▶ A TCP/IP connection must exist between the source storage system and the destination storage system.
- ▶ The SnapDrive service can perform only one task at a time. Therefore, if you are scheduling multiple tasks on a host, make sure that you do not schedule these tasks to start at exactly the same time. If multiple tasks are scheduled at the same time, only the first one will succeed and the others will fail.

### 3.4.1 Replication upon SnapDrive Snapshot copy creation

Each time a Snapshot copy of a virtual disk is created—manually or because of a Snapshot copy schedule—SnapDrive determines whether the virtual disk from which the Snapshot copy was created resides on a SnapMirror source volume. If so, after the Snapshot copy has been created, SnapDrive sends a SnapMirror

update request to all the destination volumes associated with the source volume for that virtual disk.

When you initiate a Snapshot copy of a LUN on a SnapMirror source through SnapDrive, a window with a check box labeled Initiate SnapMirror Update is displayed. The check box is selected by default.

### 3.4.2 Replication using rolling Snapshot copies and SnapDrive

You can also create a special type of Snapshot copy, called a *rolling Snapshot copy*, using the SnapMirror GUI. These Snapshot copies are used exclusively to facilitate frequent SnapMirror volume replication. Like regular Snapshot copies, rolling Snapshot copies are replicated to the SnapMirror destination volume as soon as they are created.

SnapDrive creates a new rolling Snapshot copy every time you initiate a mirror update operation (by using the Update Mirror option in the Action menu) for a specific virtual disk drive residing on a SnapMirror source volume.

To guarantee that at least one rolling Snapshot copy for each virtual disk is always available on the destination volume, SnapDrive maintains a maximum of two rolling Snapshot copies on the source volume.

### 3.4.3 How SnapDrive manages rolling Snapshot copies

When an Update Mirror operation is initiated, SnapDrive checks for any existing rolling Snapshot copies of the virtual disk containing the specified virtual disk drive (Figure 3-9).



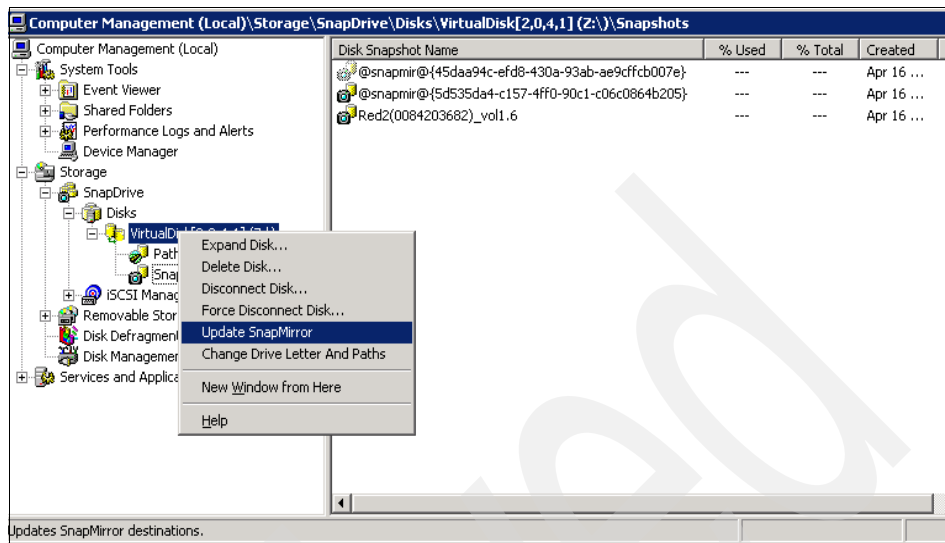


Figure 3-9 Update SnapMirror for SnapDrive virtual disk

If SnapDrive doesn't find any rolling Snapshot copies containing the virtual disk image, it creates a rolling Snapshot copy on the SnapMirror source volume. SnapDrive then initiates a SnapMirror update operation, which replicates the rolling Snapshot copy on the destination volume.

If SnapDrive finds one rolling Snapshot copy, it creates a second rolling Snapshot copy and initiates a SnapMirror update.

If SnapDrive detects two rolling Snapshot copies for the virtual disk, it deletes the older rolling Snapshot copy and creates a new one to replace it. Then SnapDrive initiates a SnapMirror update.

### 3.4.4 How rolling Snapshot copies are named using SnapDrive

The following format is used to name the rolling Snapshot copies:

@snapmir@{GUID}

GUID (Globally Unique Identifier) is a unique 128-bit number generated by SnapDrive to uniquely identify each rolling Snapshot copy (Figure 3-10 on page 84).

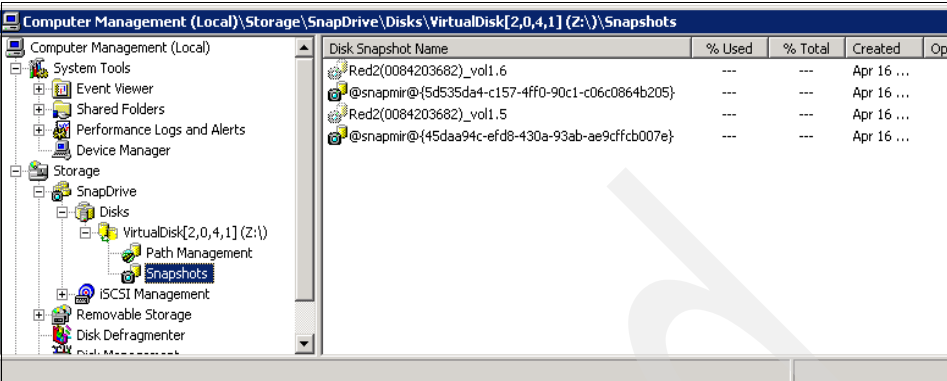


Figure 3-10 Rolling SnapShot



## **Moving SnapMirror sources for volume- and qtree-based SnapMirror**

This chapter discusses the steps necessary to move a SnapMirror relationship from one volume or N series storage system to another.

## 4.1 Overview

In a production environment, this process should be performed only in the maintenance or out-of-service window. Two scenarios are described in this section: moving the source in a VSM and moving the source in a QSM.

Whether a VSM source or QSM source is being moved to new equipment or just to newer drives, as long as there is a Snapshot copy in common on the source and destination, the transition will go smoothly. For VSM, all of the Snapshot copies are transferred as part of the SnapMirror process. For QSM, there is only one Snapshot copy in common, and different destinations have no copies in common without forcing them to be there.

## 4.2 VSM setup

The example presented here has three storage systems: the original source and destination pair and the new destination. These storage systems are called `oldsource`, `newsource`, and `destination`, and the volumes being mirrored are called `oldsourcevol`, `newsourcevol`, and `destinationvol`. If you are moving production data, you should schedule down time and make sure that new data is not added to the original source during the shuffle. For example, unexporting or unsharing the data before starting step 2 is recommended. It is assumed that SnapMirror is licensed and configured.

**Note:** For VSM, make sure that the version of Data ONTAP on the destination is greater than or equal to the version running on the new source storage system.

### 4.2.1 VSM process

1. Use SnapMirror to copy the original source to the new source, as in Example 4-1.

*Example 4-1 Copy original source to newsource system*

---

```
newsource> snapmirror initialize -S oldsource:oldsourcevol  
newsource:newsourcevol  
This process may take a while.
```

---

2. Make `oldsource` read-only.

3. To ensure that no data is lost, create a manual Snapshot copy on oldsource using the following command:

```
oldsource> snap create oldsourcevol common_snapshot
```

4. Update the destinations.

In the example, update newsource and destination based on oldsource.

Because all of the Snapshot copies are mirrored with VSM, common\_snapshot will be on all volumes.

In newsource system:

```
newsource> snapmirror update-S oldsource:oldsourcevol  
newsource:newsourcevol
```

In destination system:

```
destination> snapmirror update-S oldsource:oldsourcevol  
destination:destinationvol
```

5. Break the SnapMirror relationship.

Quiesce and break the SnapMirror relationship between oldsource and destination and oldsource and newsource.

In the destination system:

```
destination> snapmirror quiesce destinationvol  
destination> snapmirror break destinationvol
```

In the newsource system:

```
newsource> snapmirror quiesce newsourcevol  
newsource> snapmirror break newsourcevol
```

6. Update the /etc/snapmirror.conf file.

Edit the snapmirror.conf file on the destination for the new relationship. That is, replace oldsource with newsource.

Replace:

```
oldsource:oldsourcevol destination: destinationvol restart=always 0  
* * *
```

With:

```
newsource:newsourcevol destination: destinationvol restart=always 0  
* * *
```

7. Establish the new SnapMirror relationship.

Establish the relationship with **snapmirror resync**, which automatically picks the newest Snapshot copy in common to mirror. This should be common\_Snapshot (Example 4-2 on page 88).

#### Example 4-2 Establish new SnapMirror relationship

```
destination> snapmirror resync-S newsource:newsourcevol  
destination:destinationvol
```

**Note:** Any Snapshot copies older than common\_Snapshot—that is, the ones used for the last SnapMirror update—are thrown away. This is expected, and no data is lost if the original source volume was not accessible for writes following step 1.

#### 8. Verify SnapMirror.

Verify that everything worked. With **snapmirror status**, verify that the mirror is resynchronizing.

## 4.3 Moving the source for QSM

Unlike VSM, where all of the Snapshot copies from the source are mirrored to the destination, for QSM it is necessary to create a Snapshot copy on the source and force its propagation to the destination and to the new source. To accomplish this, there are two flags to the **snapmirror update** command, **-s** and **-c**. (The **-s** option tells SnapMirror to use the specified Snapshot copy on the source to synchronize with the destination, and the **-c** option instructs SnapMirror to create a Snapshot copy on a destination named after the update has completed.) You use these commands and the **snap create** command to create a Snapshot copy on the original source, and use the **SnapMirror update** command to create a Snapshot copy on the destinations (both the new source and the existing destination). Thus the Snapshot copy is in common on all volumes, allowing for the SnapMirror process to be broken from the original source and established between the new source and the existing destination.

### 4.3.1 QSM setup

The example presented here has three storage systems: the original source and destination pair and the new source. These storage systems are called **oldsource**, **newsource**, and **destination**. The **qtree** being mirrored is called **qtree**. There is nothing special to the volume names, but for clarity they are called **oldsourcevol**, **newsourcevol**, and **destinationvol**. If you are moving production data, you should schedule down time and make sure that new data is not added to the original source during the shuffle. For example, unexporting or unsharing the data before starting step 2 is recommended. It is assumed that SnapMirror is licensed and configured.

### 4.3.2 QSM process

1. Use SnapMirror to copy the original source to the newsource system (Example 4-3). This may take a while.

*Example 4-3 Copy original source to newsource system*

---

```
newsource> snapmirror initialize -S  
oldsource:/vol/oldsourcevol/qtree newsource:/vol/newsourcevol /qtree
```

---

2. Make oldsource read-only.
3. Create a manual Snapshot copy on the oldsource system by issuing the following command:

```
oldsource> snap create oldsourcevol common_snapshot
```

4. Update the mirrors.

Using the **-s** option to **snapmirror update**, synchronize newsource with oldsource and synchronize destination with oldsource based on common\_Snapshot. Using the **-c** option, have the Snapshot copy named common\_snapshot created on the destination storage systems.

In newsource system:

```
newsource> snapmirror update -c common_snapshot -s common_snapshot -S  
oldsource:/vol/oldsourcevol/qtree newsource:/vol/newsourcevol/qtree
```

In destination system:

```
destination> snapmirror update -c common_snapshot -s common_snapshot -S  
oldsource:/vol/oldsourcevol/qtree destination:/vol/destinationvol/qtree
```

5. Break the SnapMirror relationships with oldsource system.

Once you have a common Snapshot copy between newsource and destination system, you can break the relationship with oldsource before establishing the mirror between newsource and destination system.

In destination system:

```
destination> snapmirror quiesce /vol/destinationvol/qtree  
destination> snapmirror break /vol/ destinationvol /qtree
```

In newsource system:

```
newsource> snapmirror quiesce /vol/newsourcevol/qtree  
newsource> snapmirror break /vol/ newsourcevol /qtree
```

6. Update /etc/snapmirror.conf on the destination.

Edit the snapmirror.conf file on the destination for the new relationship. That is, replace oldsource with newsource.

Replace:

```
oldsource:/vol/oldsourcevol/qtree destination:/vol/  
destinationvol/qtree restart=always 0 * * *
```

With:

```
newsource:/vol/newsourcevol/qtree destination:/vol/  
destinationvol/qtree restart=always 0 * * *
```

7. Establish the new SnapMirror relationship.

Establish the relationship with **snapmirror resync** to automatically pick the newest Snapshot copy in common to both mirrors, which should be `common_snapshot`.

```
destination> snapmirror resync -S newsource:/vol/newsourcevol/ qtree  
destination:/vol/destinationvol/qtree
```

**Note:** No data is lost if the original source qtree was not accessible for writes following step 1.

8. Verify the SnapMirror relationship.

Verify that everything worked. With **snapmirror status**, verify that the mirror is resynchronizing.



## Disaster recovery failover procedures and examples

This chapter discusses the following:

- ▶ Procedure to cut over to destination SnapMirror
- ▶ Procedure to resync the source with the destination
- ▶ Procedure to rebuild the source from the destination (full disaster recovery)

## 5.1 Procedure for cutting over to destination

1. Quiesce all of the applicable SnapMirror relationships on the destination for the affected source system using the **snapmirror quiesce** command.
2. Break all of the applicable SnapMirror relationships on the destination for the affected source system using the **snapmirror break** command. At this point, the data at the destination is writable, as is the source copy.
3. Create CIFS shares or NFS exports, or both, for the data at the destination.
  - a. NFS exports

NFS exports can be populated from the mirrored `/etc/exports` file from the qtree destination of the transferred “-” qtree for the affected source system after translating them to correspond to their new location on the destination.

- b. CIFS shares

This procedure assumes the following:

- You do not have CIFS shares for the destination’s mirrored data.
- The storage systems are running in the same active directory domain and do not use local users or groups.
- The volume names on the source and destination storage system are exactly the same.
- You have a way to point all of your clients to the new source storage system for access (DFS™ or logon scripts).

For autocreated home directories, do the following:

- i. Break all of your SnapMirror relationships.
- ii. On the new source, make sure that the `cifs.home_dir` option is populated with the appropriate mirrored destination volume locations.
- iii. Run **cifs homedir load**.

For manually created directories, do the following:

- i. Make sure that all of the CIFS options on both systems are synchronized appropriately (most notably, that the Active Directory® domain is the same, DNS is properly set up for failover, and audit settings are duplicated).
- ii. Set up a Windows or UNIX batch job to copy the `/etc/cifsconfig_share.cfg` file (on the source storage system) to the SnapMirror volume or qtree with a name like `cifsconfig_share_filename.cfg`.
- iii. When the time comes to make the source live, break all of your SnapMirror relationships.

- iv. On the new source storage system, run **source /vol/<yourvolname>/cifsconfig\_share\_filename.cfg**. This shares out the data that is now readable and writable. You will get errors for duplicate share names like etc\$, c\$, HOME and others. These can be easily avoided by performing some processing in step 2 whereby these lines are removed from the cifsconfig\_share.cfg file. In addition, removing the hard-coded SID entries is a good idea.
4. Redirect all network clients from the source, which is no longer available to the destination.

## 5.2 Procedure to resync the source with the destination (minor event)

This procedure covers situations where access to the source was interrupted by some event, like a power or network outage, but the system and the data on the system were not destroyed; however, failover to the destination was performed and changes on the destination need to be restored to the source system. These procedures will not work for events where entire data volumes on the source were destroyed by some means. Such events require a limited version of the full DR procedure, but only for the affected volumes.

1. Perform a **snapmirror resync** for each qtree from the destination to the source. This will replicate any changes to the data on the destination to the source, and turn it into a read-only destination.
2. Disable NFS/CIFS access to the destination to prevent further changes to the data.
3. Perform a **snapmirror update** for all qtrees from the destination to the source to transfer any changes to the data in between the resync and disabling access.
4. Quiesce the SnapMirror relationships on the source.
5. Break the SnapMirror relationships on the source. This will turn the source back into a read-write source with all of the changes made.
6. Redirect all network clients back to the source.
7. Resync all qtrees from the source to the destination to resume normal SnapMirror operation.

## 5.3 Procedure to rebuild the source from the destination (full disaster recovery)

This procedure covers events where the source system was completely destroyed by some natural or human-caused disaster, and failover to the destination was performed. These events should be very rare.

1. Perform enough setup of the new source system to give it appropriate IP addressing and licensing (if correct licenses are not already installed from manufacturing). It is not necessary to complete NFS/CIFS/HTTP configuration; it will be restored in a later step.
2. Create new volumes on the new source to hold data to be recovered.

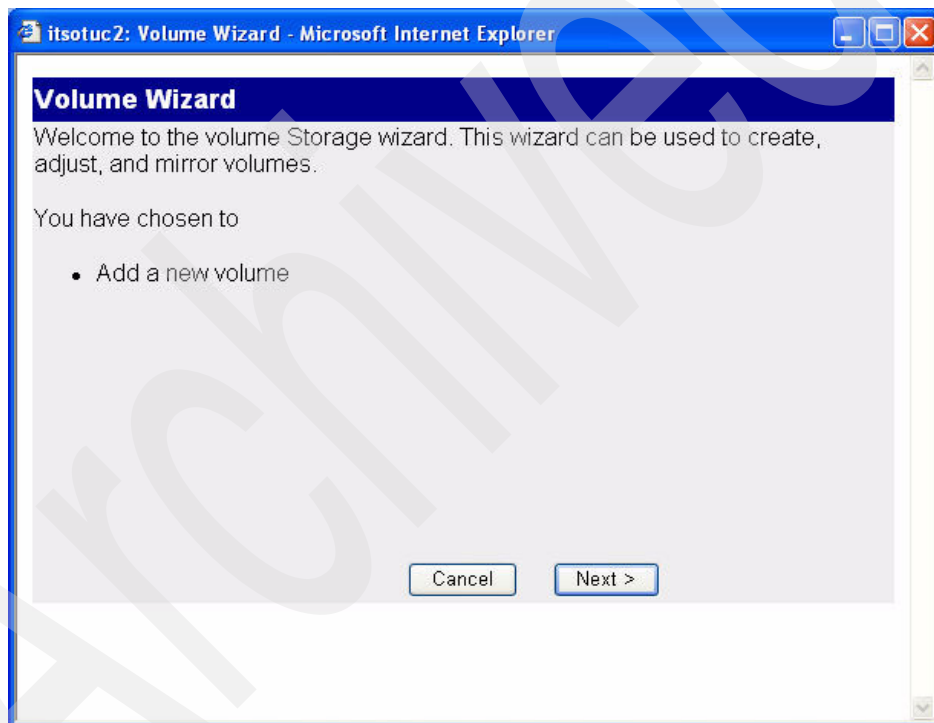


Figure 5-1 Volume creation

3. Configure SnapMirror access on the destination to allow the source to retrieve its data.

The screenshot shows the 'FilerView®' application window. The title bar is 'FilerView®'. The main window has a header 'Add SnapMirror Remote Access Entry For itsotuc1' with a help icon. Below the header is a breadcrumb 'SnapMirror → Remote Access → Add'. The form contains the following fields and options:

- Destination Host:** A text input field with the placeholder text 'Enter the name of filer allowed to access this host (itsotuc1)'.
- Access All Volumes:** A radio button that is selected. Below it is the text 'Choose this to allow access to all volumes.'
- Single Volume Access:** An unselected radio button. Below it is the text 'Choose this to specify access to a specific volume.'
- Source Volume:** A dropdown menu showing 'vol0' with a help icon. Below it is the text 'Choose the volume on host itsotuc1 to allow access to.'
- Add:** A button at the bottom right of the form.

Figure 5-2 Remote access

4. Enable SnapMirror on the new source.
5. Perform a baseline transfer for all of the qtrees in the data volumes and any qtrees on the root volume other than the “-” qtree to the new source.
6. If data existed outside of qtrees in the data volumes and was replicated via the “-” source to a qtree on the destination, perform a baseline of this qtree to a temporary qtree on the new source. Once complete, use the NDMP copy process to copy this data from the temporary qtree to its original location on the new source.
7. Repeat step 5 for the “-” qtree from the root volume.
8. Use the **config restore** command to restore the registry on the new source.
9. Reboot the new source to ensure that the registry settings are correct.
10. Disable NFS/CIFS access to the destination to prevent further changes to the data (Example 5-1 on page 96).

#### *Example 5-1 Turn off NFS*

---

```
itsotuc2*> nfs off  
NFS server is NOT running.  
itsotuc2*>
```

---

11. Perform a **snapmirror update** for all qtrees from the destination to the source to transfer any changes to the data in between the resync and disabling access.
12. If any data changed from any “-” qtrees, use NDMP copy to copy that data back to the original location from the temporary qtree, as in step 5.
13. Quiesce the SnapMirror relationships on the source.
14. Break the SnapMirror relationships on the source. This turns the source back into a read-write source with all of the changes made.
15. Redirect all network clients back to the source.
16. To correct the relationships of any “-” qtrees from the temporary recovery qtree back to “-”, break the new temporary relationship used to restore data back to the source. Then release the relationship on the source to destroy the temporary relationship. Finally, use **snapmirror initialize** to re-establish the relationships.
17. Resync all qtrees from the source to the destination to resume normal SnapMirror operation.

# LREP and SnapMirror

This chapter discusses the use of LREP in SnapMirror.

## 6.1 How LREP works

LREP is a logical replication tool that is useful for SnapMirror or SnapVault initial transfers (also commonly referred to as seeding baselines).

Offices and sites that are small enough to have name, print, and file service requirements met by a single server or a small number of servers typically have limited WAN bandwidth. Many field offices, sales offices, and regional offices also have limited bandwidth. SnapMirror is a strong solution for replicating data for DR backup, but the initialization transfer (baseline) can be crippling in a WAN environment.

Use the LREP utility to write the initial transfer (the baseline) to portable media. The portable hardware could also be an N3700 shared between offices. Ship the portable media to the data center and locally write to the destination system. No network bandwidth is used, only a manual process of moving the media from remote site to data center. Once the data is on the destination system, modify the SnapMirror relationship to reflect the actual source to destination relationship.

There are two utilities required for the entire LREP process, the `lrep_reader` and `lrep_writer`. The `lrep_writer` is used at the data center or location of the destination system, `lrep_reader` at the remote office.

### Example

This example sets up a QSM baseline transfer from filerA (source) to filerB (destination), where the source path is `/vol/srcvol/srcqtree` on filerA and the destination path is `/vol/dstvol/dstqtree` on filerB. Ensure that both filerA and filerB have SnapMirror licensed. Ensure proper access permission on filerA for the clients.

#### ***Remote office/site***

1. License and install SnapMirror on the remote office storage system (filerA) and install `lrep_reader` on the remote server, client1.
2. Arrange for local/NFS/CIFS storage for client1.

This should be 5% more than the actual source data set size. In this example, choose vol1 on filerA.

3. Mount vol1 on client1 (use CIFS map if client1 is a Windows machine) using the commands in Example 6-1.

#### *Example 6-1 Mounting LREP volume*

---

```
client1% mkdir -p /t/filerA/vol1
```



```
client1% mount -o rw filerA:/vol/vol1 /t/filerA/vol1
```

---

4. Create a directory on client1 for lrep\_reader storage using the commands in Example 6-2 (use corresponding Windows commands if client1 is a Windows system).

*Example 6-2 Create directory for lrep\_reader storage*

---

```
client1% cd /t/filerA/vol1
client1% mkdir lrep_dump
```

---

5. Verify that the lrep\_dump directory is empty or does not contain files that might be overwritten (Example 6-3).

*Example 6-3 lrep\_dump directory*

---

```
client1% ls /t/filerA/vol1/lrep_dump
```

---

6. Use the lrep\_reader utility to read the logical replication stream for a snapmirror initialize operation. The path on filerA that you are reading from is /vol/srcvol/srcqtree.
7. Navigate to the directory that contains the lrep executable and enter the command in Example 6-4.

*Example 6-4 Initialize snapmirror through lrep\_reader command*

---

```
client1% lrep_reader -p snapmirror_init -f filerB -q
/vol/dstvol/dstqtree -o /t/filerA/vol1/lrep_dump/lrep_srcqtree@0
Transfer started.
Use 'snapmirror status' on filer to monitor progress.
Transfer done.
Verify by using 'snapmirror status' on filer.
client1%
```

---

Examine one argument at a time as in Example 6-5.

*Example 6-5 lrep\_reader arguments*

---

```
-p snapmirror_init= use Qtree SnapMirror protocol
-f filerB = filerB is the final destination
-q /vol/dstvol/dstqtree = the full path on the FINAL destination
-o /t/filerA/vol1/lrep_dump/lrep_srcqtree@0 = file extension and
location, a name for the file that will be created, @number of 2GB
files (0=infinite). This feature can allow you to span multiple file
systems.
```

---

If your portable media are small, for instance 8 GB, and your data is 12 GB, you have the option of spanning multiple file systems or mountpoints using the following command:

```
-o /t/filerA/vol1/lrep_dump/lrep_srcqtree@0 -o  
/n/filerA/vol1/lrep_dump/lrep_srcqtree@0
```

**Note:** `/t/filerA/vol1/lrep_dump/lrep_srcqtree@4` would mean creating a maximum of four files, each of 2 GB, so it directs `lrep_reader` to store the first 8 GB in `/t/filerA/vol1/lrep_dump/lrep_srcqtree`.

**Note:** `<path>@0` (0 means *unlimited* here) means creating all files till the end of stream in `<path>`.

The command `lrep_reader` has started a logical replication transfer from `filerA` using source path `/vol/srcvol/srcqtree`, and it stored the logical stream in files with `/t/filerA/vol1/lrep_dump/lrep_srcqtree` as their prefix.

You also specified the destination storage system name and destination volume to track the transfer using the `snapmirror status` command.

The command `lrep_reader` created files containing logical replication stream data as in Example 6-6.

*Example 6-6 Logical replication stream files*

---

```
client1% ls /t/filerA/vol1/lrep_dump/  
-rwxr-xr-x 1 root 2147475456 Feb 3 01:32 lrep_srcqtree.0  
-rwxr-xr-x 1 root 2147475456 Feb 3 01:36 lrep_srcqtree.1  
-rwxr-xr-x 1 root 2147475456 Feb 3 01:40 lrep_srcqtree.2  
-rwxr-xr-x 1 root 2147475456 Feb 3 01:44 lrep_srcqtree.3  
-rwxr-xr-x 1 root 2147475456 Feb 3 01:48 lrep_srcqtree.4  
-rwxr-xr-x 1 root 2147475456 Feb 3 01:52 lrep_srcqtree.5  
-rwxr-xr-x 1 root 2147475456 Feb 3 01:56 lrep_srcqtree.6  
-rwxr-xr-x 1 root 2147475456 Feb 3 02:00 lrep_srcqtree.7
```

---

In this example you dump the directory containing these files to tape (your portable media). You have stored these files on system storage, so you use the storage system's native **dump** command to dump this directory to destination storage (NDMP dump can also be used), as in Example 6-7.

*Example 6-7 dump command*

---

```
filerA> dump 0f rst0a /vol/vol1/lrep_dump/
```

---

## 6.2 Data center

All steps in this section are issued at the data center unless noted otherwise.

Once the `lrep_reader` files have been successfully dumped to tape and those tapes have been successfully transferred to the data center, it is time to restore them to `filerB` (which is also a storage system in the example).

Make the files created by `lrep_reader` available on `client2` (which should have `lrep_writer` installed). In this example you are accessing the files via NFS. Read permission is necessary only for these files on `client2`. You assume that these files have been restored to `vol2` on `filerB`.

Mount `vol2` from `filerB` on `client2`, as in Example 6-8. You will delete temporary files, so mount it `r/w`.

*Example 6-8 Mounting vol2 from client2*

---

```
client2% mkdir -p /t/filerB/vol2
client2% mount -o rw filerB:/vol/vol2 /t/filerB/vol2
```

---

Run `lrep_writer` by specifying protocol (`SnapMirror`) and the logical replication file name prefix. In this example, use `/t/filerB/vol2/lrep_dump/lrep_srcqtree` as the file name's prefix. `lrep_writer` then reads the logical replication stream from `lrep_srcqtree.0`, `lrep_srcqtree.1`, and so on until the last file in this continuous sequence (Example 6-9).

*Example 6-9 lrep\_writer command*

---

```
client2% lrep_writer -p snapmirror_init
/t/filerB/vol2/lrep_dump/lrep_srcqtree
Waiting for connection.
```

---

If `file_name` is specified as source of the logical replication stream, `lrep_writer` looks for `file_name.0`, `file_name.1`, `file_name.2`, and so on, until it fails to find a file in this sequence. It assumes the end of stream at that point.

If there are files containing logical replication streams spread across multiple directories, give multiple paths to `lrep_writer`.

Now start the transfer from `filerB` to read the logical replication stream data from `client2` (Example 6-10).

*Example 6-10 Initialize snapmirror*

---

```
filerB> snapmirror initialize -S client2:/vol/srcvol/srcqtree
/vol/dstvol/dstqtree
```

---

Transfer started. Monitor progress with 'snapmirror status' or the SnapMirror log.

---

On client2, the output in Example 6-11 is present.

*Example 6-11 Output in client2*

---

```
client2%  
Waiting for connection.  
Connection accepted from filerB.  
Transfer started.  
Use 'snapmirror status' on filer to monitor progress.  
Transfer done.  
Verify by using 'snapmirror status' on filer.
```

---

The SnapMirror baseline transfer is now done (seeded). It is time to replicate data directly from filerA to filerB. From this point on use the **snapmirror update -S filerA:/vol/dstvol1/dstqtree** command on filerB to transfer incremental changes from filerA to filerB. Or, add an **/etc/snapmirror.conf** entry with a schedule for the new SnapMirror relationship to make it a permanent, scheduled replication.

# Index

## A

- ACK 59
- adding disks 50
- aggregate volume 41
- application performance 26
- application servers 2
- application testing 3
- architecture 37
- archive 2
- asynchronous 20
- asynchronous replication 19
- asynchronous update 31
- authenticate filers 65
- autosupport 52

## B

- backup 2
- backup consolidation 3
- backup windows 3
- baseline copy 19
- bidirectional pair 42
- bidirectional synchronous 41
- block 19
- block map 19
- breaking SnapMirror relationship 10
- busy locks 17

## C

- cache 27
- cascading mirrors 23
- cascading SnapMirror 51
- centralized backup operations 3
- change of state 30
- changed blocks 19, 36
- chip simulations 38
- CIFS 8
- CIFS password 8
- client system 28
- cloning 38
- cluster 41
- cluster failover 42
- config restore 95

- consistency point forwarding 25, 31
- contention 3
- convert\_unicode 6
- CP 26
- CP synchronization 26, 29
- CPU contention 3
- CPU usage 26
- create DR for SnapVault 74
- cron 5
- customer scenario 47

## D

- data block 35
- data distribution 22
- data loss 68
- Data ONTAP 17–19
- data protection 2
- data replication 2
- data sets 22
- data size 43
- database environments 3
- database server 3
- DataFabric™ Manager 61
- deadlock 42
- dedicated network 47
- delete copy 17
- dest\_filer 18
- destination file system 22
- destination storage 10
- destination system 4
- destination volume 19
- df 50
- dfdrm 69
- DFM 62
- DFM Disaster Recovery Manager 63
- disaster recovery 2
- disk bandwidth 44
- disk size 21
- disk technologies 21
- DR site 2
- dump 22
- dumpfile 72

## F

Failover 67  
failover mode 31  
Failover Policies 68  
fail-safe synchronous 30  
FC network 4  
feasible 43  
file system 18  
file-oriented 27  
Firewall Configuration 47  
flexible volumes 19  
FlexVol 19  
FlexVol deployment 21  
free space 46  
ftp 72  
fully synchronous 32

## G

giveback operation 50  
guarantee mode 41  
guidelines 46

## H

host system 5  
hostname 58

## I

image backups 35  
incremental changes 5  
incremental Snapshot copy 43  
incremental transfers 35  
independent entity 39  
initial base Snapshot copy image 43  
initialized 22  
inode 20  
insufficient memory 58  
in-sync 31  
integration 61  
iSCSI 27

## K

key differences between VSM and QSM 21

## L

latency 47  
level-0 43

license 5, 74  
license fees 25  
limited bandwidth 98  
load sharing 3  
local tape drive 43  
locks 17  
logical replication 20–21, 42  
logical replication tool 98  
long-term retention 36  
low bandwidth 43  
low throughput 46  
LREP 97  
lrep\_reader 98  
lrep\_writer 101  
LUNs 5

## M

many-to-many replication 4  
many-to-one configuration 48  
memory configuration 47  
metadata 19  
migrating 20, 43  
mirroring 2  
multihop 22  
multipath support 31  
multiple relationships 22  
multiplexing mode 31, 67

## N

NDMP 22  
network 2  
network bandwidth 52  
network configuration 47  
network utilization 3  
newsource 87  
NFS 28  
nitial baseline transfers 76  
NVLOG 25  
NVLOG forwarding 26, 31  
NVRAM journal 27

## O

objects 14  
off-site storage 3  
oldsourcevol 86  
open files 50  
optimized 22

options snapmirror.access 33  
OSSV 75  
Outstanding Interval 48  
overcommit an aggregate 41

## P

pairing 67  
parent 37  
parity 28  
partitioned 3  
performance 18, 46  
performance testing 3  
physical block 19  
physical layout 20  
physical spindles 35  
point in time 15, 37  
Policies 67  
port 10566 47  
production storage systems 2  
production systems 3

## Q

QSM 18  
qtree 4  
qtree replication 17, 20–21  
qtree SnapMirror 18, 20

## R

RAID 36  
RAID groups 46  
raw volume 35  
raw write throughput 47  
read performance 35  
Read permission 101  
Read socket timeout 59  
read-ahead 35  
read-only 3  
read-only access 2  
read-only volume 4  
read-write 10  
recovery 2  
Redbooks Web site xi  
reestablished 30  
remote data access 2  
remote destination system 4  
remote locations 3  
remote tape archive 3

replicated qtrees 22  
replicating 2–3  
replication methods 26  
replication policy 68  
restoration 36  
resynchronize 10, 14  
Resyncing the Mirror 10  
reverse 14  
root volume 48

## S

SAN 5  
schedules 4, 40  
scheduling Snapshot copy 49  
SEC regulations 71  
security of data 70  
simultaneous replication 44  
simultaneous replications 34  
snap list 15–16  
SnapLock Compliance QSM relationships 73  
SnapLock Compliance volumes 71  
SnapLock Enterprise destination volumes 72  
SnapLock Enterprise volumes 72  
SnapMirror 1  
snapmirror break 14, 92  
SnapMirror cascade 39  
SnapMirror configuration 5  
snapmirror destination volume 9  
snapmirror destinations 16  
snapmirror destinations -s 16  
snapmirror initialize 8, 40, 73  
SnapMirror license 25  
SnapMirror Monitoring Interval 63  
SnapMirror qtree replication 20  
snapmirror quiesce 92  
snapmirror release 17  
snapmirror resync 12, 14, 40  
SnapMirror storage 10  
SnapMirror technology 22  
snapmirror update 12, 88, 93  
snapmirror.access 33, 58  
snapmirror.allow 33, 57  
snapmirror.conf 5, 33  
snaprestore 40  
Snapshot 4  
Snapshot copies 48–49  
Snapshot copy 15  
Snapshot copy behavior 15

- Snapshot schedules 21
- SnapVault 22
- SnapVault secondary system 76
- snapvault start 78
- snapvault status 78
- SNMP 33
- source volumes 17
- space guarantees 41
- split 38
- srcldst 18
- srcvol 8, 13
- statit 53
- storage system 47
- Storage System Performance 48
- substantial 38
- supported volume type 41
- surviving storage system 44
- Sync timeouts 59
- synchronicity level control 32
- synchronization 3
- synchronous replication 19
- sysid 17
- sysstat 55
- System Features 61
- system fragmentation 36
- systems administrators 3

## T

- tape 35
- tape backup overhead 3
- tape device 22, 35
- TCP socket 47
- TCP window size 46
- TCP/IP 81
- theoretically 35
- throttle option 47
- tolerance 23
- traditional deployments 19
- traditional volumes 19, 35
- transfer 16–17
- traversing the network. 42

## U

- unequal disk sizes 48
- Unicode 49
- UNIX 72

## V

- visibility interval 32
- vol autosize maxgrowth 41
- vol status 12
- vol status -r 58
- Volume SnapMirror 18
- volume-based replication 21
- volume-based SnapMirror 18

## W

- WAFL 28
- waf\_l\_map\_nt\_admin\_to\_root 81
- WAN 35
- WANs 46
- weather simulations 38
- whole volumes 18
- WORM optical platters 70
- writable attributes 40
- writable LUNs 40
- Write Anywhere File Layout 19
- Write Once, Read Many (WORM) 70
- write socket timeout 59
- write throughput 47



## IBM System Storage N Series SnapMirror

(0.2" spine)  
0.17" <-> 0.473"  
90 <-> 249 pages







# IBM System Storage N Series SnapMirror



**Redbooks**

**Integrating  
SnapMirror with  
other N series  
features**

**Using SnapMirror to  
protect information  
across the enterprise**

**Understanding  
SnapMirror  
behaviors and best  
practices**

This IBM Redbook introduces the features and capabilities of SnapMirror, and describes how SnapMirror technology can be used in an IBM System Storage N Series environment to achieve enterprise data protection.

SnapMirror provides a fast and flexible enterprise solution for mirroring or replicating data over local area, wide area, and Fibre Channel networks. By providing a simple solution for replicating data, SnapMirror addresses several critical application areas, including disaster recovery, remote data access, simulating production environments for testing purposes, creating remote archives, and load sharing.

This book is intended for IT professionals who are evaluating SnapMirror technology as well as those who are deploying and architecting a SnapMirror solution.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)