EMC® Solutions Enabler Symmetrix SRM CLI

Version 7.6

Product Guide

REV 01

EMC²

Copyright © 2002- 2013 EMC Corporation. All rights reserved. Published in the USA.

Published November 2012

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided as is. EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose. Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

EMC^2, EMC, and the EMC logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. All other trademarks used herein are the property of their respective owners.

For the most up-to-date regulatory document for your product line, go to the technical documentation and advisories section on the EMC online support website.

CONTENTS

Preface

Part 1 **Concepts and Procedures**

Chapter 1 **SRM Overview**

Storage Resource Management overview 18
Data object mapping commands 18
Database object commands 19
Host file system command 20
Logical volume commands 20
Performance statistics commands 21
Daemon service 22
Daemon authorization 22
Setting operating system level permissions 24

Chapter 2 **Data Object SRM**

Overview 26
Data object terminology and architecture 26
Examining device partitions 27
Examining device data objects 29
Examining logical volume data objects 31
Examining file system data objects 33

Chapter 3 **Database SRM**

Overview 36
Database architecture and terminology 36
Establishing the database connection 48
Setting UNIX environment variables 48
Connecting to the database 48
Database startup options 49
Database shutdown options 51
Using database daemons 53
What are daemons? 53
Why use daemons? 53
Running database daemons 54
Listing database instances 56
Examining database files 57
Examining database file attributes 57
Listing tablespace files 58
Translating database devices to Symmetrix groups 59
Examining tablespaces 60
Examining tablespace attributes 60
Listing tablespace files 61
Listing tablespace tables 62
Listing segments in a tablespace 62
Translating tablespace devices to Symmetrix groups 63
Examining schemas 64

Listing schema files 64
Listing schema tables 64
Examining schema attributes 65
Listing segments in a schema 66
Examining tables 66
Examining table attributes 67
Examining segments 67
Examining segment attributes 68
Invoking database I/O control 69
Freezing the database 69
Thawing the database 69
Hot backup control 69
Checkpoint 70
Archive log 70
SQL Server snapshot 70
Using the EMC Oracle ASM library 71
Usage of CLI asmdscvr 72

Chapter 4 File System SRM

Overview 74
Terminology 74
Finding file systems 74
Examining a file system 75
Finding directories and files 76
Examining directories 77
Listing files 77
Examining files 78

Chapter 5 Logical Volume SRM

Overview 82
LVM terminology 82
Mirror configurations 83
Logical volume configurations 87
Mirror conditions 88
Viewing volume groups 88
Listing the volume groups 88
Volume group details 89
Volume group control operations 92
Translating volume groups to Symmetrix groups 94
Viewing logical volumes 96
Listing logical volumes 96
Logical volume details 96
Logical volume conditions 98
Logical volume states 98
Logical volume attributes 98
Logical volume control operations 99
Creating a logical volume 99
Viewing extents 101
Extent conditions 102
Expanded list 102
No extents in list 102

Chapter 6 Statistics SRM

Retrieving statistics 106
Supported metrics 106

Part 2 Operational Examples

Chapter 7 SRM Examples

Example 1: Displaying relational database objects 118
Examining Oracle database objects 118
Examining IBM DB2/UDB database objects 120
Examining Informix database objects 121
Examining Microsoft Exchange database objects 123
Example 2: Mapping files and other disk storage objects 125
Example 3: Displaying volume groups and logical volumes 137
Example 4: Deporting and importing a volume group 142
Example 5: Mapping files 144
Example 6: Displaying a logical volume 147

FIGURES

	Title	Page
1	Objects, extents, and data blocks	26
2	Oracle database architecture	37
3	Segments, extents, and data blocks	38
4	SQL database architecture	40
5	Sybase database architecture	41
6	Informix database architecture	42
7	DB2/UDB database architecture	44
8	Exchange database architecture	45
9	System view of the daemon process	53
10	Database daemon libraries for communication	55
11	Volume group	83
12	Simple mirror configuration	84
13	Concatenated mirror configuration	85
14	Striped mirror configuration	85
15	RAID 5 Veritas and Windows 2000 LDM configuration	86

TABLES

	Title	Page
1	Data object mapping SRM commands	18
2	Database object commands	19
3	Host tile system command	20
4	Logical volume commands	20
5	Performance statistics commands	21
6	SRM control operations	23
7	UNIX database environment variables	48
8	Database daemon names	55
9	Logical volume configuration types	87
10	Logical volume mirror configurations	87
11	Mirror condition descriptions	88
12	Volume group types	90
13	Volume group control options	93
14	Logical volume condition descriptions	98
15	Options for logical volume control operations	99
16	Logical volume extent conditions	102
17	SRM statistics commands	106
18	Metric options for Oracle database types	110
19	Metric options for SQL Server databases	111
20	Metric options for Sybase databases	112
21	Metric options for IBMUDB databases	113
22	Extent condition descriptions	139

PREFACE

As part of an effort to improve its product lines, EMC periodically releases revisions of its software and hardware. Therefore, some functions described in this document might not be supported by all versions of the software or hardware currently in use. The product release notes provide the most up-to-date information on product features.

Contact your EMC representative if a product does not function properly or does not function as described in this document.

Note: This document was accurate at publication time. New versions of this document might be released on the EMC online support website. Check the EMC online support website to ensure that you are using the latest version of this document.

Purpose

This document describes how to configure and use EMC Solutions Enabler for Storage Resource Management (SRM).

Audience

This guide provides both conceptual and reference information for command-line users and script programmers that focus on examining SRM information related to various data objects and data handling facilities within a host system.

Related documentation

The following EMC publications provide additional information:

- *EMC Solutions Enabler Release Notes*
- *EMC Solutions Enabler Installation Guide*
- *EMC Symmetrix Security Configuration Guide*
- *EMC Solutions Enabler SYMCLI Command Reference Guide*
- *EMC Solutions Enabler Symmetrix Array Controls CLI Product Guide*
- *EMC Solutions Enabler Symmetrix Array Management CLI Product Guide*
- *EMC Solutions Enabler Symmetrix SRDF Family CLI Product Guide*
- *EMC Solutions Enabler Symmetrix SRDF/Star CLI Product Guide*
- *EMC Solutions Enabler Symmetrix TimeFinder Family CLI Product Guide*
- *EMC Solutions Enabler Symmetrix Migration CLI Product Guide*
- *EMC Symmetrix Remote Data Facility Product Guide*
- *EMC Symmetrix Remote Data Facility for VMAX 10K Series Product Guide*
- *EMC Symmetrix TimeFinder Product Guide*
- *EMC Symmetrix TimeFinder for VMAX 10K Series Product Guide*
- *EMC host connectivity guides for [your operating system]*

Conventions used in this document

EMC uses the following conventions for special notices:

⚠CAUTION

CAUTION, used with the safety alert symbol, indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

Note: A note presents information that is important, but not hazard-related.

IMPORTANT

An important notice contains information essential to software or hardware operation.

Typographical conventions

EMC uses the following type style conventions in this document:

Normal	Used in running (nonprocedural) text for: • Names of interface elements, such as names of windows, dialog boxes, buttons, fields, and menus • Names of resources, attributes, pools, Boolean expressions, buttons, DQL statements, keywords, clauses, environment variables, functions, and utilities • URLs, pathnames, filenames, directory names, computer names, links, groups, service keys, file systems, and notifications
Bold	Used in running (nonprocedural) text for names of commands, daemons, options, programs, processes, services, applications, utilities, kernels, notifications, system calls, and man pages Used in procedures for: • Names of interface elements, such as names of windows, dialog boxes, buttons, fields, and menus • What the user specifically selects, clicks, presses, or types
Italic	Used in all text (including procedures) for: • Full titles of publications referenced in text • Emphasis, for example, a new term • Variables
`Courier`	Used for: • System output, such as an error message or script • URLs, complete paths, filenames, prompts, and syntax when shown outside of running text
`Courier bold`	Used for specific user input, such as commands
`Courier italic`	Used in procedures for: • Variables on the command line • User input variables
< >	Angle brackets enclose parameter or variable values supplied by the user
[ ]	Square brackets enclose optional values
|	Vertical bar indicates alternate selections — the bar means “or”
{ }	Braces enclose content that the user must specify, such as x or y or z
...	Ellipses indicate nonessential information omitted from the example

Where to get help

EMC support, product, and licensing information can be obtained on EMC Online Support, as described next.

Note: To open a service request through EMC Online Support, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or to answer any questions about your account.

Product information

For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to EMC Online Support (registration required) at:

`https://support.EMC.com`

Technical support

EMC offers a variety of support options.

Support by Product — EMC offers consolidated, product-specific information on the Web at:

`https://support.EMC.com/products`

The Support by Product web pages offer quick links to Documentation, White Papers, Advisories (such as frequently used Knowledgebase articles), and Downloads, as well as more dynamic content, such as presentations, discussion, relevant Customer Support Forum entries, and a link to EMC Live Chat.

EMC Live Chat — Open a Chat or instant message session with an EMC Support Engineer.

eLicensing support

To activate your entitlements and obtain your Symmetrix license files, visit the Service Center on `https://support.EMC.com`, as directed on your License Authorization Code (LAC) letter emailed to you.

For help with missing or incorrect entitlements after activation (that is, expected functionality remains unavailable because it is not licensed), contact your EMC Account Representative or Authorized Reseller.

For help with any errors applying license files through Solutions Enabler, contact the EMC Customer Support Center.

If you are missing a LAC letter, or require further instructions on activating your licenses through the Online Support site, contact EMC's worldwide Licensing team at licensing@emc.com or call:

- North America, Latin America, APJK, Australia, New Zealand: SVC4EMC (800-782-4362) and follow the voice prompts.
- EMEA: +353 (0) 21 4879862 and follow the voice prompts.

PART 1

Concepts and Procedures

This guide is divided into two parts: Part 1 provides concepts and procedures of the Storage Resource Management (SRM) and Part 2 contains operational examples.

Part 1 contains the following chapters:

Chapter 1, “SRM Overview”

Provides an overview of the SYMCLI SRM component that focuses on the open system tasks of mapping and examining various objects on your host system, their relational databases, and the attached storage arrays.

Chapter 2, “Data Object SRM”

Describes the host system SRM. It explains how the extents of datafiles and objects can be examined using SYMCLI commands.

Chapter 3, “Database SRM”

Describes the host system database SRM for various installed relational databases, and how the databases and their structure can be examined using SYMCLI commands.

Chapter 4, “File System SRM”

Describes the host file system SRM and how the file systems and their structure can be examined using SYMCLI commands.

Chapter 5, “Logical Volume SRM”

Describes the host system logical volume SRM to Symmetrix devices and how logical volumes and their groups can be examined using SYMCLI commands.

Chapter 6, “Statistics SRM”

Describes the SRM statistics and explains how to generate them using SYMCLI commands.

CHAPTER 1
SRM Overview

This chapter provides an overview of the Solutions Enabler Storage Resource Management component that focuses on the open system tasks of mapping and examining various storage objects on the host system.

- Storage Resource Management overview 18
- Daemon service 22

Storage Resource Management overview

The Storage Resource Management (SRM) component extends the basic SYMCLI command set to include SRM commands. These commands display the attributes of various objects in a host system, including relational databases and EMC® storage arrays.

The SYMCLI commands support SRM in the following areas:

- Data objects
- Relational databases
- File systems
- Logical volumes and volume groups
- Performance statistics

Note: For a list of all supported platforms and versions for Solutions Enabler, refer to the E-Lab™ Interoperability Navigator, which can be found at `http://elabnavigator.EMC.com`.

Data object mapping commands

The data object mapping commands display the mapping of storage devices and the characteristics of datafiles and objects, as described in Table 1.

Table 1 Data object mapping SRM commands

Command	Argument	Displays
`symrslv`	`pd`	Logical-to-physical mapping information about any physical device.
	`lv`	Logical-to-physical mapping information about a logical volume.
	`file`	Logical-to-physical mapping information about a file.
	`dir`	Logical-to-physical mapping information about a directory.
	`fs`	Logical-to-physical mapping information about a file system.
`sympart`	`show`	Shows a detailed partition for the specified host device.
	`list`	Lists partition information for all the devices connected to a host.

For more information about data object mapping, refer to Chapter 2 "SRM Overview." For more information about the `symrslv` command, refer to the *EMC Solutions Enabler SYMCLI Command Reference Guide*.

Database object commands

The database object commands list the attributes that describe database structures, files, tablespaces, and user schemas, as described in Table 2.

Table 2 Database object commands

Command	Argument	Actions
symrdb	list	Lists various physical and logical database objects: • Current relational database instances available • Tablespaces, tables, files, or schemas of a database • Files, segments, or tables of a database tablespace or schema
	show	Shows information about a database object: • Tablespace, tables, file, or schema of a database • File, segment, or a table of a specified tablespace or schema
	stats	Displays performance statistics for databases.
	rdb2dg rdb2cg	Translates the devices of a specified database into a device group or a composite group.
	tbs2dg tbs2cg	Translates the devices of a specified tablespace into a device group or a composite group.
	startup	Provides startup options for a database manager instance.
	shutdown	Provides shutdown options for a database manager instance.
symioctl	freeze	Freezes I/O to a specified database application.
	thaw	Thaws I/O to a specified database application.
	begin backup	Places objects into hot backup mode.
	end backup	Takes objects out of hot backup mode.
	checkpoint	Issues a checkpoint to the RDBMS.
	archive log	Archives the current log.
	begin snapshot	Begins a snapshot backup on SQLServer 2000 and higher.
	end snapshot	Saves snapshot metadata and resumes writes on SQLServer 2000 and higher.
	restore snapshot	Restores previously saved snapshot metadata on SQLServer 2000 and higher.
	abort snapshot	Terminates the snapshot without saving metadata and resumes writes on SQLServer 2000 and higher.

For information about database mapping, refer to Chapter 3 “Database SRM.” For more information about the `symrdb` and `symioctl` commands, refer to the *EMC Solutions Enabler SYMCLI Command Reference Guide*.

Host file system command

The host file system command lists the file systems that are in use on a host operating system. The command provides listings and attributes that describe file systems, directories and files, and their mappings to physical devices and extents, as described in Table 3.

Table 3 Host tile system command

Command	Argument	Displays
`symhostfs`	`list`	A list of file systems, files, or directories.
	`show`	Detailed information about a file system or file system object.

For more information about file system mapping, refer to Chapter 4 "File System SRM." For more information about the `symhostfs` command, refer to the *EMC Solutions Enabler SYMCLI Command Reference Guide*.

Logical volume commands

The logical volume commands map logical volumes and display the underlying storage devices. Logical volume architecture defined by a Logical Volume Manager (LVM) is a means for advanced applications to improve performance through the strategic placement of data, as described in Table 4.

Table 4 Logical volume commands (page 1 of 2)

Command	Option	Description
`symlv`	`list`	Displays a list of logical volumes in a specified volume group.
	`show`	Displays detail information (including extent data) about a logical volume.
	`add`	Adds mirror images to a logical volume of the specified type.
	`create`	Creates a logical volume of the specified type.
	`delete`	Deletes a logical volume of the specified type.
	`extend`	Extends (grows) a logical volume of the specified type.
	`reduce`	Reduces (shrinks) a logical volume of the specified type.
	`remove`	Removes mirrors of a logical volume of the specified type.
	`stats`	Shows performance statistics about logical volumes.

Table 4 Logical volume commands (page 2 of 2)

Command	Option	Description
symvg	adddev	Extends a volume group by adding the specified devices to the volume group.
	create	Creates a volume group using the specified devices.
	deport	Deports a specified volume group so it can be imported later.
	destroy	Destroys a volume group.
	import	Imports a specified volume group.
	list	Displays a list of volume groups defined on your host system by the logical volume manager.
	recover	Recovers a failed volume group.
	rescan	Rescans all the volume groups.
	rmdev	Reduces a volume group by removing the specified devices from the volume group.
	show	Displays more detail information about a volume group.
	vg2cg	Translates volume groups to composite groups.
	vg2dg	Translates volume groups to device groups.

For more information about logical volume mapping, refer to Chapter 5 "Logical Volume SRM." For more information about the symlv and symvg commands, refer to the *EMC Solutions Enabler SYMCLI Command Reference Guide*.

Performance statistics commands

The performance statistics commands retrieve statistics about a host's CPU, disk, memory, logical volumes, and databases, as described in Table 5.

Table 5 Performance statistics commands

Command	Argument	Displays
symhost	stats	Performance statistics for the host CPU, disk, and memory.
	show	Host configuration information.
symlv	stats	Performance statistics about logical volumes.
symrdb	stats	Performance statistics for databases.

For more information about statistics, refer to Chapter 6 "Statistics SRM." For more information about the symhost, symlv, and symrdb commands, refer to the *EMC Solutions Enabler SYMCLI Command Reference Guide*.

Daemon service

The SRM daemon (storsrmd and storsrmd64) is a process or a service that allows certain non-root users and non-administrators to perform some SRM operations. For API calls on UNIX systems, the storsrmd daemon starts automatically for non-root or non-administrator users even if the call requires root or administrator privilege. Also on UNIX systems, the daemon can be started manually by an authorized user (root or administrator). However, for API calls on Windows systems, storsrmd needs to be running prior to a call that requires root or administrator privileges. Therefore, on Windows Systems SRM Daemon (storsrmd and storsrmd64) service can be configured to start automatically at boot time, or it can be started manually by an authorized user (root or administrator) before running any SRM API calls.

Authorized users are allowed to control daemons using the `stordaemon` command line utility and can start the SRM daemon as follows:

`stordaemon start storsrmd` (or `storsrm64` for 64-bit systems)

Note: The SRM daemon is currently supported on AIX, Sun Solaris, HP_UX, Tru64 UNIX, Linux, and Windows systems. The SRM daemon only supports local client connections. The application must run either on the same host or a different host and connect to a SYMAPI server that is local to the daemon. The application cannot communicate directly with a daemon on another host.

Non-root and non-administrative users must be defined in the `daemon_users` file to obtain authorization for using the SRM daemon and other daemon services.

Daemon authorization

Access to SRM functionality is controlled by limiting permission to the SRM daemon. This access is controlled using the common daemon authorization file, `daemon_users`. This file is located in the following directories:

UNIX	`/var/symapi/config/daemon_users`
Windows	`c:\Program Files\EMC\SYMAPI\config\daemon_users`

Note: It is important to protect this file so that only privileged administrators can modify it.

Users meeting any of the following criteria will be permitted to control and use the SRM daemon:

- Authorized users; UNIX users with root access and Windows users that are a members of the Administrators group
- Users listed in the `daemon_users` file located on each host from which they require access

For example, the following lines in the `daemon_users` file would permit users named `smith` and `jones` to use the SRM daemon:

```
smith    storsrmd
jones    storsrmd
```

Various SRM control operations for authorized local users can be defined by adding the appropriate control operation keyword to the `daemon_users` file command line in the third column as shown in the following example:

```
smith     storsrmd     file_allocate, file_extend
jones     storsrmd     file_allocate, file_extend
```

Note: Multiple control operations can be added to the line, separated by a comma.

Table 6 lists the available SRM control operations by keyword that can be added for a user. The keyword must be added to the `daemon_users` file to authorize the user to perform the specified SYMCLI control operations.

Table 6 SRM control operations

Keyword	Control operation	SYMCLI command
`file_allocate`	Allocate a file	Not available
`file_extend`	Extend a file	Not available
`file_trim`	Trim a file	Not available
`fs_flush`	Flush a file system	Not available
`fs_freeze`	Freeze a file system	Not available
`fs_thaw`	Thaw a file system	Not available
`lv_create`	Create a logical volume	`symlv create`
`lv_delete`	Delete a logical volume	`symlv delete`
`lv_resize`	Resize a logical volume	`symlv extend` `symlv reduce`
`lv_mirror_add`	Add a logical volume mirror	`symlv add`
`lv_mirror_remove`	Remove a logical volume mirror	`symlv remove`
`vg_create`	Create a volume group	`symvg create`
`vg_delete`	Delete a volume group	`symvg destroy`
`vg_device_add`	Add a device to a volume group	`symvg adddev`
`vg_device_remove`	Remove a device from a volume group	`symvg rmdev`
`vg_recover`	Recover a volume group	`symvg recover`

Note: For any directories and files being accessed for SRM control and mapping operations, operating system level permission is required. Refer to next section titled "Setting operating system level permissions".

Setting operating system level permissions

The operating system level permissions for any directories and files being accessed for the SRM control and mapping operation must be set for the user. The following examples show various administrator settings for directories and files to allow read (r), write (w) and execute (x) privileges:

Permits root user access:

```
-rwx------rootother/usr/vxfs/root.bin
```

```
Permits user smith and root access:
```

```
-rwx------smithsymapi/usr/vxfs/smithroot.bin
```

```
Permits all users access:
```

```
-rwxrwxrwxrootother/usr/vxfs/allusers.bin
```

CHAPTER 2
Data Object SRM

This chapter explains how the extents of datafiles and other objects can be displayed using SYMCLI commands.

- Overview 26
- Examining device partitions 27
- Examining device data objects 29
- Examining logical volume data objects 31
- Examining file system data objects 33

Overview

The data object commands display the extents of datafiles and objects and partition information of a physical device.

These commands are supported on the following operating systems:

- HP-UX
- IBM AIX
- Sun Solaris
- Tru64 UNIX
- Windows 2000, 2003, 2008
- Linux

Data object terminology and architecture

This section describes data object terminology and SRM architecture.

Partitioned physical devices are formed from or comprise a large SCSI device. Partitioning allows users to divide large storage devices into more manageable areas called *partitions*. Each partitioned area is named (`ppdevname`) to function as a separate device.

A *data object* is an entity such as a file, a logical volume, a file system, or a physical device used to store or back up data. A data object consists of a set of extents (see Figure 1). Within a Symmetrix® array, an object can have extents in various devices.

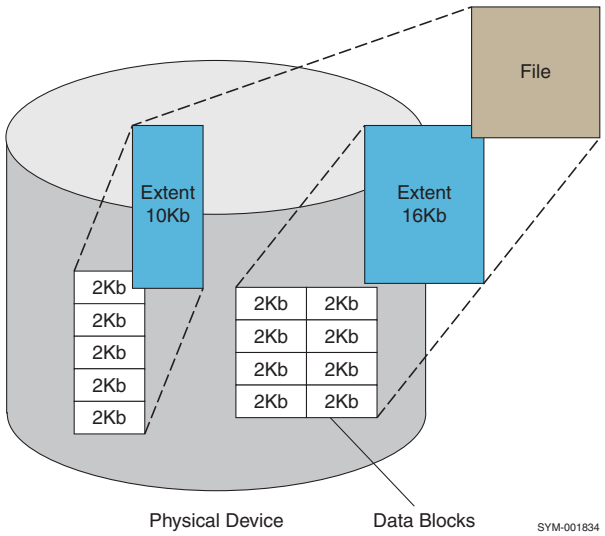

Figure 1 Objects, extents, and data blocks

An *extent* is a logical storage structure that holds a contiguous set of data blocks, which are allocated by the host system. One or more extents make up a data object.

A *data block* is a logical storage structure that is the smallest unit of storage and I/O used by the host. A number of contiguous data blocks make up an extent.

Examining device partitions

The `sympart` command displays information about partitions for host devices.

The command displays detailed information about the geometry and layout of host devices. This includes information specific to the partitions of the device such as the partition name, type, attributes, offset into the full device, and the size of the partition.

Partition offset and size information can be displayed in blocks (`-blocks`), kilobytes (`-kb`) or megabytes (`-mb`).

To list the partitions for a host physical device, use the following syntax

```
sympart show PdevName
```

The following is a sample command and output:

```
sympart show c0t0d0s0
```

```
Raw Partition Name             : /dev/rdsk/c0t0d0s0
Full Device Name               : /dev/rdsk/c0t0d0s2
Bytes Per Sector               : 512
Sectors Per Track              : 574
Tracks Per Cylinder            : 10
Sectors Per Cylinder           : 5740
Number of Cylinders            : 12437
Number of Accessible Cylinders : 12437
Partition Style : VTOC
Partition Table Size : 96 Bytes
Max Partition Capacity: 2097152m
Max Number of Partitions : 8
Partition Address overlapped : FALSE


Number of Partitions for Host Device (8):
{
-----------------------------------------------------------------------
Name                 Type               Attributes     Offset      Size
-----------------------------------------------------------------------
/dev/rdsk/c0t0d0s0   Root filesystem       None            0m    32803m
/dev/rdsk/c0t0d0s1   Swap partition        U           32803m     2049m
/dev/rdsk/c0t0d0s2   Full disk             None            0m    34852m
/dev/rdsk/c0t0d0s3   Unassigned partition  None            0b        0b
/dev/rdsk/c0t0d0s4   Unassigned partition  None            0b        0b
/dev/rdsk/c0t0d0s5   Unassigned partition  None            0b        0b
/dev/rdsk/c0t0d0s6   Unassigned partition  None            0b        0b
/dev/rdsk/c0t0d0s7   Unassigned partition  None            0b        0b
}

Legend for the Partition Attributes:

(U): Unmountable          (H): Has Hidden sectors          (A): Active
(O): Read-Only            (B): Bootable
                          (R): Recognized
                          (W): ReWritten
```

To list partition information for all of the devices connected to a host, enter:

```
sympart list
```

```
PartitionName       DeviceName          PartitionType     Offset  PartSize Status
-------------------------------------------------------------------------------
/dev/sda            /dev/sda            Full disk             0b   140014m SUCCESS
/dev/sda1           /dev/sda            Linux native p       31k      102m SUCCESS
/dev/sda2           /dev/sda            Linux native p      102m    81925m SUCCESS
```

```
/dev/sda3          /dev/sda           Swap partition    82027m     8197m SUCCESS
/dev/sdb           /dev/sdb           Full disk             0b      500m SUCCESS
/dev/sdb1          /dev/sdb           Linux native p        1b      191m SUCCESS
/dev/sdc           /dev/sdc           Full disk             0b      500m SUCCESS
/dev/sdc1          /dev/sdc           Linux native p       29k      125m SUCCESS
/dev/sdc2          /dev/sdc           Extended parti      125m      374m SUCCESS
/dev/sdc3          /dev/sdc           Unused partiti        0b        0b SUCCESS
/dev/sdc4          /dev/sdc           Unused partiti        0b        0b SUCCESS
/dev/sdc5          /dev/sdc           Linux native p      125m      125m SUCCESS
/dev/sdd           /dev/sdd           Full disk             0b      500m SUCCESS
/dev/sde           /dev/sde           Full disk             0b      500m SUCCESS
/dev/sdf           /dev/sdf           Full disk             0b      500m SUCCESS
...
```

The `sympart list` command can be filtered using the following options:

- `-count` *`FirstNMatches`* — Lists only first N matches found where FirstNMatches is the number of partitions, for example:

  `sympart list -count 3`

- `-device_type` *`DeviceType`* — Lists only partition information of a specified device type, using the following device type values:

  `sympart list -device_type` `[ SYMMETRIX | CLARIION ]`

- `-label` *`PartLabel`* — Lists only partition information of devices with a specified partition label, using the following partition label values:

  `sympart list -label` `[ MBR | GPT | VTOC | EFI ]`

- `-type` *`PartType`* — Lists only partition information of devices with a specified partition type, using the following partition type values:

  `sympart list -type`

  ```
  PART_UNASSIGNED | PART_BOOT       |
  PART_ROOT       | PART_SWAP       |
  PART_USR        | PART_FULL_DISK  |
  PART_STAND      | PART_VAR        |
  PART_HOME       | PART_ALT_SECTOR |
  PART_CACHE      | PART_RESERVED   |
  PART_VX_PUBLIC  | PART_VX_PRIVAT  |
  PART_EXTENDED   | PART_FAT12      |
  PART_FAT16      | PART_FAT32      |
  PART_FAT32_X13  | PART_HUGE       |
  PART_IFS        | PART_LDM        |
  PART_NTFT       | PART_OS2BOOT    |
  PART_PREP       | PART_UNIX       |
  PART_XENIX_1    | PART_XINT13     |
  PART_XINT13_EX  | PART_VALID_NTFT |
  PART_UNUSED     | PART_XENIX_2    |
  PART_VERSION6   | PART_SYSTEM_V   |
  PART_VERSION8   | PART_VERSION7   |
  PART_BSD_4_1    | PART_BSD_4_2    |
  PART_ADVFS      | PART_LSMPUBLIC  |
  PART_LSMPRIVAT  | PART_LSMSIMPLE  |
  PART_LSMNOPRIV  | PART_DATABASE   |
  PART_RAWDATA    | PART_DRD        |
  PART_CNX        | PART_CDFS       |
  PART_LINUX      | PART_LINUX_LVM  ]
  ```

- `-exclude` — Lists the devices that do not pass the specified filter, for example the following command will list all the devices except those with a FAT32 partition type:

  `sympart list -exclude -type PART_FAT32`

Examining device data objects

The `symrslv` command displays information about data objects down to the physical-extent level.

The command displays detailed logical-to-physical mapping information about a physical device object. Specifically, it provides details about the physical extents of these objects.

Support for VMware and AIX VIO virtualized clients was added in Solutions Enabler V7.0. This allows the `symrslv` command to resolve array devices behind the virtual disk. The host, with HBAs and Symmetrix array, creates virtualized environments, or guests. The host pools together disks and can then carve out virtual disks for guests. Solutions Enabler can now identify Symmetrix devices behind the virtual disk.

The `symrslv pd` command is used to determine which Symmetrix or CLARiiON® devices need to be examined.

To examine a host physical device, enter:

```
symrslv pd PdevName
```

The following is a sample command and output:

```
symrslv pd /dev/rdsk/c5t6d0s2
```

```
Absolute Path                 : /dev/rdsk/c5t6d0s2
Resolved Object Type          : Physical Device
Resolved Object Size          : 2m
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 1

Number of Mirrors for object (1):
{
1) Mirror Name                :
Mirror Physical Extents (1):
{
-----------------------------------------------------------------------
Size  Array  Dev   Offset PPdevName                   Offset Attr
-----------------------------------------------------------------------
2m  03122 0082       0b /dev/rdsk/c5t6d0s2              0b (S)
}
Mirror Physical Devices (1):
{
---------------------------------------------------------------------
Array  Dev  PPdevName               PdevName                Attributes
---------------------------------------------------------------------
03122 0082  /dev/rdsk/c5t6d0s2      /dev/rdsk/c5t6d0s2             (S)
}
Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

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

- `-expand` option displays multiple contiguous extents as separate extents. The default behavior is to logically collapse contiguous extents into one large extent.

- `-no_extents` option displays information about the object without the detailed extent information.
- `-phys_collapse` option displays physically contiguous extents as one larger extent. The list of extents returned are not logically consistent.
- `-pdev_extents` option displays extents at the physical device (`pdev`) level. It will not expand extents at the underlying metadevice level.

Examining logical volume data objects

The `symrslv` command displays the data mapping attributes and the extents of a specified logical volume (of the LVM).

The `symvg` and `symlv` commands can be used to determine which logical volume to examine.

To examine a logical volume, enter:

```
symrslv -g VgName lv LvolName
```

For example, to display information about logical volume `/dev/BigLV`, enter:

```
symrslv -g /dev/BigVG lv BigLV
```

The following is sample output from this command:

```
Absolute Path                 : /dev/BigVG/rBigLV
Resolved Object Type          : HP-UX LVM Logical Volume
Resolved Object Size          : 10000m
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 1

Number of Mirrors for object (1):
{
1) Mirror Configuration

Mirror Physical Extents (3):
----------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                   Offset
----------------------------------------------------------------------
4314m  03003 0123       1m /dev/rdsk/c2t1d5                1m (M)
4315m  03003 0124       0b /dev/rdsk/c2t1d5             4315m (m)
1370m  03003 0125       0b /dev/rdsk/c2t1d5             8631m (m)

Mirror Physical Devices (3)
{
----------------------------------------------------------------------
Array Dev   PPdevName               PdevName
----------------------------------------------------------------------
03003 0123  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5           (M)
03003 0124  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5           (m)
03003 0125  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5           (m)
}
Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

This output shows that the physical device underlying the logical volume is a Symmetrix metadevice. It also lists the attributes for the logical volume, mirror physical extents, and mirror physical devices.

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

- `-expand` option displays multiple contiguous extents as separate extents. The default behavior is to logically collapse contiguous extents into one large extent.

- `-no_extents` option displays information about the object without the detailed extent information.
- `-phys_collapse` option displays physically contiguous extents as one larger extent. The list of extents returned are not logically consistent.
- `-pdev_extents` option displays extents at the physical device (`pdev`) level. It will not expand extents at the underlying metadevice level.

Examining file system data objects

The `symrslv` command lists the data mapping attributes and the extents of a data object specified as a file system or a regular datafile.

The `symhostfs` command can be used to determine which file system to examine.

To display a file system, enter:

```
symrslv fs FileSystemName
```

To display a datafile, enter:

```
symrslv file FileName
```

To display a directory, enter:

```
symrslv dir Directory
```

The command lists the following file attributes:

- Absolute pathname (full pathname)
- Object type determined by the specific platform
- Object size in blocks
- Number of trailing bytes in the last block
- Number and list of device names where the file or file system resides
- File system mount point
- File system device name
- Block offset of each extent
- Size in blocks of each extent

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

- `-expand` option displays multiple contiguous extents as separate extents. The default behavior is to logically collapse contiguous extents into one large extent.
- `-no_extents` option displays information about the object without the detailed extent information.
- `-phys_collapse` option displays physically contiguous extents as one larger extent. The list of extents returned are not logically consistent.
- `-pdev_extents` option displays extents at the physical device (`pdev`) level. It will not expand extents at the underlying metadevice level.

Refer to Chapter 4, “File System SRM,” for more information on file systems.

CHAPTER 3
Database SRM

This chapter describes how the elements of datafiles and other objects can be displayed using SYMCLI commands.

- Overview 36
- Establishing the database connection 48
- Using database daemons 53
- Listing database instances 56
- Examining database files 57
- Translating database devices to Symmetrix groups 59
- Examining tablespaces 60
- Translating tablespace devices to Symmetrix groups 63
- Examining schemas 64
- Examining tables 66
- Examining segments 67
- Invoking database I/O control 69
- Using the EMC Oracle ASM library 71

Overview

The database commands display database mapping and characteristics.

The commands list attributes that describe databases structures, files, tablespaces, and user schemas. Typically, the database commands described in this chapter work with Oracle, Informix, SQL Server, Sybase, Exchange Server, SharePoint Portal Server, and DB2/UDB database applications. For information about the most current databases and host platforms supported, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Database architecture and terminology

A relational database server is the basis for information management (RDBMS), which reliably manages a large amount of data in a multiuser environment. The server must allow many users concurrent access to the same data without degradation of performance. A database server must also prevent unauthorized access and provide efficient solutions for failure recovery.

A *database instance* or an *Oracle server instance* refers to a set of database operating system processes, or threads, running on a host. The RDBMS also allocates and manages an area of host memory for caching data pages from disk. Some RDBMS products allow multiple instances to execute concurrently on the same host, each accessing its own physical database.

A database has both physical and logical structures. In the following text, terminology relating to SYMCLI database mapping is described for various RDBMS products.

Oracle architecture

In Oracle, a *schema* is a collection of logical database objects that are available to support a specific user's realm and concept of the data in a common database. Schema objects are logical structures that directly refer to the database's data.

Typical Oracle logical schema objects are:

- Tables
- Indexes
- Views
- Clusters
- Database links

There is no relationship between tablespace and schema. Objects in the same schema can be in different tablespaces.

In Oracle, a *table* is the basic object and unit of storage, which is a collection of data suitable for quick reference. Each table is a data structure defined with a table name and a set of columns and rows with data occupying each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.

Partitioned tables are typically used when a table has grown too large for a single object. Partitioning allows users to divide massive table data into more manageable pieces called *partitions*. Each partitioned table is stored in a separate segment. Optionally, each partition can be stored in a separate tablespace, which has the following advantages:

- Contains the impact of any damaged data
- Allows for back up and recovery each partition independently
- Balances the I/O load by mapping partitions to different disk drives

A *tablespace* is a named storage pool that physically allocates space for the database files. As shown in Figure 2, one or more table and index structures make up the database files of a tablespace. The data is stored logically in tablespaces, and physically in datafiles that are associated with the corresponding tablespace.

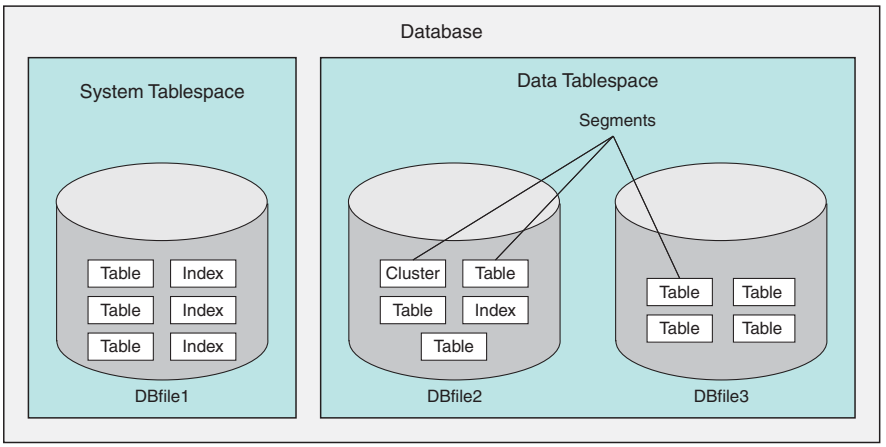

Figure 2 Oracle database architecture

Note: Every Oracle database contains a tablespace named SYSTEM, which Oracle creates automatically when the database is created. The SYSTEM tablespace always contains the data dictionary tables for the entire database.

A *segment* is a set of extents that contains all the data blocks for a specific logical storage structure or object within a tablespace (see Figure 3 on page 38). For example, one or more extents are allocated to each table and to each associated index. A segment and all its extents are stored in one tablespace. Within a tablespace, a segment can span datafiles or have extents with data from more than one file. There are four basic types of segments:

- Table segment — partitioned, nonpartitioned, or clustered
- Index segment — an index
- Temporary segment — a workspace (for SQL query processing)
- Roll back segment — one or more for each database (holds old values to allow rolling back a transaction)

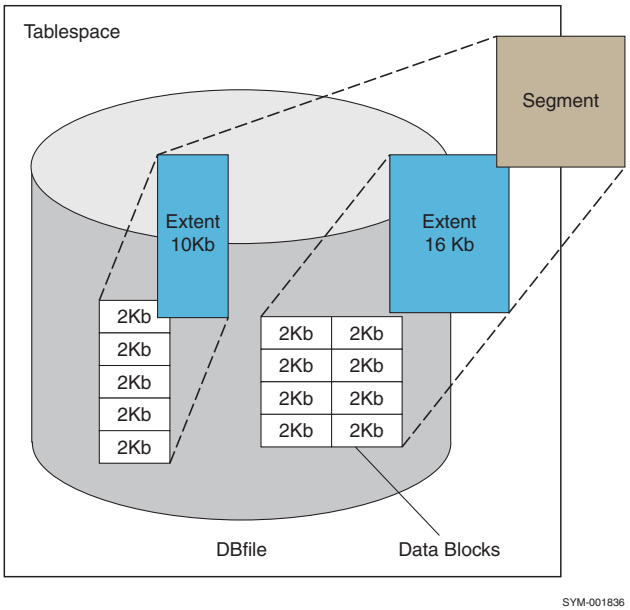

Figure 3 Segments, extents, and data blocks

An *extent* is a logical storage structure that holds a contiguous string of data blocks, which are allocated by the Oracle processes (or an RDBMS) for the management of a particular database file. One or more extents make up a segment.

A *data block* is a logical storage structure that is the smallest unit of storage and I/O used by the database. When the database is created, a block size is specified in bytes. A number of contiguous data blocks make up an extent.

An *index* is an optional structure associated with a table that increases the data retrieval performance. Like large tables, large indexes can be partitioned to make them more manageable. Indexes are created on one or more columns of a table. Indexes are useful when an application often needs to make queries to a table for a range of rows or a specific row. Indexes are logically and physically independent of the table data and can, therefore, be dropped at any time.

A *cluster* is an optional method of storing table data. Clustered tables are a group of tables physically stored together using the same data blocks because they share common columns and are often used together. Disk access time can improve for many joint operations. Because the row data is physically stored together, data access performance improves.

A *view* is a custom-tailored presentation of the data contained in one or more tables. A view takes the SQL sequence of a query, stores it, and treats it as a table. This allows a view to be conceptually thought of as a "stored query."

A *database link* is a named object that describes a path from one database to another. Database links are implicitly used when a reference is made to a global object name in a distributed database.

A *data dictionary* is required for each database and contains a set of tables and views that provide a read-only reference about the database. It stores information about the logical and physical structure of the database. It contains a list of valid database users, table integrity constraints, and space allocation parameters for a schema object.

SQL Server architecture

An SQL Server database is a collection of logical objects directly referring to the data in the database. Many databases are possible in a SQL Server database instance. Typically, in any SQL Server database instance, there are four system databases and one or more user databases. Each database has a specific owner that can grant or revoke permissions to other users.

Typical SQL Server logical database objects are:

- Tables
- Indexes
- Views

The database *owner* is associated with a user within a database instance. All permissions and ownership of objects in the database are controlled by the owner's user account. Typically, the owner of an SQL Server database is referred to as user `dbo`. This means user *xyz* of database `foo` and user *abc* of database `foobar` are both referred to as user `dbo` for their respective databases.

In SQL Server, a *table* is the basic object and unit of storage, which is a collection of data suitable for quick reference. Each table is a data structure defined with a table name, and a set of columns and rows with data occupying each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.

In SQL Server, an *index* is an optional structure associated with a table that increases the data retrieval performance. Indexes are created on one or more columns of a table. Indexes are useful when an application often needs to make queries to a table for a range of rows or a specific row. Indexes are logically and physically independent of the table data and can, therefore, be dropped at any time.

A *view* is a custom-tailored presentation of the data contained in one or more tables. A view takes the SQL sequence of a query, stores it, and handles it as a table. This allows a view to be conceptually thought of as a "stored query."

A *filegroup* is a named storage pool that physically allocates space for the database files. As shown in Figure 4 on page 40, one or more table and index structures make up the database files of a filegroup. The data is stored logically in filegroups and physically in datafiles that are associated with the corresponding filegroups.

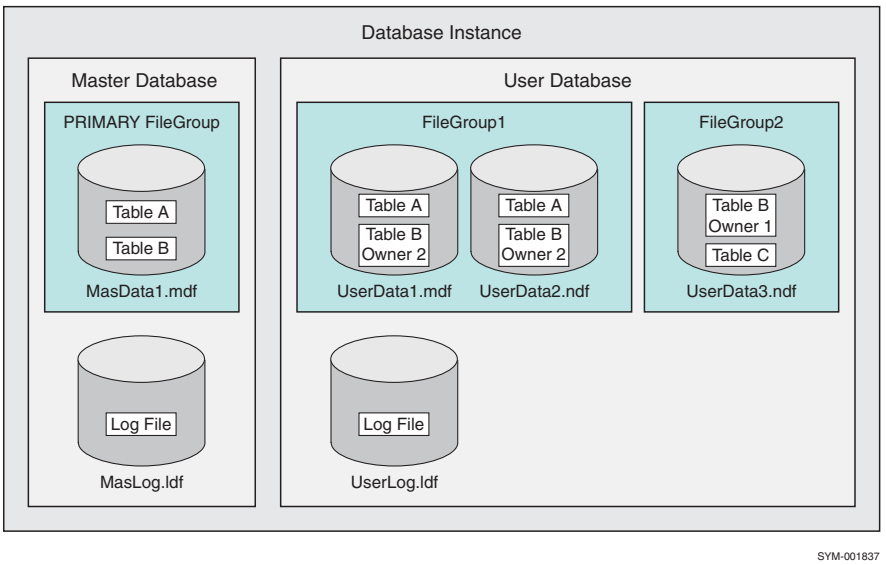

Figure 4 SQL database architecture

Note: Every SQL Server database instance contains four system databases named `master`, `tempdb`, `msdb`, and `model`, which an SQL Server creates automatically when the database instance is created. The `master` database contains the data dictionary tables for the entire database instance.

Filegroups are divided into logical units of storage called *extents*. An extent is a set of data blocks that contains all or part of the data for a specific logical storage object within a database. One or more extents are allocated to each table and to each associated index. An extent can span datafiles within any filegroup.

An *extent* is a logical storage structure that holds a contiguous string of data blocks, which are allocated by the RDBMS processes for the management of a particular database file.

A *data block* is a logical storage structure that is the smallest unit of storage and I/O used by the SQL Server database. When the database is created, a block size is specified in bytes. A number of contiguous data blocks make up an extent.

Sybase architecture

A database *owner* is associated with a user within a Sybase database instance. All permissions and ownership of objects in the database are controlled by the owner's user account. Typically, the owner of a Sybase database is referred to as user `dbo`. This means user *xyz* of database `foo` and user *abc* of database `foobar` are both referred to as user `dbo` for their respective databases.

In Sybase, a *table* is the basic object and unit of storage, which is a collection of data suitable for quick reference. Each table is a data structure defined with a table name and a set of columns and rows with data occupying each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.

In Sybase, an *index* is an optional structure associated with a table that increases the data retrieval performance. Indexes are created on one or more columns of a table. Indexes are useful when an application often needs to make queries to a table for a range of rows or a specific row. Indexes are logically and physically independent of the table data and can, therefore, be dropped at any time.

A *view* is a custom-tailored presentation of the data contained in one or more tables. A view takes the SQL sequence of a query, stores it, and handles it as a table. This allows a view to be conceptually thought of as a "stored query."

An *instance* is a collection of datafiles, known as devices. There can be one or more devices defined for each Sybase instance. Databases are created and defined on pieces of one or more devices. The administrator decides which devices to use for a particular database at database create time.

As shown in Figure 5 on page 41, one or more table and index structures make up a device of a database. The data is stored logically in databases and physically in devices that are associated with the corresponding database.

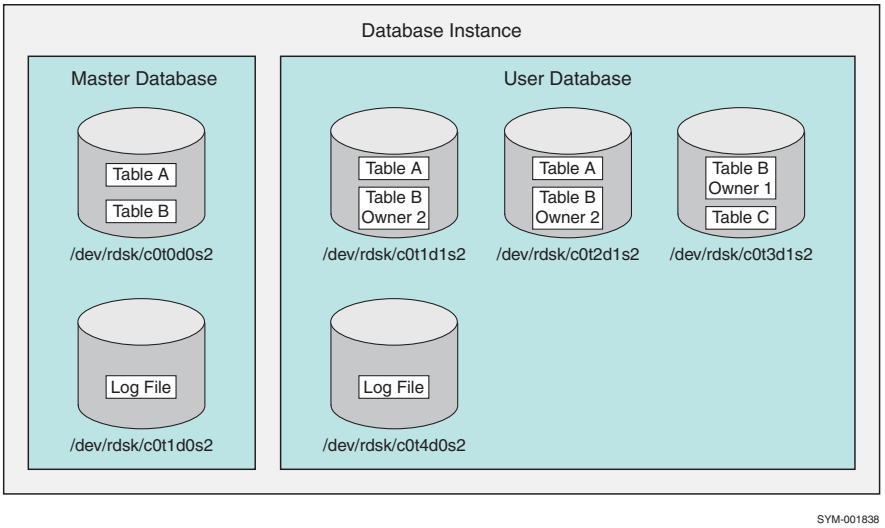

Figure 5 Sybase database architecture

Note: Every Sybase instance contains four system databases named `master`, `tempdb`, `sybsystemprocs`, and `model`, which Sybase creates automatically when the database instance is created. The `master` database contains the data dictionary tables for the entire database instance.

An *extent* is a logical storage structure that holds a contiguous string of data blocks, which are allocated by the RDBMS processes for the management of a particular Sybase database file.

A *data block* is a logical storage structure that is the smallest unit of storage and I/O used by the Sybase database. When the database is created, a block size is specified in bytes. A number of contiguous data blocks make up an extent.

Informix architecture

In Informix, a *database* is a collection of logical objects (tables, indexes, locator objects, views, functions, etc.), tablespaces, and logical and physical logs. There can be many databases in an Informix server instance. A database can be defined on one or more tablespaces. In an Informix database instance there are typically two system databases and one or more user databases. A database has a specific owner that can grant or revoke permissions for other users.

In Informix, a *tablespace* is a named storage pool that physically allocates space for the database *chunks* (datafiles). A tablespace can be defined by one or more chunks. As shown in Figure 6, one or more table and index structures make up the chunks of tablespace. The data is stored logically in tablespaces and physically in chunks that are associated with the corresponding tablespaces.

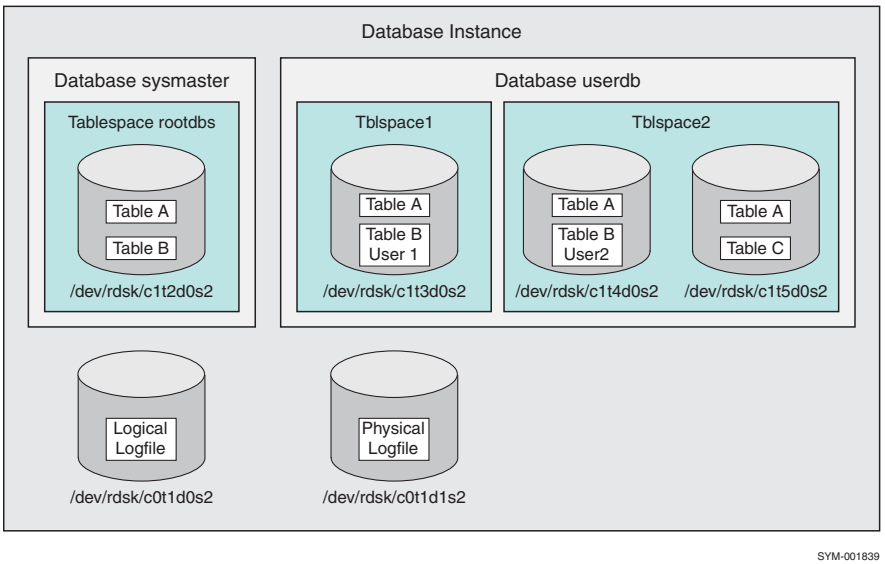

Figure 6 Informix database architecture

Note: Every Informix database instance contains two system databases (named `sysmaster` and `sysutils`), which an Informix server creates automatically when the database instance is created. The `sysmaster` database contains the data dictionary tables for the entire database instance.

Chunks are divided into logical units of storage called *extents*. An extent is a logical storage structure made up of a contiguous string of data blocks that contain all or part of the data for a specific logical storage object within a database. One or more extents are allocated to each table and to each associated index. An extent can span chunks within a data space. The data blocks in the extent are allocated by the RDBMS processes for the management of a particular database file.

A *data block* is a logical storage structure that is the smallest unit of storage and I/O used by the database. A number of contiguous data blocks make up an extent.

A *table* is the basic object and unit of storage, which is a collection of data used for quick reference. Each table is a data structure defined with a table name and a set of columns and rows with data occupying each cell. A row corresponds to a single record.

In Informix, *fragmented tables* are typically used when a table has grown too large for a single object. Fragmentation allows users to divide massive table data into more manageable pieces. There are several ways to fragment a table including round-robin, hash, expression, and hybrid.

An *index* is an optional structure associated with a table to increase the performance of data retrieval. Indexes are created on one or more columns of a table. Indexes are useful when an application often needs to make queries to a table for a row or range of rows. Indexes are logically and physically independent of table data and can be discarded at any time. An index can be fragmented similarly to tables. Additionally, the index can be detached (reside in different tablespaces) from the data with which it is associated.

A *view* is a custom-tailored presentation of the data contained in one or more tables. A view takes the SQL sequence of a query, stores it, and handles it as a table. This allows a view to be conceptually thought of as a *stored query*.

IBM DB2/UDB architecture

In IBM DB2/UDB, an *instance* (also called a database manager) comprises DB2/UDB code and its associated data structures that manage the storage and access of data. Each instance is created and managed by a separate operating system user ID. The home directory for the user ID contains files that describe the instance. Multiple instances can coexist on the same host server. The default user ID is `db2inst1`. However, a typical host server running multiple V6.1 and V7.1 instances could have user IDs such as `db2v7in2`, and `db2v8in1`.

In DB2/UDB, a *database* is a collection of logical objects, configuration files, and recovery logs. There can be many databases in a DB2/UDB database instance. When and instance is created, it contains no databases. A database has a specific owner that can grant or revoke permissions to other users.

Typical DB2/UDB logical database objects are:

- Tables
- Indexes
- Views
- Functions
- LOBs

A *tablespace* is a place to store logical objects. A tablespace can be either system managed space (SMS) or database managed space (DMS). For an SMS tablespace, each container is a directory in the file space of the operating system. The operating system's file manager controls the storage space. For a DMS tablespace, each container is either a fixed size preallocated file or a physical device such as a disk. The database manager controls the storage space. Each tablespace can be spread over one or more containers.

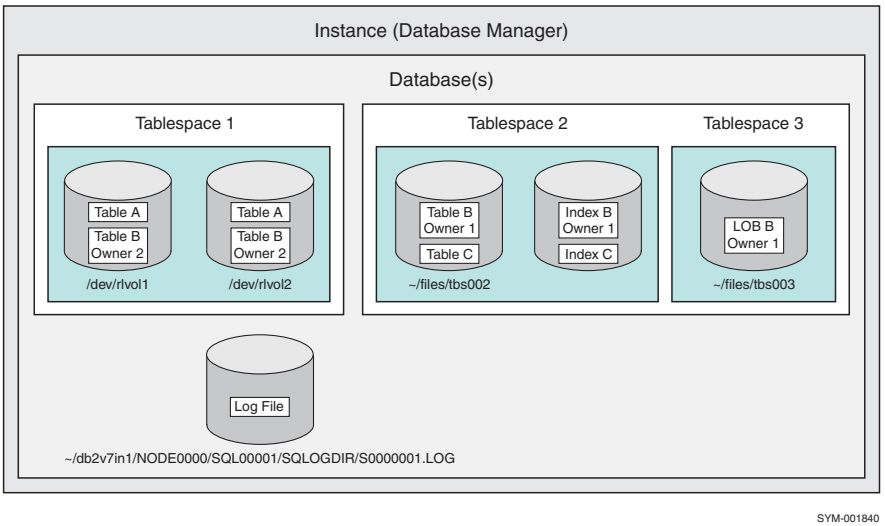

Figure 7 DB2/UDB database architecture

Each database must contain at least three tablespaces:

- Catalog (SYSCATSPACE)
- User (USERSPACE1)
- Temporary (TEMPSPACE1)

Additional table spaces are typically created to contain tables, indexes, and LOBs.

A *table* is the basic object and unit of storage, which is a collection of data used for quick reference. Each table is a data structure defined with a table name and a set of columns and rows with data occupying each cell. A row corresponds to a single record. An *index* is an optional structure associated with a table to increase the performance of data retrieval. Indexes are created on one or more columns of a table. Indexes are useful when an application often needs to make queries to a table for a row or range of rows. Indexes are logically and physically independent of table data and can be discarded at any time.

A *view* is a custom-tailored presentation of the data contained in one or more tables. A view takes the SQL sequence of a query, stores it, and treats it as a table. This allows a view to be conceptually thought of as a *stored query.*

Containers are divided into logical units of storage called extents. An *extent* is a contiguous string of data blocks that contains all or part of the data for a specific logical storage object within a database. One or more extents are allocated to each table and to each associated index. DB2/UDB spreads extents across containers using a round-robin allocation policy.

A *data block* is a logical storage structure that is the smallest unit of storage and I/O used by the database subsytem. When the database subsystem is created, a block size is specified in bytes. A number of contiguous data blocks make up an extent.

Exchange Server architecture

Exchange Server allows two types of databases, a mailbox store and a public store. An Exchange *mailbox store* is a binary tree that consists entirely of electronic mail data. This data is logically, but not physically grouped into mailboxes. The actual mailbox objects and associated metadata are maintained in the Windows Active Directory, where they are stored as attributes on a Windows user object. An Exchange *public store* is almost identical to the mailbox store, except that all of the data is organized logically into a folder hierarchy instead of mailboxes.

From the top down, as shown in Figure 8, an Exchange Server is a single *instance* of Exchange consisting of storage groups and databases. Each Exchange Server is allowed a maximum of four storage groups. Within SYMCLI, an Exchange storage group will be referred to as an Exchange tablespace. Each Exchange storage group consists of anywhere from one to five databases and a set of transaction logs. All of the databases in a storage group share the same set of transaction logs. All data is written to the transaction logs first, and then placed in memory, and finally committed to the datafiles on disk. A checkpoint file (whose name is a variation of `E0n.chk`, where *n* is from 0 to 3) is maintained for each storage group to track which transactions have been committed to their respective databases, and which have not been committed.

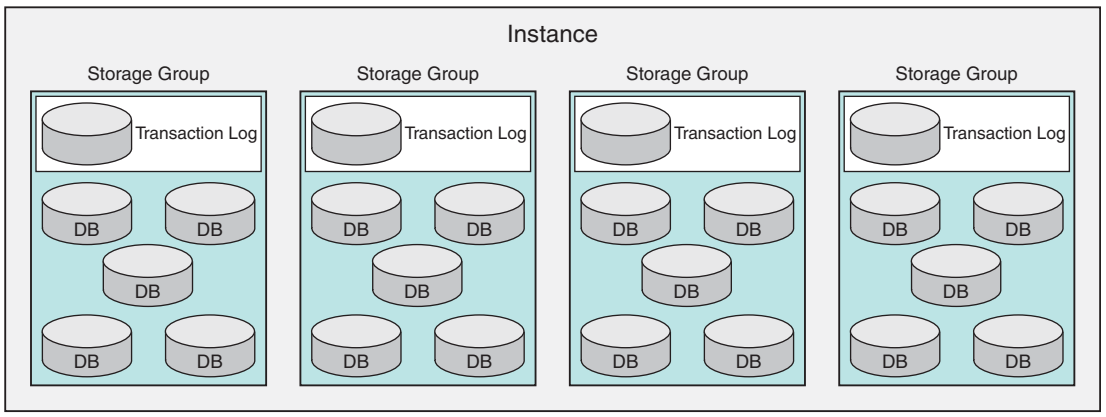

Figure 8 Exchange database architecture

Different databases in different storage groups can have the same name. To make Exchange database names unique within SYMCLI, they should be in the form *storagegroup\database.* For example, if storage group `SG1` and `SG2` both have a database `DB1`, the database names in SYMCLI should be `SG1\DB1` and `SG2\DB1`, respectively. Because the storage group name is specified with the database name, commands that pass both the database name (using the `-db` option) and the storage group name (using the `-tbs` option) will return as invalid.

An Exchange database consists of two datafiles, a binary datafile, and a streaming file. The binary datafile has an extension of `.edb` and is laid out as a binary tree. The streaming file has an extension of `.stm` and is laid out flat, in 64 KB runs. The binary datafile is the traditional datafile that stores the majority of typical email data. The streaming file is a new file designed to store native Internet content in large, sequential chunks. The sequential layout of the streaming file allows for faster retrieval of Internet content such as movies, sound files, and pictures.

Exchange databases can be brought online and offline independent of the other databases on the server. This process is referred to as *mounting* and *dismounting* the database. The Solutions Enabler SYMCLI provides control functions for the mounting and dismounting of databases.

EMC Solutions Enabler treats both the transaction log files and the database datafiles as database files. Be aware of this when querying the Exchange Server for the files in a given database. Each file returned will be tagged with its appropriate type, either data or log.

There is no concept of the following database objects within Exchange: a schema, a table, or a segment. The concept of both an extent and a data block is maintained through the underlying file system, NTFS, rather than Exchange itself. Exchange data blocks are always 4 KB.

SharePoint Portal Server

SharePoint Portal Server is a web portal server that integrates document management and search capabilities into a single server product. Because SharePoint Portal Server runs on top of Exchange Server, the architecture for SharePoint Portal Server is identical to that of Exchange Server. Specifically, SharePoint Portal Server 2001 is restricted to using a single storage group and a single public store.

Generic SafeWrite

For generic applications, **EMC Double Checksum for Generic Applications** provides the Generic SafeWrite feature to help protect critical applications from incurring an incomplete write, and subsequent torn page, due to a failure with a component connected to the Symmetrix Front End Channel Adapter. Generic SafeWrite is most often used to protect against corruption from HBA and link failures, including server crashes, where essentially, it will help protect against fractured writes that can occur before the data reaches the Symmetrix array.

A Relational Database Management System (RDBMS), such as Microsoft Exchange, structures data within database files using pages (also referred to as blocks). Pages within a database are the smallest allocation unit size possible for a database object (such as a table or a row). For example, the page size for Microsoft Exchange is 4 KB and for Oracle, although it can be configured, is usually set to 8 KB. If an incomplete page is written to a database file, a corruption to the database will occur. The resulting corruption is commonly referred to as a *torn page*.

Torn pages are only detected by most RDBMSs after the corruption has been written, when that area of the database is read, which could be long after when the corruption was introduced. In general, the only recovery from a torn page is to perform a restore from a backup (some RDBMSs allow page level restores, while others require a complete database restore). Torn pages can occur due to failures in various components that lie between the RDBMS and the storage array. Some of these components include the Operating System, File System, Logical Volume Manager, I/O Driver, Host Bus Adapter, Fibre or SCSI link and Storage Adapter.

EMC Double Checksum for Generic Applications uses the Generic SafeWrite feature to help protect critical applications from incurring incomplete writes, and subsequent torn pages, due to a failure with a component connected to the Symmetrix Front End Adapter. Most often, Generic SafeWrite will be used to protect against corruption that occurs when the HBA and link fails (including server crashes). In this scenario, Generic SafeWrite will protect against fractured writes that occur before the data reaches the Symmetrix array.

Note: Generic SafeWrite has been created to be used with RDBMSs. Applications intended for use with Generic SafeWrite include, but are not limited to Microsoft Exchange, Microsoft SQL Server, DB2/UDB and Oracle. Generic SafeWrite is not intended for use with applications where torn pages are not a concern, such as fileshares or FTP servers.

For more information on enabling and using the Generic SafeWrite feature, refer to *Implementing Generic SafeWrite for generic applications* in the *EMC Solutions Enabler Array Controls CLI Product Guide*.

Establishing the database connection

Setting UNIX environment variables

Before any SYMCLI database command session is started, UNIX environment variables must be defined on the host system. Table 7 lists the variables for each database type.

Table 7 UNIX database environment variables

For database	Set variables
ALL	LD_LIBRARY_PATH[a] LD_LIBRARY_PATH_64[b] SHLIB_PATH[c] LIBPATH[d]
Oracle	ORACLE_HOME ORACLE_SID PATH
Informix Dynamic Server	INFORMIXDIR ONCONFIG INFORMIXSERVER PATH
Sybase	SYBASE SYBASE_ASE[e] DSQUERY[f] PATH SYBSE_OCS
IBM DB2/UDB	DB2INSTANCE

a. Set for Solaris and Tru64 UNIX and point to the DB client libraries.

b. Set for Solaris 64 bit and point to the DB client libraries (only if you need to override LD_LIBRARY_PATH).

c. Set for HP-UX and point to the DB client libraries.

d. Set for AIX and point to the DB client libraries.

e. Set for Sybase startup/shutdown DB; when set during daemon autostart, all other applications that use the daemon must also set the SYBASE_ASE environment variable.

f. Must point to the correct Sybase server, i.e., Sybase Monitoring server or Sybase Adaptive server. For statistics, it must point to the Sybase Monitoring server.

For more specific information about setting these variables for your system platform and database, see your System Administrator.

Connecting to the database

A connection must be established between the database and the database commands to allow access to the database. To connect to the database, set `SYMCLI_RDB_CONNECT` to your username and password. When working with one type of database, use `SYMCLI_RDB_TYPE` to save keystrokes with a set of commands.

Note: For a nonroot user on Solaris, the Sybase database daemon must be manually started from the root user before calling any Sybase database functions.

If the connection is remote, a network service name must be supplied with the password.

For SYMCLI mapping access to an Oracle database, the user must have one of the following:

- "Select any table" privilege
- DBA role
- SYSDBA role

To invoke control access of an Oracle database, the user must have administrator privileges.

Database startup options

The database control commands allow for start up and shut down of the specified database server manager instance. With the SYMCLI client/server capability, the database manager instance start up or shut down can be done on the client side. This removes the limitation of most databases, that only allows startup and shutdown of their database manager instance on the database server side.

The CLI command `symrdb startup -type DbType` supports Sybase, Oracle, SQL Server, and DB2/UDB databases.

Note: Database startup and shutdown options are not available for Informix databases.

Sybase

The following options are available for Sybase databases:

```
symrdb startup -type Sybase -f runserver_file [-m] [-t delay_time]
```

where:

runserver_file — Specifies the absolute path name of a runserver file used as a reference each time a Sybase server is restarted.

`-m` — Starts the database in single user mode.

delay_time — Specifies the estimated time to startup the Sybase server in seconds.

SQLServer

The following startup options are available for SQLServer databases:

```
symrdb startup -type SqlServer -s instance [-c] [-f] [-m] [-n] [-x]
                    [-p master_file_path] [-e error_log_path]
                    [-l master_log_path] [-g virtual_addr_space]
                    [-t trace_number]
```

where:

instance — Instance name to be started.

master_file_path — Master database file.

error_log_path — Error log file.

master_log_path — Master database log file.

virtual_addr_space — Amount of virtual address space in megabytes.

trace_number — Trace number.

DB2/UDB

The following startup options are available for DB2/UDB databases:

```
symrdb startup -type IBMUDB [-c computer]
[-n node [ADDNODE -u hostname -p port
[-nt netname] [NODE | CATALOG -tsn tablespace_node]] |
[RESTART [-u hostname] [-p port] [-nt netname]] |
[STANDALONE]]
```

where:

ADDNODE — Issues the ADD NODE command.

RESTART — Issues the RESTART DATABASE command.

STANDALONE — Start the node in STANDALONE mode.

NODE — Specifies that the containers for the temporary tablespaces should be the same as those for the specified node.

CATALOG — Specifies that the containers for the temporary tablespaces should be the same as those for the catalog node of each database.

profile — Specifies the name of the profile.

node — Specifies the node number.

hostname — Specifies the system name.

port — Specifies the port number.

netname — Specifies the net name.

tablespace_node — Specifies the node number from which the temporary tablespace definitions should be obtained.

computer — Specifies the computer name.

Oracle

The following startup options are available for Oracle databases:

```
symrdb startup -type Oracle [[FORCE][RESTRICT][PFILE=FileName][QUIET]
                      [MOUNT [-db DbName] |
                      [ [OPEN | OPEN_READ_ONLY | OPEN_READ_WRITE |
                     OPEN_READ_WRITE_RECOVER | RECOVER | OPEN_RECOVER]
                      [-db DbName] ] | NOMOUNT]] |
                      [ [PFILE=FileName] MIGRATE [QUIET]]
```

where:

FileName — Specifies the file to be used while starting up the instance.

DbName — Specifies the database name to mount or open. Refer to the Oracle document for the definition.

`FORCE` — Shuts down the current Oracle instance (if it is running) with the shutdown option ABORT, before restarting it.

`RESTRICT` — Allows only Oracle users with the RESTRICTED SESSION system privilege to connect to the database.

`QUIET` — Suppresses the display of the System Global Area information for the starting instance.

`MOUNT` — Mounts the database but does not open it.

`OPEN` — Mounts and opens database.

`NOMOUNT` — Causes the database not to be mounted upon instance startup.

`MIGRATE` — Starts the database in OPEN MIGRATE mode and sets system initialization parameters to specific values required to enable the database upgrade or downgrade scripts to run.

`OPEN_RECOVER` — Opens the database, and performs media recovery, if necessary, before starting the instance.

`OPEN_READ_ONLY` — Specifies READ ONLY to restrict users to read-only transaction, preventing them from generating redo logs.

`OPEN_READ_WRITE` — Specifies READ WRITE to open the database in read/write mode, allowing users to generate redo logs. This is the default.

`OPEN_READ_WRITE_RECOVER` — Specifies READ WRITE to open the database in read/write mode, and specifies that media recovery should be performed, if necessary, before starting the instance.

Database shutdown options

The CLI command `symrdb shutdown -type DbType` supports Oracle, SQLServer, Sybase, and DB2/UDB databases.

Note: Database startup and shutdown options are not available for Informix databases.

Sybase

The following shutdown options are available for Sybase databases:

```
symrdb shutdown -type Sybase [-f srvname] [-w {wait | nowait}]
```

where:

srvname — Specifies the logical name by which the backup server is known in the server's system table.

wait — Brings the server down in an orderly fashion.

nowait — Shuts the server down immediately.

Prior to using these commands, the database software must be installed properly and the database instance must be created.

SQLServer

The following shutdown options are available for SQLServer databases:

```
symrdb shutdown -type SqlServer -s instance
```

where:

instance — Specifies the instance name to be shut down.

DB2/UDB

The following shutdown options are available for the DB2/UDB database:

```
symrdb shutdown -type IBMUDB [FORCE | DROP]
  [DROP_ACT | CONTINUE | TERMINATE] [-f profile] [-n node]
```

The shutdown options for IBM DB2/UDB:

FORCE — Issues the FORCE APPLICATION (ALL) command.

DROP — Drops the node from the db2nodes.cfg file.

DROP_ACT — Identifies an initial call.

CONTINUE — Identifies a subsequent call. Continue processing after a prompt.

TERMINATE — Identifies a subsequent call. Terminate processing after a prompt.

profile — Specifies the name of the profile.

node — Specifies the node number.

Oracle

The following shutdown options are available for Oracle databases:

```
symrdb shutdown -type Oracle [ABORT | IMMEDIATE | NORMAL | TRANSACTIONAL
[LOCAL]]
```

where:

ABORT — Proceeds with the fastest possible shutdown. Does not wait for calls to complete or users to disconnect.

IMMEDIATE — Does not wait for current calls to complete, prohibits further connects, and closes and dismounts the database. Finally, shuts down the instance. Does not wait for connected users to disconnect. Does not require instance recovery on next startup.

NORMAL — Waits for currently connected users to disconnect from the database, prohibits further connects, and closes and dismounts the database. Finally, shuts down the instance. Does not require instance recovery on next startup. NORMAL is the default option.

TRANSACTIONAL — Performs shutdown of an instance while minimizing interruption to clients. No client can start a new transaction on the instance.

LOCAL — Specifies a transactional shutdown on the local instance only.

Using database daemons

EMC Solutions Enabler provides database utilities called *daemons*. This section describes what daemons are, explains why they are useful, and provides an overview about how to use them.

What are daemons?

A daemon is a process, or service, that helps an application manage database calls. The daemon service facilitates communication between applications using the SYMCLI for database mapping/control and one or more database servers.

The connection between an application and a daemon is local. This means that the application and daemon are either on the same host, or the application is on a host that connects to a SYMAPI server that resides on the same host as the daemon, as shown in Figure 9. A single daemon can support connections to multiple instances/databases.

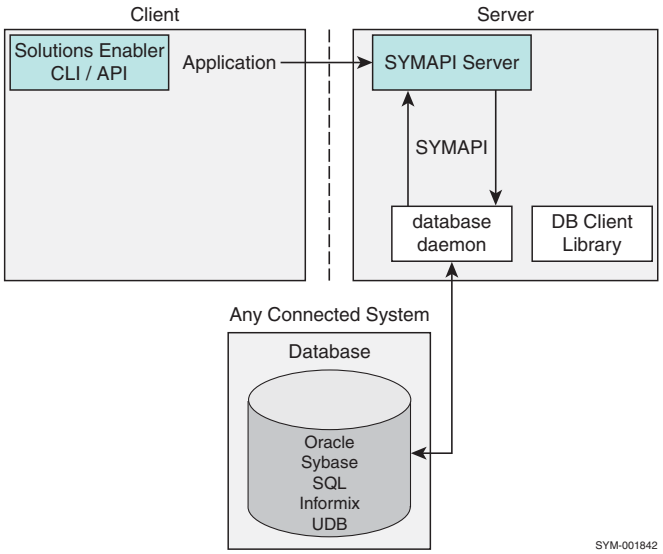

Figure 9 System view of the daemon process

During the connection phase, any database environment variables are propagated from the application to the daemon (or for remote connections, to the SYMAPI server), allowing the daemon to set the variables.

Why use daemons?

The daemons improve the speed of database mapping operations by using a persistent database connection, a fast communication mechanism, and parallel operations. Also, by running a daemon as a privileged user, the application does not have to run as a privileged user.

Current applications require no source code changes. Applications use the daemon communication mechanism when the daemon service is running, and the database engine binaries mechanism when the daemon service is not running.

Running database daemons

On the Windows platform, a database daemon should be run as a service. After the service is installed, it appears in the service list dialog box and can be accessed by clicking the services icon in the control panel (Windows NT) or the administrative tools icon (Windows 2000). The service does not restart at boot time unless the startup options are modified through the service list dialog box.

The database daemons automatically start when the application starts a database instance. Subsequent commands to the database will cause applications to use the daemon that was started. This is the default behavior, however the `stordaemon` command can be used to start and stop daemons. The APIs will use the daemon if it has been already been manually started.

A daemon can be manually started and stopped at any time. If an application is already connected using the daemon and the daemon is stopped, the application continues to run (using the database engine binaries) but may experience degraded performance. If an application is already connected without the daemon and the daemon is started, the application automatically starts using the daemon.

Note: If the database server is shutdown while a daemon is running, the daemon must be restarted to reconnect to the server.

A daemon can be started when the system is booted, and stopped when the system is shut down. The `stordaemon` command can be used to start, stop, or query the status of daemons. The `stordaemon start` command requires that the name of the daemon be specified. For example, to start the daemon for an Oracle database, enter the following:

```
stordaemon start stororad
```

By default, the `stordaemon` command waits 30 seconds to verify that the daemon is running. To override this, enter:

```
stordaemon start stororad -wait 0
```

Similarly, the `stordaemon shutdown` command stops a daemon.

It is necessary to set the LD_LIBRARY_PATH environment variable, or its equivalent counterpart environment variable, for all database daemons except Oracle and SQL Server. See Table 7 on page 48 for the list of environment variables.

The database daemons use the Solutions Enabler libraries to communicate with the application, and the DBMS client libraries to communicate with the database, as shown in Figure 10 on page 55. These libraries can be 32-bit or 64-bit. However, both libraries must use the same bit size.

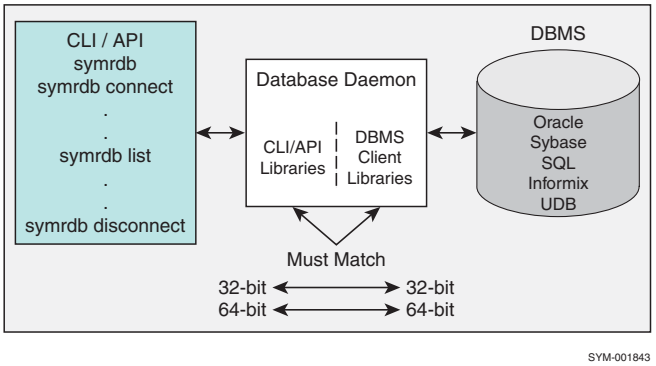

Figure 10 Database daemon libraries for communication

Note: The 64-bit database daemons (`storora64d` and `storsysbs12.5_64d`) require the optional 64-bit libraries (`/usr/symcli/shlib`) and DBMS client libraries at product installation time.

Table 8 lists the set of database daemons.

Table 8 **Database daemon names**

Daemon name	Database type
stororad	Oracle
storora64d	Oracle 64-bit
storudbd	DB2/UDB
storifmxd	Informix
storifmx64d	Informix (64-bit)
storsqld	SQL
storsybs12d	Sybase Version 12
storsybs12.5d	Sybase Version 12.5
storsybs12.5_64d	Sybase Version 12.5 (64-bit)

Refer to the *EMC Solutions Enabler Installation Guide* for more information about installing and using daemons.

Listing database instances

The following command lists the available databases (database instances):

```
symrdb list -type Oracle
```

For Oracle databases, it lists the current Oracle database instance. The following is sample output from this command:

```
DATABASE NAMES (ORACLE 8.0.6.0.0):

  Database Name
  -----------

  64806
```

Sample output from this database is used throughout this chapter.

Examining database files

The following command identifies all the files of a specified database:

```
symrdb list -type Oracle FILE -db DbName
```

The command displays a list of database files and their attributes as follows:

- Database version
- Name of the files
- Type of file: control, data, or log
- Database file status: online, offline

For more information about the `symrdb list` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Examining database file attributes

The following command lists the attributes and the extent information of a specific file in a specified database:

```
symrdb show -type Oracle FILE FileName -db DbName
```

For example, to examine a specific file on an Oracle database named `64806`, enter:

```
symrdb show -type oracle FILE /usr/oracle/oradata/64806/rbs01.dbf -db 64806
```

The following is sample output from this command:

```
DATABASE            : 64806
DATABASE FILE NAMES (ORACLE 8.0.6.0.0):

Database File Name : /usr/oracle/oradata/64806/rbs01.dbf

  Database File Type                   : Data
  Database File Status                 : Online
  Database File Size                   : 10m

  Database File Data Offset            : 2k
  Database File Block Size             : 2k

  Database Allocated Block Size        : 10m

  Database Free Block Size             : 9m

  Database File Unallocated Block Size : N/A


Absolute Path                 : /usr/oracle/oradata/64806/rbs01.dbf
Resolved Object Type          : SunOS UFS File
Resolved Object Size          : 10m
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 1
File System Mount Point       : /
File System Device Name       : /dev/dsk/c0t1d0s0

Number of Mirrors for object (1):
{
1) Mirror Configuration
```

```
Mirror Physical Extents (69):
{
----------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                   Offset
----------------------------------------------------------------------
64k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          11654m
32k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          11654m
1m     N/A  N/A      N/A /dev/rdsk/c0t1d0s0          12409m
120k   N/A  N/A      N/A /dev/rdsk/c0t1d0s0          12410m
...
Mirror Physical Devices (1):
{
----------------------------------------------------------------------

Array Dev   PPdevName               PdevName
----------------------------------------------------------------------
N/A   N/A   /dev/rdsk/c0t1d0s0      /dev/rdsk/c0t1d0s2
...
}
```

The command shows the attributes and extent information about the file as follows:

- Database version
- Type of file: control, data, or log
- Database file status: online, offline
- File size
- Offset (blocks) to data from the beginning of the file
- Size of the data blocks in the database
- Allocated blocks for the database
- Free blocks
- Extent information

The `-expand` option expands the extent information displayed in the list.

The `-no_extents` option shows information about the object without the details of extent information.

For more information about the `symrdb show` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Listing tablespace files

The following command lists all the files of a specified tablespace and database:

```
symrdb list -type Oracle FILE -tbs TblSpName -db DbName
```

The command displays a list of files and their attributes as follows:

- Database version
- Name of the files
- Type of file: control, data, or log
- Database file status: online, offline

Translating database devices to Symmetrix groups

The devices of an existing database can be translated and defined as a Symmetrix device group or composite group.

DB devices to device groups

To create an RDF1 device group named `Prod1dg` with only the standard devices from the database named `Prod1db`, enter:

```
symrdb -db Prod1db rdb2dg Prod1dg -nobcv -dgtype RDF1
```

To create a REGULAR device group named `Prod2dg` with only the R2 BCV devices from the database named `Acc2db`, enter:

```
symrdb -db Acc2db rdb2dg Prod2dg -r2 -bcv -dgtype REGULAR
```

DB devices to composite groups

To create a composite group named `ProdRcg` with the R1 standard and R1 BCV devices from the database named `ADB4db`, enter:

```
symrdb -db ADB4db rdb2cg ProdRcg -cgtype RDF1
```

To create a composite group named `myrdfcg` with only the R1 standard devices from the database named `Accdb`, enter:

```
symrdb -db Accdb rdb2cg myrdfcg -nobcv -cgtype RDF1
```

Virtual devices can be added to a device group or composite group by using the `-vdev` option.

Examining tablespaces

The following command lists all the tablespaces of a specified database:

```
symrdb list -type Oracle TBS -db DbName
```

For example, to list the tablespaces in the Oracle database `64806`, enter:

```
symrdb list -type oracle TBS -db 64806
```

The following is sample output from this command:

```
DATABASE          : 64806
TABLE SPACE NAMES (ORACLE 8.0.6.0.0):

  Table Space  Name              Type       Status     Size (mb)  Free (mb)
  -------------------            ------     --------   ---------- ----------

  RBS                            Permanent  Online     30         26
  SYSTEM                         Permanent  Online     64         46
```

The command displays a list of tablespaces and their attributes as follows:

- Database version
- Names of the tablespaces
- Type of tablespace: temporary, permanent, table, index, partitioned table, or partitioned index
- Tablespace states: online, offline, readonly, or restricted
- Tablespace size in MB
- Free space available in MB

For more information about the `symrdb list` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference Guide*.

Examining tablespace attributes

The following command lists the attributes of a tablespace of a specified database:

```
symrdb show -type Oracle TBS TblSpName -db DbName
```

For example, to see the attributes of the tablespace `SYSTEM`, enter:

```
symrdb show -type oracle TBS SYSTEM -db 64806
```

The following is sample output from this command:

```
DATABASE          : 64806
TABLE SPACE NAMES (ORACLE 8.0.6.0.0):


Database Table Space Name : SYSTEM

Database Table Space Type       : Permanent
Database Table Space Status     : Online

Number of Host Files (1):
    {
1) Database File Name: /usr/oracle/oradata/64806/system01.dbf
```

```
Absolute Path                  : /usr/oracle/oradata/64806/system01.dbf
Resolved Object Type           : SunOS UFS File
Resolved Object Size           : 64m
Number of Trailing Bytes       : 0
Extent byte offset to data     : 0
Number of Physical Devices     : 1
File System Mount Point        : /
File System Device Name        : /dev/dsk/c0t1d0s0

Number of Mirrors for object (1):
{
1) Mirror Configuration

Mirror Physical Extents (417):
{
----------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                   Offset
----------------------------------------------------------------------
64k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          11654m
32k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          11654m
80k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          12004m
1m    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          12004m
...
}
Mirror Physical Devices (1):
{
----------------------------------------------------------------------

Array Dev   PPdevName               PdevName
----------------------------------------------------------------------
N/A   N/A   /dev/rdsk/c0t1d0s0      /dev/rdsk/c0t1d0s2
...
}
```

The command shows the attributes and extent information about the tablespace.

The `-expand` option expands the extent information displayed in the list.

The `-no_extents` option shows information about the object without the details of extent information.

For more information about the `symrdb show` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Listing tablespace files

The following command lists all the files of a specified tablespace and database:

```
symrdb list -type Oracle FILE -tbs TblSpName -db DbName
```

If the connection environment variables have been previously set (for any database type) described in Table 8 on page 55, then only enter:

```
symrdb list FILE -tbs TblSpName -db DbName
```

The command displays a list of files and their attributes as follows:

- Database version
- Name of the files
- Type of file: control, data, or log
- Database file status: online, offline

Listing tablespace tables

The following command lists the tables of a specified tablespace:

```
symrdb list -type Oracle TABLE -tbs TblSpName -db DbName
```

For example to list the tables for tablespace SYSTEM, enter:

```
symrdb -type oracle list TABLE -tbs SYSTEM -db 64806
```

The following is sample output from this command:

```
TABLE SPACE NAME     : SYSTEM
DATABASE TABLE NAMES (ORACLE 8.0.6.0.0):

  Table Name                    Owner                          Type
  -------------                 ------                         --------
ACCESS$                       SYS                            Simple Table
AQ$_MESSAGE_TYPES             SYS                            Simple Table
AQ$_QUEUE_STATISTICS          SYS                            Simple Table
```

The command displays a list of table names, and for each table provides the owner and the table type.

Listing segments in a tablespace

The segments of a specified tablespace can be listed. A segment (storage unit) exists for every data table, partition, and every index.

To list the segments of a tablespace, enter:

```
symrdb list -type Oracle SEG -tbs TblSpName -db DbName
```

For example, to list the segments of tablespace SYSTEM, enter:

```
symrdb -type oracle -db 64806 list SEG -tbs SYSTEM -db 64806
```

The following is sample output from this command:

```
DATABASE               : 64806
TABLE SPACE NAME       : SYSTEM
DATABASE SEGMENT NAMES (ORACLE 8.0.6.0.0):

Segment Name                  Owner                          Type
--------------                ------                         --------

1.527                        SYS                        Cache
C_COBJ#                      SYS                        Cluster
C_FILE#_BLOCK#               SYS                        Cluster
SYS_C00512                   SYS                        Index
SYS_IL0000000227C00002$      SYS                        LOB_Index
SYS_LOB0000000235C00002$     SYS                        LOB_Segment
SYS_LOB0000000239C00002$     SYS                        LOB_Segment
SYSTEM                       SYS                        Rollback
ACCESS$                      SYS                        Table
```

The command displays a list of segments, and for each segment provides the owner and segment type.

Translating tablespace devices to Symmetrix groups

The devices of an existing tablespace can be translated and defined as a Symmetrix device group or composite group.

TBS devices to device groups

To create an RDF1 device group named `Prod1dg` with only the standard devices from the tablespace named `Prod1tbs`, enter:

```
symrdb -tbs Prod1tbs tbs2dg Prod1dg -nobcv -dgtype RDF1
```

To create a REGULAR device group named `Prod2dg` with only the R2 BCV devices from the tablespace named `Acc2tbs`, enter:

```
symrdb -tbs Acc2tbs tbs2dg Prod2dg -r2 -bcv -dgtype REGULAR
```

TBS devices to composite groups

To create a composite group named `ProdRcg` with the R1 standard and R1 BCV devices from the tablespace named `ADB4tbs`, enter:

```
symrdb -tbs ADB4tbs tbs2cg ProdRcg -cgtype RDF1
```

To create a composite group named `myrdfcg` with only the R1 standard devices from the tablespace named `Acctbs`, enter:

```
symrdb -tbs Acctbs tbs2cg myrdfcg -nobcv -cgtype RDF1
```

The `-vdev` option is used to add virtual devices to a device group or composite group.

Examining schemas

The following command lists all the schemas of a specified database:

```
symrdb list -type Oracle SCHEMA -db DbName
```

For example, to list the schemas for Oracle database 64806, enter:

```
symrdb list -type oracle SCHEMA -db 64806
```

The following is sample output from this command:

```
DATABASE     : 64806
SCHEMA NAMES (ORACLE 8.0.6.0.0):

  Schema Name
  -----------

  SYS
  SYSTEM
  DBSNMP
```

The command displays a list of the schema names for the database. For more information about the `symrdb list` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference Guide*.

Listing schema files

The following command lists all the files of a specified schema:

```
symrdb list -type Oracle FILE -schema SchemaName -db DbName
```

The command displays a list of files and their attributes as follows:

- Database version
- Name of the files
- Type of file: control, data, or log
- Database file state: online, offline

Listing schema tables

The following command lists the tables of a specified schema:

```
symrdb list -type Oracle TABLE -schema SchemaName -db DbName
```

The command displays a list of tables and their attributes as follows:

- Database version
- Name of the tables
- Table owner
- Table type: partitioned, nonpartitioned, clustered, index-organized, nested, or N/A

Examining schema attributes

The following command lists the attributes of a schema of a specified database:

```
symrdb show -type Oracle SCHEMA SchemaName -db DbName
```

For example, to display the attributes of schema `SYSTEM`, enter:

```
symrdb show -type oracle SCHEMA SYSTEM -db 64806
```

The following is sample output from this command:

```
Database Schema Name : SYSTEM


Number of Host Files (1):
{
1) Database File Name: /usr/oracle/oradata/64806/system01.dbf

Absolute Path                 : /usr/oracle/oradata/64806/system01.dbf
Resolved Object Type          : SunOS UFS File
Resolved Object Size          : 64m
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 1
File System Mount Point       : /
File System Device Name       : /dev/dsk/c0t1d0s0

Number of Mirrors for object (1):
{
1) Mirror Configuration

Mirror Physical Extents (417):
{
-----------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                   Offset
-----------------------------------------------------------------------
64k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          11654m
32k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          11654m
80k    N/A  N/A      N/A /dev/rdsk/c0t1d0s0          12004m
...
}
Mirror Physical Devices (1):
{
-----------------------------------------------------------------------

Array Dev   PPdevName               PdevName
-----------------------------------------------------------------------
N/A   N/A   /dev/rdsk/c0t1d0s0      /dev/rdsk/c0t1d0s2
...
}
```

The output displays the attributes and extent information about the schema.

The `-expand` option expands the extent information displayed in the list.

The `-no_extents` option shows information about the object without the details of extent information.

For more information about the `symrdb show` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Listing segments in a schema

The segments of a specified schema can be listed. A segment (storage unit) exists for every data table, partition, and every index in a tablespace.

To list the segments in a schema, enter:

```
symrdb list -type Oracle SEG -schema SchemaName -db DbName
```

For example, to list the segments of schema `SYSTEM`, enter:

```
symrdb -type oracle -db 64806 list SEG -schema SYSTEM
```

The following is sample output from this command:

```
DATABASE                : 64806
SCHEMA NAME             : SYSTEM
DATABASE SEGMENT NAMES (ORACLE 8.0.6.0.0):

Segment Name                  Owner                  Type
--------------                ------                --------

AQ$_QUEUES_CHECK              SYSTEM                  Index
AQ$_QUEUES_PRIMARY            SYSTEM                  Index
AQ$_QUEUE_TABLES_PRIMARY      SYSTEM                  Index
AQ$_SCHEDULES_CHECK           SYSTEM                  Index
AQ$_SCHEDULES_PRIMARY         SYSTEM                  Index
```

The command displays a list of segments, and for each one provides the owner and type.

Examining tables

The following command lists the tables of a specified tablespace or schema:

```
symrdb list -type Oracle TABLE <-tbs TblSpName>|<-schema SchemaName> -db DbName
```

The command displays a list of tables and their attributes as follows:

- Database version
- Names of the tables
- Type of table: partitioned, nonpartitioned, clustered, index-organized, nested, or N/A
- Table owner
- Index-organized tablename
- Nested parent table information
- Cluster name information

For more information about the `symrdb list` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Examining table attributes

The following command lists the attributes of a table of a specified tablespace or schema:

```
symrdb show -type Oracle TABLE TableName <-tbs TblSpName>|<-schema SchemaName> -db DbName
```

The command shows the attributes and extent information about the table as follows:

- Database version
- Name of table
- Type of table: partitioned, nonpartitioned, clustered, index-organized, nested, or N/A
- Table owner
- Index-organized tablename
- Nested parent table information
- Cluster name information

The `-expand` option expands the extent information displayed in the list.

The `-no_extents` option shows information about the object without the details of extent information.

For more information about the `symrdb show` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Examining segments

The following command lists all the segments of a specified tablespace or schema:

```
symrdb list -type Oracle SEG <-tbs TblSpName>|<-schema SchemaName>
```

For example, to display the segments for tablespace SYSTEM, enter:

```
symrdb -type oracle -db 64806 list SEG -tbs SYSTEM
```

The following is sample output from this command:

```
DATABASE                : 64806
TABLE SPACE NAME        : SYSTEM
DATABASE SEGMENT NAMES (ORACLE 8.0.6.0.0):

  Segment Name                Owner                          Type
  --------------              ------                         --------

1.527                          SYS                          Cache
C_COBJ#                        SYS                          Cluster
SYS_C00512                     SYS                          Index
SYS_IL0000000227C00002$        SYS                          LOB_Index
SYS_LOB0000000235C00002$       SYS                          LOB_Segment
SYS_LOB0000000239C00002$       SYS                          LOB_Segment
SYSTEM                         SYS                          Rollback
ACCESS$                        SYS                          Table
AQ$_MESSAGE_TYPES              SYS                          Table
```

The command displays a list of segments, their owner, and the type. The segment types can include the following:

- Cache
- Cluster
- Deferred rollback
- Index
- Index partition
- LOB index
- LOB segment
- Rollback
- Table
- Table partition
- Temporary
- Index-organized table
- Nested table
- N/A

For more information about the `symrdb list` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Examining segment attributes

The following command lists the attributes of a segment of a specified tablespace or schema:

```
symrdb show -type Oracle SEG SegmentName <-tbs TblSpName | -schema SchemaName> -db
   DbName
```

The command shows the attributes about the segment as follows:

- Database version
- Name of the segment
- Type of segment
- Segment owner
- Index-organized table
- Nested parent table information
- Number of cluster tables in cluster
- List of cluster table members

For more information about the `symrdb show` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Invoking database I/O control

The `symioctl` command, invokes I/O control operations to a specified relational database or database object(s).

Note: Set SYMCLI_RDB_CONNECT to your username and password for access to the specified database.

Freezing the database

The following command freezes all I/O access to a specified relational database:

```
symioctl freeze -type DbType Object Object
```

IBM DB2/UDB and SQL Server allow some or all databases to be specified. Oracle and Informix allow a user to freeze or thaw an entire DB system.

If the connection environment variables have been previously set, as described in Table 8 on page 55, then only enter:

```
symioctl freeze Object Object
```

For example, to freeze databases HR and Payroll, enter:

```
symioctl freeze HR Payroll
```

The freeze action can be used in conjunction with the TimeFinder® or SRDF® split operation. The freeze suspends the database updates being written to disk.

Note: SQL Server users should avoid freezing the "master" database as this can lead to unpredictable behavior.

Thawing the database

Once the freeze action is completed, the split may proceed. When the split operation completes, a `symioctl` thaw command must be sent to resume full I/O access to the database instance. For example:

```
symioctl thaw
```

For more information about the `symioctl` syntax, refer to the *EMC Solutions Enabler SYMCLI Command Reference*.

Note: Oracle freeze/thaw cannot be performed using a SQL Net connection. The connection must be issued locally or through the SYMAPI server.

Hot backup control

For Oracle only, hot backup control can only be performed on a list of tablespace objects, which must be performed before and after a freeze/thaw command. The steps required to split a group of BCV devices are as follows:

1. Issue the `symioctl begin backup` command.

2. Issue the `symioctl freeze` command.
3. Split standard and BCV pairs. This may involve several steps depending on the environment.
4. Issue the `symioctl thaw` command.
5. Issue the `symioctl end backup` command.

Checkpoint

The following command requests the relational database to perform a checkpoint:

```
symioctl checkpoint
```

Oracle and Informix checkpoint the entire database system. SQL Server allows some or all databases to be specified. DB2/UDB does not support checkpoint.

Archive log

For Oracle only, the current log can be archived. This forces the log writer process to stop writing to the current log file and switches to the next available online log file.

SQL Server snapshot

A snapshot of an SQL Server database is a copy of the data and log files at a single point in time. Microsoft added support for snapshot backup/restore in SQL Server 2000. The `symioctl` command now provides an interface to the SQL Server snapshot feature. When combined with TimeFinder instant split, database copies can be created quickly for backup, reporting, testing, and initialization of warm standby servers.

The snapshot arguments for the `symioctl` command extend the freeze/thaw functionality to allow both physical and logical restore of an SQL Server database. The snapshot backup will save metadata about the database in a file, which will be needed for a subsequent snapshot restore.

The TimeFinder split should be preceded by the following command:

```
symioctl begin snapshot Object SAVEFILE savefile
```

The *`Object`* is a single database name and the *`savefile`* is the name of the file used to save the database metadata. By default, the *`savefile`* is not overwritten. To replace it, either remove it or specify the `-overwrite` option.

The TimeFinder split should be followed by this command:

```
symioctl end snapshot Object
```

The *`Object`* is the same database name used to begin. Note that all writes to the database are suspended during the begin/end snapshot sequence so TimeFinder instant split should be used to minimize transaction delays.

If a problem occurs and the backup needs to be terminated, enter the following command:

```
symioctl abort snapshot Object
```

The *`Object`* is the same database name used to begin.

The TimeFinder restore should be followed by this command:

```
symioctl restore snapshot Object SAVEFILE savefile
```

The *Object* is the name of the database to restore and the *savefile* is the name of the file previously used to save the database metadata. By default, SQL Server performs a logical database recovery. To apply additional transaction logs, specify the `-norecovery` option. To use the restored database as a standby database, specify the `-standby` option.

Using the EMC Oracle ASM library

Oracle10g includes a storage management and provisioning feature called automatic storage management (ASM). ASM provides file system and volume manager capabilities, which are built into the Oracle database kernel. These capabilities work with the EMC Solutions Enabler libraries.

The following describes the disk string syntax for the Oracle disk discovery process:

```
SetDesc:= EMC:Expression
Expression:= Term | Expression 'OR' Term
Term  := Factor | Term 'AND' Factor
Factor:= '(' Expression ')' |
           'NOT' Factor |
           ScalarAttribute Relop Integer-Constant |
           ScalarAttribute Relop
             Integer-Constant Multiplier |
           StringAttribute '==' Character-String
ScalarAttribute:= 'SIZE' |'size'|'BLKSZ'|'blksz'|
StringrAttribute:= 'PATH'|'path'|'FGROUP'|'fgroup'|
Relop           := '=' | '!='| '<' |'>'|'<='|'>='
Multiplier:= 'K' | 'k' | 'M' | 'm' | 'G' | 'g'
```

where:

The value of the `'PATH' | 'path'` attribute is a valid pathname expression for a device's special file node, or a symbolic link name referring to the device.

The value of the `'FGROUP' | 'fgroup'` is a valid failure group name returned from the ASM library during discover.

The `Multiplier` terminal symbols specify Kilo, Mega, and Giga multipliers of integer-constant values respectively.

Notes:

1. `'PATH'` is assumed if no `StringrAttribute` is specified.
2. The input devices must be raw character devices. On Linux, it can be done by using the command 'raw' to bind the raw devices to a block device.
3. '`EMC`:' must be specified in `"SetDesc := EMC:Expression"`.
4. All the devices must be visible to the Oracle user.

Following are some examples:

`ASM_DISKSTRING='EMC:PATH = /dev/rdsk/* AND SIZE > 1G'`

This constructs a disk set of all configured devices known to ASM matching the pattern `/dev/rdsk/*` and of capacity greater than one gigabyte.

```
create diskgroup dg1 external redundancy disk 'EMC:/dev/rdsk/c3t0d1s6'
```

This creates a diskgroup `dg1` with external redundancy using device `/dev/rdsk/c3t0d1s6`.

Usage of CLI asmdscvr

The `asmdscvr` command is a utility implementing the Oracle ASM discovery function. It allows administrators to find the disks that are already in disk groups, and disks that are available for adding to disk groups. Discovery makes the characteristics of the disk available to ASM. Disks discovered through `asmdscvr` do not need to be available through normal operating system interfaces.

The `asmdscvr` utility is included in the Solutions Enabler installation package, by default, in directories under the directory `/usr/symcli/bin`.

The `asmdscvr` utility provides the following:

```
asmdscvr -d <SetDesc> | [-h] | [-g] [-m owner,group,mode]
```

where:

`-d <SetDesc>`— Executes an asm_discover() call with the disk set descriptor `<SetDesc>`

`-m` — Sets owner/group/mode of file to owner, group, mode

`-h` — Prints this message

`-g` (like `-h`) — Explains the BNF grammar for `<setdesc>`

```
<setdesc> GRAMMAR BNF:
```

A disk set descriptor `<setdesc>`, is a string conforming to this abstract syntax notation grammar:

Notes:

1. 'EMC:' is not required in `"SetDesc := Expression"`.
2. All the devices must be visible to the user.

Examples

To fetch all configured devices known to ASM matching the pattern `/dev/rdsk/*`, enter:

```
asmdscvr -d "PATH=/dev/rdsk/*"
```

To fetch all configured devices known to ASM of capacity greater than two gigabytes, enter:

```
asmdscvr -d "SIZE >= 2G"
```

To fetch all configured devices known to ASM matching the pattern /dev/rdsk/* and of capacity greater than two gigabytes, enter:

```
asmdscvr -d "PATH = /dev/rdsk/* AND SIZE > 2G"
```

CHAPTER 4
File System SRM

This chapter describes host file system commands and how the file systems and their structure can be examined using SYMCLI commands.

- Overview 74
- Finding file systems 74
- Examining a file system 75
- Finding directories and files 76
- Examining files 78

Overview

The file system commands list attributes of file systems, directories, and files, and their mapping to physical devices and extents.

The file system commands support the following operating systems and file types:

- HP-UX [hfs, VxFS, NFS]
- IBM AIX (jfs), AIX 5.1 (jfs, jfs2), VxFS, NFS
- Linux (ext2, ext3, ext4, VxFS, OCFS, OCFS2, GFS, XFS, NFS)
- Sun Solaris (ufs, VxFS, NFS)
- Tru64 UNIX (ufs, AdvFS, NFS)
- VMware file system (VMFS)
- Windows 2000, 2003, 2008 (NTFS, FAT32, CIFS, NFS)

Note: In Solutions Enabler V7.1 and later, SRM commands support file system/logical volume mapping for the virtual disk on the virtual machines. VMware ESX Server 3.5 and later are supported. The *EMC Solutions Enabler Symmetrix Array Management CLI Product Guide* provides information for setting up VMware virtual disk mapping support.

Terminology

A *file system* is the overall structure in which files are named, stored, and organized by an operating system. Typically, there is more than one file system and file system type being managed by an operating system. A file system consists of files, directories, and file management structures on disk needed to locate and access these items.

An *extent* is a block of storage space on a disk, which is reserved by the file system for the storage of a particular file or program.

An *i-node* is an internal file system data structure (for UNIX-based operating systems) that describes an individual file. An i-node contains the file type, owner, size, modified date, and physical device location of the file. Typically, a table of i-nodes is stored near the beginning of a file system.

Finding file systems

The following command lists the file systems that are on the host system:

```
symhostfs list
```

The following is sample output:

```
Mounted File Systems

  Device Name                    Type         Mount Point
  ------------                   ------       -----------

  /dev/vg00/lvol3                HP VxFS        /

  /dev/vg00/lvol1                HFS            /stand
```

```
  /dev/vg00/lvol8                HP VxFS        /var

  /dev/vg00/lvol7                HP VxFS        /usr

  /dev/vg00/lvol4                HP VxFS        /tmp

  api193:(pid787)                Unknown Type   /net
```

The command lists the following attributes for each file system:

- Device name where the file system is located
- File system type
- File system mount point

To list mounted files systems with NFS mount points, use the `-nfs` option as shown in the following example:

`symhostfs list -nfs`

The following is sample output:

```
Mounted File Systems

  Device Name                                  Type         Mount Point
  ------------                                 ------       -----------
  api40:/usr3/ftpusers/outgoing                 NFS            /apikits
```

Examining a file system

The following command lists the attributes of a specified file system:

`symhostfs show` *`ObjectName`*

To specify the file system, use the file system mount point, as shown in the following example:

`symhostfs show /stand`

The following output shows a Linux file system with the mount point `/stand` in a clustered environment. It is mounted on the logical volume `lvol1` created from the underlying device `c0t3d0`.

```
File System Mount Point         : /stand

File System Type                : EXT3
Mount Options                   : Read Write | Clustered

File System Device Name         : /dev/vg00/lvol1

Size                            : 82m
Number Free Blocks              : 117778
Fragment Size                   : 1024 bytes
Block Size                      : 8192 bytes

Number Inodes                   : 13440
Number Free Inodes              : 13403

Mount Time                      : Mon 17-Mar-2003 16:18

Absolute Path                 : /stand
Resolved Object Type          : File System
Resolved Object Size          : 82m
```

```
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 1
File System Mount Point       : /stand
File System Device Name       : /dev/vg00/lvol1

Number of Mirrors for object (1):
{
1) Mirror Configuration

Mirror Physical Extents (1):
{
Number of Mirrors for object (1):
1) Mirror Configuration

Mirror Physical Extents (1):

----------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                   Offset
----------------------------------------------------------------------
82m    N/A  N/A      N/A /dev/rdsk/c0t3d0                3m

Mirror Physical Devices (1):
----------------------------------------------------------------------

Array Dev   PPdevName               PdevName
----------------------------------------------------------------------
N/A   N/A   /dev/rdsk/c0t3d0        /dev/rdsk/c0t3d0
}
```

Note: Clusterd file systems are only supported on Solaris and Red Hat (global file systems) clustered environments.

By using the following options, the returned extent information can be expanded or collapsed:

- The `-expand` option expands the extent information displayed in the list.
- The `-no_extents` option shows information about the object without the details of extent information.

Finding directories and files

The following command lists the subdirectories of any root or parent directory:

```
symhostfs list -dir dirname
```

The following shows a sample command and output:

```
symhostfs list -dir /stand
```

```
Directory Name : /stand

Directory Name                 Type           Perms     Owner  Group
--------------                 ------       ----------  -----  -----

lost+found                     HFS          drwxr-xr-x      0      0
build                          HFS          drwxr-xr-x      0      3
dlkm                           HFS          drwxr-xr-x      0      3
dlkm.vmunix.prev               HFS          drwxr-xr-x      0      3
```

The command lists all subdirectories found in the specified parent directory.

The information can be displayed recursively for subdirectories by using the `-R` option. The following sample shows a portion of the output:

```
...
Directory Name : /stand/lost+found

Directory Name : /stand/build

Directory Name                 Type           Perms     Owner  Group
--------------                 ------       ----------  -----  -----

mod_wk.d                       HFS          drwxr-xr-x      0      3


Directory Name : /stand/build/mod_wk.d

Directory Name : /stand/dlkm

Directory Name                 Type           Perms     Owner  Group
--------------                 ------       ----------  -----  -----

system.d                       HFS          drwxr-xr-x      0      3
node.d                         HFS          drwxr-xr-x      0      3
mod.d                          HFS          drwxr-xr-x      0      3
...
```

Examining directories

The following command lists the attributes of a directory:

```
symhostfs show dirname
```

The following is a sample command and output:

```
symhostfs show /stand/dlkm/system.d
```

```
Directory Name                  : /stand/dlkm/system.d
Directory Type                  : HFS
Directory Size                  : 0b

Directory Mode                  : 40755
Directory Permission            : drwxr-xr-x
Directory Owner ID              : 0
Directory Group ID              : 3

Number of Symbolic Links        : 2

Last Access Time                : Tue 18-Mar-2003 11:57
Last Modification Time          : Mon 02-Dec-2002 15:19
Last Status Change Time         : Mon 02-Dec-2002 15:19
```

The command lists the directory attributes, properties, and access times.

Listing files

The following command lists all the files of a directory:

```
symhostfs list -file dirname
```

The command lists all files found in the specified directory as shown in the following sample output:

```
symhostfs list -file /stand
```

```
Directory Name  : /stand
FileSystem Type : HFS

  File Name                                 Size    Perms      Owner  Group
  ---------                                 ----  ----------  -----  -----

  ioconfig                                    5k  -rw-r--r--      0      3
  bootconf                                    1b  -rw-r--r--      0      3
  system                                      2k  -r--r--r--      0      3
  vmunix                                     11m  -rwxr-xr-x      0      3
  kernrel                                     1b  -r--r--r--      0      3
  rootconf                                    1b  -rw-------      0      0
  system.prev                                 2k  -r--r--r--      0      3
  vmunix.prev                                11m  -rwxr-xr-x      0      3
```

The information can be displayed recursively for subdirectories by using the `-R` option. The following sample shows a portion of the output:

```
Directory Name  : /stand/build
FileSystem Type : HFS

  File Name                                 Size    Perms      Owner  Group
  ---------                                 ----  ----------  -----  -----
  conf.c                                     64k  -rw-r--r--      0      3
  config.mk                                   6k  -rwxr--r--      0      3
  tune.h                                      3k  -rw-r--r--      0      3
  space.h                                     1b  -rw-r--r--      0      3
  conf.o                                     56k  -rw-r--r--      0      3
  conf.SAM.c                                 64k  -rw-r--r--      0      3
  config.SAM.mk                               6k  -rwxr--r--      0      3
  conf.o.old                                 54k  -rw-r--r--      0      3
  conf.SAM.o                                 55k  -rw-rw-rw-      0      3
  function_names.c                            1b  -rw-r--r--      0      3
  function_names.o                            1k  -rw-r--r--      0      3
```

This output displays the filename, size, permissions, owner, and group for each file in the directory.

Examining files

The following command lists the attributes of a file:

```
symhostfs show filename
```

The command shows the file attributes, as shown in the following sample output:

```
symhostfs show /stand/system
...
  File Name                       : /stand/system
  File Type                       : HP-UX HFS File
  File Size                       : 2k
  Number of Trailing Bytes        : 49
  Extent byte offset to data      : 0

  File Mode                       : 100444
  File Permission                 : -r--r--r--
  File Owner ID                   : 0
  File Group ID                   : 3

  Number of Symbolic Links        : 1

  Last Access Time                : Mon 17-Mar-2003 16:20
```

```
  Last Modification Time          : Mon 02-Dec-2002 15:12
  Last Status Change Time         : Mon 02-Dec-2002 15:12

  Device Name                     : /dev/vg00/rlvol1
  Inode Number                    : 14
  File System Fragment Size       : 2b

  Number of Extents in File       : (1)
      {
      -----------------------------------------
      Extent    Offset in    Extent      Offset
      Type      Device       Size        in File
      -----------------------------------------
      Data           121k        2k           0b
      }
    Absolute Path                 : /stand/system
    Resolved Object Type          : HP-UX HFS File
    Resolved Object Size          : 2k
    Number of Trailing Bytes      : 49
    Extent byte offset to data    : 0
    Number of Physical Devices    : 1
    File System Mount Point       : /stand
    File System Device Name       : /dev/vg00/lvol1

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration

         Mirror Physical Extents (1):
         {
         ----------------------------------------------------------------------

               Size  Array Dev    Offset PPdevName                   Offset
         ----------------------------------------------------------------------
                 2k    N/A  N/A      N/A /dev/rdsk/c0t3d0                3m
         }

         Mirror Physical Devices (1):
         {
         ----------------------------------------------------------------------

         Array Dev   PPdevName                PdevName
         ----------------------------------------------------------------------
         N/A   N/A   /dev/rdsk/c0t3d0         /dev/rdsk/c0t3d0
}
}
```

Using the following options, the returned information can be restricted to just data extents or just metadata extents, and/or the extent information expanded or collapsed:

The `-meta`|`-data` options restrict the listing to just metadata objects or just data objects.

The `-expand` option expands the extent information displayed in the list.

The `-no_extents` option shows information about the object without the details of extent information.

Note: Some journaled file systems (AIX 5.1 JFS2 in particular) do not provide an API to access internal file extent information. Therefore, file mapping must use disk-based extent information. Unfortunately, these file systems only provide a single mechanism (unmount) to flush file metadata to i-nodes on disk. The sync mechanism only flushes metadata to the journal (log). Because of this, recent changes to the file system may not be visible until the file system is unmounted and remounted.

CHAPTER 5
Logical Volume SRM

This chapter describes the host system logical volume SRM and how logical volumes and their groups can be examined using SYMCLI commands.

- Overview 82
- Viewing volume groups 88
- Volume group control operations 92
- Translating volume groups to Symmetrix groups 94
- Viewing logical volumes 96
- Logical volume control operations 99
- Viewing extents 101

Overview

The logical volume commands allow for mapping logical volumes to a detailed view of storage. Logical volume architecture defined by a logical volume manager (LVM) is a means for organizing physical disk storage for optimal use by host applications.

These commands describe volume groups, logical volumes, and their mapping to physical devices and extents.

Note: Refer to the E-Lab Interoperability Navigator at http://elabnavigator.emc.com for a list of the supported logical volume managers for Solutions Enabler.

LVM terminology

A LVM establishes a volume group by designating a set of physical devices as its members. The physical devices form the pool of disk space that is then used to define logical volumes, or virtual devices, which are made available to applications and databases. A logical volume is seen by users and applications as another physical device.

The user controls how physical disk space is used to create a logical volume by specifying its characteristics—how many copies of the data to maintain, how many physical devices to use, how to store the data on each device, and how much space to allocate.

To create a logical volume, disk space from the volume group's pool can be allocated in units of logical extents. For each logical extent, there are one or more corresponding physical extents, depending on the number of mirrors, or copies of the data to be maintained.

A physical extent is a specified number of disk blocks, and is the smallest unit of disk space that can be assigned to a logical volume. A physical extent's size is defined when a volume group is created and is applied to all physical devices in the group. The physical extents associated with a logical volume can be located on one or more physical devices, and may or may not be contiguous.

Note: Exceptions are the logical volume managers (VxVM, ptx/SVM, LSM, DiskADM, and LDM) that allow the size of physical extents, or subdisks to vary.

A *mirror* is a copy of a logical volume's data.

Figure 11 on page 83 illustrates a volume group that contains a logical volume (VOL1) whose logical extents are mapped to the physical devices PDEV1 and PDEV2.

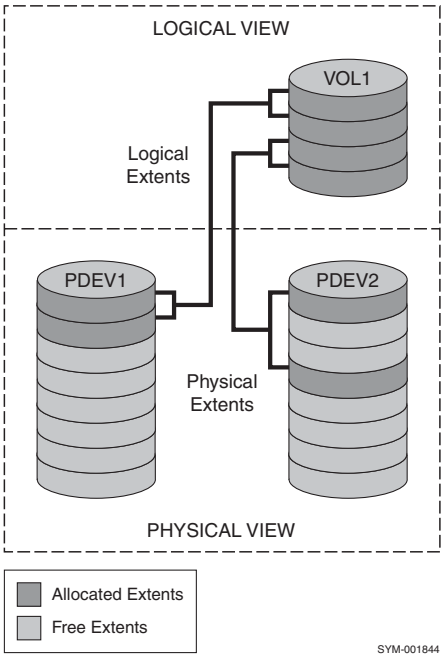

Figure 11 Volume group

Microsoft Windows

Volume mount points are NTFS directories that can be associated with specific logical volumes in a persistent manner. When the associated volume is formatted with a file system (NTFS, FAT, CDFS, etc.), then the volume mount point is similar to a UNIX file system mount point. When the associated volume is "raw" (not formatted), then the volume mount point is similar to a UNIX logical volume path. When a single Windows 2000 volume is associated with a drive letter and multiple volume mount points, then the volume mount point is similar to a UNIX symbolic link.

Windows 2000 assigns a *unique volume name* (UVN) to each volume when it is created. The UVN is a string of the following form:

```
\\?\Volume{GUID}\
```

where *GUID* is a globally unique identifier, such as:

```
e63553c7-e894-11d4-821f-806d3472696f
```

The `mountvol` command displays the UVN and assigned drive letters (if any) and volume mount points (if any) for each volume. Note that the Windows 2000 Configuration Manager allows users to create a volume without assigning any drive letters or volume mount points. If the Windows 2000 volume is assigned a drive letter, then the logical volume name is in the `V:` form. If the volume is not assigned a drive letter, then the logical volume name is in the form `Volume6` (the logical volume name generated by LDM).

Mirror configurations

A mirror configuration indicates how the data in the logical volume is stored on physical devices. The logical volume configuration is derived from the configurations of each of its mirrors.

The following are the basic mirror configurations:

- Simple
- Concatenated
- Striped
- RAID 5 (For VxVM, LDM, and DiskADM only)

Simple configuration

A simple mirror configuration implies that all physical extents are on the same physical device. In Figure 12, all physical extents are on one physical device, PDEV1.

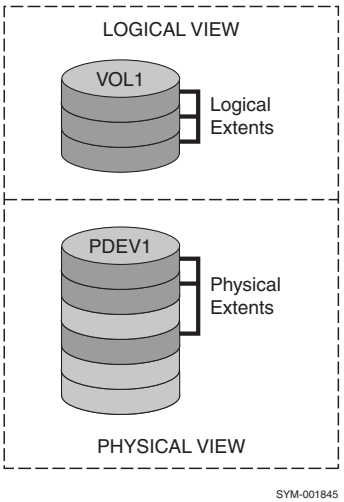

Figure 12 Simple mirror configuration

Concatenated configuration

A concatenated mirror configuration is comprised of physical extents that are on two or more physical devices.

In Figure 13, the logical extents on VOL1 point to the physical extents on PDEV1 and PDEV2.

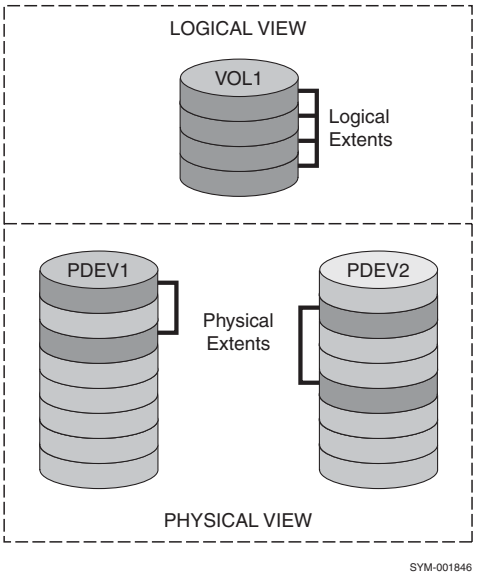

Figure 13 Concatenated mirror configuration

Striped configuration

In a striped mirror configuration, the data is written across multiple devices, with multiple stripes of data being written to each extent.

In Figure 14, VOL1 is a striped logical volume whose logical extent data is interleaved across multiple physical extents on PDEV1, PDEV2, PDEV3, and PDEV4.

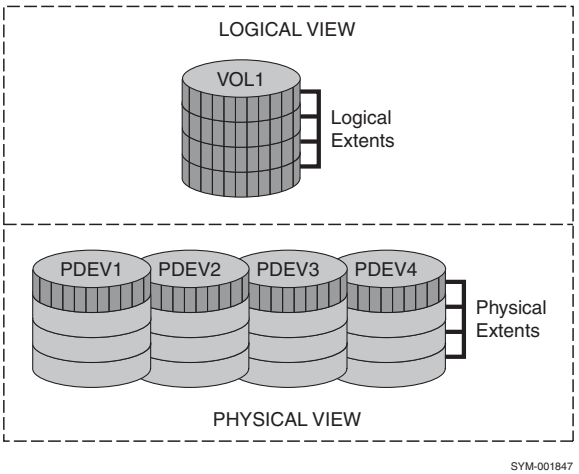

Figure 14 Striped mirror configuration

RAID 5 configuration

In a RAID 5 mirror configuration (for VxVM, Windows NT DiskADM, and LDM), the data is written across multiple devices, with multiple stripes of data and one stripe of parity data being written in sets across all devices. The stripe containing the parity data is shifted

across each device in the group. By spreading out the parity evenly across all devices, I/O performance is increased. In the case of a disk failure, a stripe's data can be reconstructed using the parity algorithm against the data in the available stripes.

In Figure 15, VOL1 is a logical volume with three logical extents. RAID 5 contains four columns, so the data from a logical extent will be written in stripes across all four columns, with one stripe containing parity information. In the figure, the stripes are spread out over four physical devices: PDEV1, PDEV2, PDEV3, and PDEV4.

For Windows NT, DiskADM (`WINDISK.EXE`), the parity shifts across all four columns, starting from the left-most column.

For Veritas VxVM and Windows 2000 LDM, the parity stripe shifts across all four columns, starting at the right-most column. As it shifts, the order in which the data stripes are written to the other columns also changes. The data stripes are written starting from the right of the parity column to the last column, and then resumes at the first column.

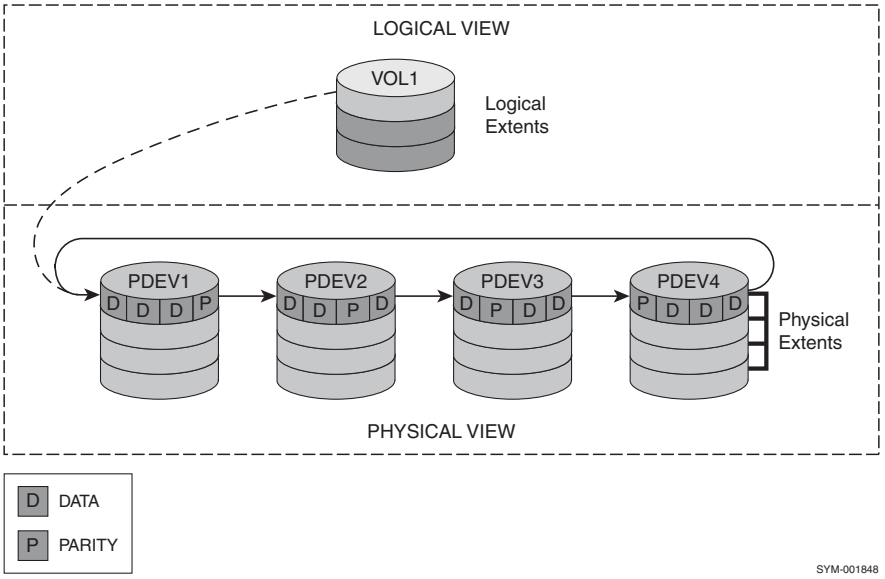

Figure 15 RAID 5 Veritas and Windows 2000 LDM configuration

Logical volume configurations

The logical volume configuration is determined by analyzing the configurations of its mirrors. The logical volume configurations can be viewed using the `symlv show` command. The possible logical volume configuration types are listed in Table 9.

Table 9 Logical volume configuration types

Type	Description
Concatenated	Only one copy of a logical volume's data is stored on disk, and that data is stored in multiple physical extents on two or more physical devices.
Mirror Concatenated	Multiple copies of a logical volume's data are maintained on disk. The data within each copy is stored in multiple physical extents on two or more physical devices.
Mirror Mixed	This type applies to Veritas VxVM only. Multiple copies of a logical volume's data are maintained in separate mirrors. The mirrors store their data in different configurations, either simple, concatenated, or striped.
Mirror Simple	Multiple copies of a logical volume's data are maintained in separate mirrors. Each mirror's data is stored in a series of extents on one physical device.
Mirror Striped	This type applies to VxVM, LSM, and ptx/SVM only. Multiple copies of a logical volume's data are maintained in separate mirrors. Each mirror's set of data is stored in a set of stripes written to multiple physical devices.
RAID5	This type applies to Veritas VxVM, DiskADM, and LDM volumes only. Only one copy of a logical volume's data is stored on disk, written in stripes across multiple physical devices. Included in each set of stripes is a stripe containing parity data that will allow any one stripe's data to be regenerated if that data is corrupted.
Simple	Only one copy of a logical volume's data is stored on disk, and all of its data is stored on one physical device in a series of extents.
Striped	Only one copy of a logical volume's data is stored on disk, with that data being written in stripes across multiple physical devices.

Given the mirror configuration type and the number of mirrors, the logical volume configuration type is determined as shown in Table 10.

Table 10 Logical volume mirror configurations (page 1 of 2)

Logical volume configuration type	Mirror configuration	Number of mirrors
Mirror Simple	Simple	2 or more
Mirror Concatenated	Concatenated	2 or more
Mirror Striped	Striped	2 or more
Mirror Mixed	Varied	2 or more
Simple	Simple	1

Table 10 Logical volume mirror configurations (page 2 of 2)

Logical volume configuration type	Mirror configuration	Number of mirrors
Concatenated	Concatenated	1
Striped	Striped	1
RAID5	RAID5	1 with parity data

Mirror conditions

Table 11 describes the types of mirror conditions.

Table 11 Mirror condition descriptions

Condition type	Description
Stale	The mirror contains some physical extents that are not consistent with corresponding extents in another mirror
Sync	All mirrors of the logical volume are synchronized
Empty	There is no data on the mirror
IOFail	There is an I/O failure on one of the devices
No Device	One or more mirror devices are missing
Offline	The mirror is offline
Initializing	The mirror is initializing

Viewing volume groups

With `symvg`, volume groups and group details can be viewed, as currently defined by the logical volume manager on the host system.

Note: Applications performing operations with the EMC PowerPath® Volume Manager must set the variable SHLIB_PATH (for HP only) to `/usr/lib`, and the variable LD_LIBRARY_PATH (for Solaris only) and LIBPATH (on AIX only) to point to the path that contains the PowerPath Volume Manager libraries.

Listing the volume groups

To list all the volume group names on the host system, enter:

```
symvg list
```

The command returns a list of volume group names and their logical volume manager type, as shown in the following sample output:

```
VOLUME GROUPS (HP-UX LVM):

                                          PE    Max      Max      Num      Num
  Name            State    Attribute      Size  Devices Volumes Devices Volumes

  /dev/vg00       Enabled  N/A              4m       16     255       1       9
  /dev/BigVG      Enabled  N/A              4m       16     255       1       1
```

```
/dev/nmktestvg  Enabled  N/A              4m       16     255       2       0
/dev/testvg     Enabled  N/A              4m       16     255       2       0
```

On Windows NT Disk Administrator, the task of establishing a volume group does not exist. For this volume type, a volume group name of `rootnt` is returned. This corresponds to other LVM manager default groups of `rootvg` or `rootdg`. For the Linux GFS pool volume manager, all the pools on the system are grouped under `gfspool`.

Volume group details

To view the logical volume information for a specified volume group, use the following form:

```
symvg show VgName
```

For example, to view details about volume group `/dev/BigVG`, enter:

```
symvg show /dev/BigVG
```

The following is sample output from this command:

```
   Volume Group Name : /dev/BigVG
   Volume Group Type : Linux LVM

       Volume Group State             : Enabled

       Volume Group Attributes        : Clustered | Read only

       Group's Physical Extent Size   : 4096k

       Max Number of Devices in Group :  16
       Max Number of Volumes in Group : 255

       Number of Devices in Group     :   1
       Number of Volumes in Group     :   1

       Physical Device Members (1):
           {
-------------------------------------------------------
                                 Sym                Cap
PdevName                  SymID  Dev  Att. Sts      (MB)
-------------------------------------------------------
/dev/rdsk/c2t1d5          03003  0123 (M)  RW     21577
       }
```

Note: Solutions Enabler Symmetrix SRM V7.5 supports Veritas clusters on Linux platforms. The mapping information for Veritas clustered volumes and filesystems, or file indentifies that the volume `/filesystem/file` is a clustered one. This information is displayed in the Volume Group Attributes field of the object.

This command provides the following information:

- Volume group name
- Logical volume manager type
- Volume group state
- Volume group attributes, such as whether it is a member of a cluster and if it is writeable

- Physical extent size
- Maximum number of logical and physical devices allowed
- Actual number of logical and physical devices in the group
- Number and names of the physical devices

Note: Run `symcfg sync` before running `symvg show` to make sure that the device status reports correctly.

The following information about each physical device member of the volume group also displays:

- Physical device name
- Symmetrix ID
- Symmetrix device name
- Device attributes
- Device status
- Device capacity in megabytes

Volume group types

The supported current operating systems and their associated logical volume managers (volume group type) are listed in Table 12.

Table 12 Volume group types

Operating system	Volume group type	VgType
HP-UX	HP-UX LVM HP-UX VxVM EMC PowerPath Volume Manager	HP_LVM HP_VXVM EMC_PVM
IBM AIX	AIX VxVM AIX LVM EMC PowerPath Volume Manager	AIX_VXVM AIX_LVM EMC_PVM
SunOS (Solaris)	SunOS VXVM Solstice Disk Suite[a] EMC PowerPath Volume Manager Oracle ASM Volume Manager[b]	SUN_VXVM SUN_SOLSTICE EMC_PVM ORACLE_ASM
Tru64 UNIX	OSF1 LSM	OSF1_LSM
Windows 2000, 2003, 2008	LDM VXVM	NT_LDM WIN_LDM WIN_VXVM
Linux	VxVM LINUX LVM GFS Pool Oracle ASM Volume Manager[b]	LINUX_VXVM LINUX_LVM LINUX_POOL ORACLE_ASM
AS/400	LVM	AS400_LVM

a. On the Solaris platform with SUN_SOLSTICE volume manager, a volume group named solstice cannot be created.
b. Set the following noted environmental variables for Solutions Enabler operations with the Oracle ASM Volume Manager and add the `$SYMAPI_ASM_HOME/bin` to your user path to point to the path that contains the Oracle ASM Volume Manager binaries. Set the SYMCLI_ASM_CONNECT variable to your username and password for access to the specified database, set SYMAPI_ASM_HOME to the location of the Oracle ASM binaries, and set SYMAPI_ASM_SID to the ASM instance name.

Volume group states

The volume group state describes the state of the volume groups. Possible values are:

- Enabled — Access to the volume group and its logical volume members is allowed.
- Disabled — Access to the volume group is not allowed.

Volume group control operations

The `symvg` command provides the following volume group control operations:

`create` — Creates a new volume group.

`destroy` — Deletes a volume group.

`recover` — Recovers a failed volume group.

`adddev` — Adds devices to a volume group.

`rmdev` — Removes devices from a volume group.

In addition, a logical volume group's metadata can be deported from a system to storage, and imported later to another host, such as a backup server.

Note: Refer to the *EMC Solutions Enabler SYMCLI Command Reference* for descriptions of the command syntax and options, and the E-Lab Interoperability Navigator at `http://elabnavigator.EMC.com` for supported Solutions Enabler platforms.

Creating a volume group

The following command creates a new volume group:

```
symvg create VgName [-p PartitionSize] PdevName...
```

The *VgName* is the name of the new volume group, and the *PdevName* is the name of the device(s) that belong to the new volume group.

The `-p` *PartitionSize* option specifies the partition size in megabytes. This option is only available for AIX_LVM.

Adding and removing devices

The following command extends a volume group by adding specific devices:

```
symvg adddev VgName PdevName...
```

The *VgName* is the name of the volume group, and the *PdevName* is the name of the device(s) that are being added to the volume group.

The following command reduces a volume group by removing specific devices:

```
symvg rmdev VgName PdevName...
```

The *VgName* is the name of the volume group, and the *PdevName* is the name of the device(s) that are removed from the volume group.

Destroying a volume group

The following command deletes a volume group:

```
symvg destroy VgName
```

Recovering a volume group

The following command recovers a failed volume group:

```
symvg recover VgName
```

Importing and deporting operations

The `deport` operation deports a volume group from one host system so that it can be imported on another host system of the same type. After a volume group is deported, that group is not available on the system from which it was deported.

As an example, if a volume group spans several disks, and a copy of that volume group is needed on another system, perform the following actions:

1. Deport the volume group.
2. Split the disks contained in the volume group as BCVs.
3. Import the volume group encapsulating the BCVs into another host system. The import operation makes the volume group available to this new host system.

In addition to the `import` and `deport` operations, the volume groups can be rescanned. The `rescan` operation is currently supported only for the Logical Disk Manager (LDM) volume groups on the Windows 2000 platform and the HP-UX LVM volume manager.

Control options

Table 13 describes the options that can be used with the `import` and `deport` operations for volume groups.

Table 13 Volume group control options

Option	Action	Description
`-overwrite`	`deport`	Used with the `-mapfile` option. Overwrites an existing mapfile if set.
`-cluster`	`import`	When set, imports a Windows VxVM volume group as a cluster.
`-clear`	`import`	When set, imports a volume group and clears the host ID on the volume group. This flag is for Veritas volume managers only.

Refer to the *EMC Solutions Enabler SYMCLI Command Reference* for a list of supported LVMs for the `import` and `deport` operations.

Translating volume groups to Symmetrix groups

The devices of an existing logical volume group can be translated and defined as a Symmetrix device group or composite group.

Standard devices (non-BCV) and their device/composite group must have matching types. BCV devices and their device/composite group can have different types. For example, an RDF1 composite group can contain R1 standard devices and/or have associated R1 or R2 BCV devices.

The following command creates a new device or composite group from a volume group:

```
symvg -h -v -type VgType -force -sid SymmID
     -rdfg GrpNum -R1 | -R2 -bcv | -nobcv | -vdev

   vg2dg VgName DgName
      -dgtype REGULAR | RDF1 | RDF2 | ANY

   vg2cg VgName CgName
      -cgtype REGULAR | RDF1 | RDF2 | ANY
      -apidb | -rdf_consistency
```

A group type `ANY` allows both non-RDF and RDF STD (R1, R11, R2, R22, and R21) devices in a single composite group or device group.

Additional options can be used to filter the devices added to the group. The `-sid` and `-RDFG` options can limit the devices to a specific Symmetrix ID or SRDF RA group.

Note: Composite groups often contain devices from multiple Symmetrix systems and (for SRDF) multiple RA groups.

Use the `-force` option to do a partial add (i.e., add whichever devices can be added). For example, consider a volume group with two R1 devices. If one device is already in an existing group and an attempt is made to add this device to a new group, the command will fail because a device cannot be in two different (device or) composite groups.

Volume groups to device groups

The following are examples of how to use the `symvg vg2dg` command:

- To create an `RDF1` device group named `newdg` with only the standard devices from the volume group named `thisvg`, enter:

  ```
  symvg vg2dg thisvg newdg -nobcv -dgtype RDF1
  ```

- To create a `REGULAR` device group named `newdg` with only the R1-BCV devices from the volume group named `thisvg`, enter:

  ```
  symvg vg2dg thisvg -R1 -bcv newdg -dgtype REGULAR
  ```

- To create an `ANY` device group name `newdg` from the volume group named `thisvg`, enter:

  ```
  symvg vg2dg thisvg  newdg -dgtype ANY
  ```

Volume groups to composite groups

The following are examples of how to use the `symvg vg2cg` command:

- To create an `RDF1` composite group named `newdg` with the R1 and R1-BCV devices from the volume group named `thisvg`, enter:

  ```
  symvg vg2cg thisvg newcg -cgtype RDF1 -R1 -bcv
  ```

- To create an `RDF1` composite group named `newcg` with only the R1 standard devices from the volume group named `thisvg`, enter:

  ```
  symvg vg2cg thisvg newcg -R1 -nobcv -cgtype RDF1
  ```

- To create an `ANY` composite group name `newdg` from the volume group named `thisvg`, enter:

  ```
  symvg vg2cg thisvg  newcg -cgtype ANY
  ```

Viewing logical volumes

Logical volumes are created by the logical volume manager as members of a volume group. A logical volume is seen by applications as another physical device. The data within a logical volume is stored on disk within one or more mirrors, and within each mirror as sets of allocated disk blocks called extents.

Listing logical volumes

To list all the logical volumes of a specified volume group, enter:

```
symlv -g VgName list
```

For example, to list the logical volumes for volume group `/dev/BigVG`, enter:

```
symlv -g /dev/BigVG list
```

The following is sample output from this command:

```
Volume Group Name: /dev/BigVG
Volume Group Type: HP-UX LVM

                                                        Num      Num Log
         Name        Configuration   State     Cond   Mirrors  Extents
-------------------- -------------  -------  --------  -------  -------

BigLV                Simple         Enabled  Sync       1      2500
```

The command displays the volume group name and the logical volume manager type, and then lists the logical volumes and their attributes.

Note: The above CLI command example is not supported on Linux LVM platforms using the absolute path for volume groups (`/dev/VgName`) if there are no logical volumes present in the volume group.

Logical volume details

The following command lists details (including extent data) about a logical volume of a specified volume group:

```
symlv -g VgName show LvolName
```

For example, to view details about logical volume `BigLV` in volume group `/dev/BigVG`, enter:

```
symlv -g /dev/BigVG show BigLV
```

A sample of the output from this command follows:

```
symlv -g /dev/BigVG show BigLV

  Logical Volume Name            : BigLV
  Logical Volume Pathname        : /dev/BigVG/rBigLV
  Volume State                   : Enabled
  Volume Configuration           : Simple
  Volume Condition               : Sync
  Volume Attributes              : N/A

  Logical Volume Size            : 10000m
```

```
  Number of Logical Extents      : 2500

  Number of Logical Volume Mirrors (1):
     {
  1) Mirror Configuration        : Simple
     Mirror State                : Enabled
     Mirror Flags                : Concat META
     Mirror Condition            : Sync

     Number of device Partitions : 1
     Number of Physical devices  : 1
     Number of Storage devices   : 3

     Mirror Physical Extents (2502):
       {
       ---------------------------------------------------------------------

       Condition     Size  Array Dev    Offset PPdevName                   Offset
       ---------------------------------------------------------------------
       Sync            4m  03003 0123       1m /dev/rdsk/c2t1d5              1m(M)
       Sync            4m  03003 0123       5m /dev/rdsk/c2t1d5              5m(M)
       Sync            4m  03003 0123       9m /dev/rdsk/c2t1d5              9m(M)
       Sync            4m  03003 0123      13m /dev/rdsk/c2t1d5             13m(M)
       Sync            4m  03003 0123      17m /dev/rdsk/c2t1d5             17m(M)
...(etc. many more)
}
     Mirror Physical Devices (3):
       {
       ---------------------------------------------------------------------

       Array Dev   PPdevName               PdevName
       ---------------------------------------------------------------------
       03003 0123  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5                 (M)
       03003 0124  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5                 (m)
       03003 0125  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5                 (m)
...
}
Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

This output shows that the logical volume `BigLV` is created from an underlying Symmetrix concatenated metadevice.

Logical volume conditions

Table 14 describes the possible conditions of a logical volume.

Table 14 Logical volume condition descriptions

Logical volume condition	Description
Degraded RAID5	This condition is for VxVM, DiskADM, and LDM only. One of the devices in the RAID 5 set is unavailable. Inaccessible data is being generated using the parity check formula.
Degraded	In a simple or striped logical volume, there is at least one good copy of data available.
Initializing	The logical volume is initializing.
NeedSync	A mirror is stale.
NoDev	A physical device is unavailable.
Stale	The logical volume contains a mirror with some physical extents that are not consistent with corresponding extents in another mirror.
Sync	All physical extents in all mirrors of the logical volume are synchronized.
SyncInProg	A member of the volume is being regenerated.
Unusable RAID5	This condition is for VxVM, DiskADM, LDM only. The RAID 5 volume is not usable. This implies that access to two subdisks has failed.

Logical volume states

The possible logical volume states are as follows:

- Enabled — The logical volume is available for use.
- Disabled — The logical volume is not available for use.
- Detached — The logical volume is not accessible for I/O operation, but can be enabled again (VxVM only).

Logical volume attributes

The possible logical volume attributes are as follows:

- Multipath — The logical volume has data stored on a Symmetrix device, which has multiple paths from the host.
- DRL enabled — This condition is for VERITAS VxVM only. The logical volume has enabled logging of volume updates to the dirty region log (DRL) to improve recovery of data among mirrors after any system failures.
- Read only — The logical volume has been limited to read-only access.

Logical volume control operations

The `symlv` command provides contains the following logical volume control operations:

`create` — Creates a logical volume of the specified type.

`delete` — Deletes a logical volume.

`add` — Adds mirror images to a logical volume.

`remove` — Removes mirrors of a logical volume.

`extend` — Extends (grows) a logical volume.

`reduce` — Reduces (shrinks) a logical volume.

In addition, performance statistics for logical volumes can be displayed. Refer to Chapter 6, "Statistics SRM," for more information.

Creating a logical volume

The following command creates a logical volume:

```
symlv -g VgName create LVolName -size Size
```

The *`LVolName`* is the name of the logical volume, *`VgName`* is the logical volume group name, and *`Size`* is the size of the logical volume in 512 byte blocks. An optional suffix can be specified to indicate the unit of size measurement. The optional suffixes supported are:

- `b` size in blocks
- `k` size in KBs
- `m` size in MBs

For example, to create a simple volume called `testlv` of size 500 MB in the volume group named `testvg`, enter:

```
symlv -g testvg create testlv -size 500m
```

Note: When creating a logical volume with Solstice Volume Manager, at least one device must be specified using the `-pd` option.

Table 15 lists other options for the logical volume control commands.

Table 15 Options for logical volume control operations

Option	Description
`-nmirror`	The number of mirrors in the logical volume.
`-mir`	The name of the mirror to remove.
`-striped | RAID5`	The type of logical volume; striped or RAID 5.
`-ncols`	The number of stripe columns.

Table 15 Options for logical volume control operations (continued)

Option	Description
`-strsize`	The size of each stripe column in 512 byte blocks.
`-pd`	A list of the device names used for the operation.
`-type`	Volume group type. The default *VgType* is assumed if no *VgType* is specified.

Deleting a logical volume

The following command deletes a logical volume of the specified type:

```
symlv -g VgName -type VgType delete LVolName
```

The *LVolName* is the name of the logical volume and *VgName* is the logical volume group name.

For example, to delete a volume called `testlv` from the volume group named `testvg`, enter:

```
symlv -g testvg -type HP_VXVM delete testlv
```

Adding and removing mirrors

The following command adds a mirror image to a logical volume of the specified type:

```
symlv -g VgName add LVolName -nmirror Mirrors
```

The options `-striped|RAID5`, `-ncols`, `-strsize`, and `-pd can be used` with this command (see Table 15).

The following command removes a mirror from a logical volume of the specified type:

```
symlv -g VgName remove LVolName -nmirror Mirrors
```

This command can use the option `-mir` for the name of the mirror to remove .

Extending and reducing logical volumes

The following command extends (grows) a logical volume of the specified type:

```
symlv -g VgName extend LVolName -size Size
```

This command can use the `-pd` option to list the physical device name(s).

To reduce (shrink) a logical volume of the specified type, use the following form:

```
symlv -g VgName reduce LVolName -size Size
```

This command can use the `-pd` option to list the physical device name(s).

Viewing extents

The following command lists the physical extents associated with each mirror:

```
symlv -g VgName show LvolName [-expand]
```

Along with higher level logical-volume-specific information, this command also displays the physical extents of each of the mirrors, as shown in the following sample output:

```
symlv -g /dev/BigVG show BigLV -expand

Logical Volume Name            : BigLV
Logical Volume Pathname        : /dev/BigVG/rBigLV
Volume State                   : Enabled
Volume Configuration           : Simple
Volume Condition               : Sync
Volume Attributes              : N/A

Logical Volume Size            : 10000m

Number of Logical Extents      : 2500

Number of Logical Volume Mirrors (1):
   {
1) Mirror Configuration        : Simple
   Mirror State                : Enabled
   Mirror Flags                : Concat META
   Mirror Condition            : Sync

   Number of device Partitions : 1
   Number of Physical devices  : 1
   Number of Storage devices   : 3

   Mirror Physical Extents (2502):
     {
     ----------------------------------------------------------------------
     Condition     Size  Array Dev    Offset PPdevName                 Offset
     ----------------------------------------------------------------------
     Sync            4m  03003 0123       1m /dev/rdsk/c2t1d5              1m(M)
     Sync            4m  03003 0123       5m /dev/rdsk/c2t1d5              5m(M)
     Sync            4m  03003 0123       9m /dev/rdsk/c2t1d5              9m(M)
     Sync            4m  03003 0123      13m /dev/rdsk/c2t1d5             13m(M)
(...
}
```

If the Symmetrix device containing the physical extent is determined to be a metadevice, and the extent spans multiple metadevice members, the extent will be broken into multiple extent definitions. Each new definition will describe the portion of the extent that resides on a unique Symmetrix metadevice member.

If the physical extent is part of a logical volume that does not reside on a Symmetrix or CLARiiON device, the *Array* and *Dev* fields will display as *"N/A".*

The output for the extents indicates the attributes of the device on which the extent resides. The attributes are described using a legend block at the end of the output. The attributes are:

- (C) CLARiiON device
- (S) Symmetrix device
- (M) Symmetrix device metahead
- (m) Symmetrix device metamember

Extent conditions

Table 16 describes the possible conditions of a logical volume extent.

Table 16 Logical volume extent conditions

Condition	Description
Initializing	The extent is initializing.
Offline	The extent is offline.
Resync	The physical extents are being resynchronized.
Stale	The extent contains a mirror with some physical extents that are not consistent with corresponding extents in another mirror.
Sync	All physical extents in all mirrors of the logical volume are synchronized.

Expanded list

The `-expand` option, expands display of extent information.

With the default option (`-collapse`), any physical extents that are stored contiguously on disk (which represents contiguous logical extents in the volume) collapse together to form one extent. The size in the extent is adjusted to reflect the total size of the included extents. The offset is offset of the first extent in the collapsed set, resulting in a reduction in the mirror description of the number of physical extents.

Note: In a striped mirror, all extents that are physically contiguous on one physical device partition are collapsed into one extent. The extent list is not logically contiguous.

With the `-expand` option, and the mirror's configuration type is striped, the extent list is revised to present a unique extent description for each stripe in the mirror. The extent size is the stripe size. The count of the physical extents is the size of the logical volume divided by the size of the stripe.

No extents in list

The attributes of a logical volume mirror can be displayed without the details of extent information.

For example, use the `-no_extents` option to show nonextent LVM mirror detail of a particular volume, enter:

```
symlv -g Prodvg show vol1 -no_extents
```

The following is sample output from this command:

```
symlv -g /dev/BigVG show BigLV -no_extents

Logical Volume Name            : BigLV
Logical Volume Pathname        : /dev/BigVG/rBigLV
Volume State                   : Enabled
Volume Configuration           : Simple
Volume Condition               : Sync
Volume Attributes              : N/A
```

```
  Logical Volume Size            : 10000m

  Number of Logical Extents      : 2500

  Number of Logical Volume Mirrors (1):
     {
  1) Mirror Configuration        : Simple
     Mirror State                : Enabled
     Mirror Flags                : Concat META
     Mirror Condition            : Sync

     Number of device Partitions : 1
     Number of Physical devices  : 1
     Number of Storage devices   : 3

     Mirror Physical Extents (0):

     Mirror Physical Devices (3):
       {
       ---------------------------------------------------------------------

       Array Dev   PPdevName               PdevName
       ---------------------------------------------------------------------
       03003 0123  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5                 (M)
       03003 0124  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5                 (m)
       03003 0125  /dev/rdsk/c2t1d5        /dev/rdsk/c2t1d5                 (m)
...
}
```

CHAPTER 6
Statistics SRM

This chapter describes statistics and performance information for host-based SRM objects and explains how to generate them using SYMCLI commands.

- Retrieving statistics 106

Retrieving statistics

The Solutions Enabler SYMCLI commands can retrieve statistics and performance information for various host-based SRM objects. Specifically, statistics can be obtained for the host's CPU, memory, disks, logical volume manger, and databases.

The SRM statistics commands query the host system to capture and display performance statistics.

Supported metrics

While statistics are the actual measured counts, metrics are the units and measures those counts represent (such as, number of memory swap-outs).

Each host, for which statistics can be returned, may support a different set of metrics. Currently, the supported platforms are: SunOS, HP-UX, AIX, Tru64 UNIX, Windows, and Linux.

The `symhost stats` command retrieves the specified (CPU, memory, disk, or all) metrics and displays them with the current time stamp.

The `symlv stats` command retrieve the volume I/O statistics that can help in performance monitoring. The amount of information retrieved depends on the LVM type and the operating system of the host.

The `symrdb stats` command retrieves performance statistics for the specified database type. The database types can have different metrics. Currently, the supported database types for statistics are Oracle, SQL Server, Sybase, and IBM DB2/UDB.

These commands provide options to how often and how long to display to display the statistics. The `-i` (interval) option indicates how often to display the statistics (in seconds) and the `-c` (count) option indicates the number of times to display statistics. If the count is not specified, and an interval (`-i`) is specified, statistics will display indefinitely.

Table 17 lists the SRM statistics commands.

Table 17 SRM statistics commands

Command	Argument	Actions
`symhost`	`show` `stats`	Displays host configuration information. Displays performance statistics.
`symlv`	`stats`	Displays performance statistics about logical volumes.
`symrdb`	`stats`	Displays performance statistics about the specified database.

symhost examples

The following three examples show the command and system output when retrieving statistics for the host CPU, memory, and disk.

To display statistics about all host processors every 30 seconds for one hour, enter:

```
symhost stats -i 30 -c 120 -type CPU
```

The following is an example of the output:

```
13:32:33      CPU    %User     %Sys     %WIO    %Idle    Int/s  Calls/s  CtxSw/s
13:32:33        0      0.0      0.0    100.0      0.0    401.7      1.6      9.3
13:32:33        2      0.0      0.1      0.0     99.8    101.4      9.2     84.7
13:33:03        0      0.0      0.0    100.0      0.0    401.0      8.0     35.9
13:33:03        2      0.0      0.2      0.0     99.7    101.7     17.1     70.3
```

Where the first column provides the time of day, and the other columns (from left to right) are as follows:

- CPU — CPU number/ID
- %User — 100 * (CPU busy time in user mode / elapsed time)
- %Sys — 100 * (CPU busy time in system mode / elapsed time)
- %WIO — 100 * (CPU idle time for wait I/O / elapsed time)
- %Idle — 100 * (CPU idle time / elapsed time)
- Int/s — Interrupts per second
- Calls/s — System calls per second
- CtxSw/s — Process context switches per second

To display statistics about host memory every 30 seconds for one hour, enter:

```
symhost stats -i 30 -c 120 -type MEMORY
```

The following is an example of the output:

```
13:22:18      Pi/s    Ppi/s     Po/s    Ppo/s     Si/s    Psi/s     So/s    Psos
13:22:18     716.6   1340.0      3.4      5.3      0.0      0.0      0.0      0.0
13:22:48     716.6   1340.0      3.4      5.3      0.0      0.0      0.0      0.0
```

Where the first column provides the time of day, and the other columns (from left to right) are as follows:

- Pi/s — Page in requests per second
- Ppi/s — Number of pages paged in per second
- Po/s — Page out requests per second
- Ppo/s — Number of pages paged out per second
- Si/s — Swap in requests per second
- Psi/s — Number of pages swapped in per second
- So/s — Swap out requests per second
- Psos — Number of pages swapped out per second

To display statistics about all host disks every 30 seconds for one hour, enter:

```
symhost stats -i 30 -c 120 -type DISK
```

The following is an example of the output:

```
14:09:01     DISK     RW/s   R/s    W/s    KbRW/s  KbR/s   KbW/s   %Busy   %Wait
14:09:01     c0t6d0   0.0    0.0    0.0    0.0     0.0     0.0      0.0     0.0
14:09:01     c0t0d0   0.0    0.0    0.0    0.0     0.0     0.0      0.0     0.0
```

Where the first column provides the time of day, and the other columns (from left to right) are as follows:

- DISK — Disk name
- RW/s — Read and write requests per second
- R/s — Read requests per second
- W/s — Write requests per second
- KbRW/s — KB read and written per second
- KbR/s — KB read per second
- KbW/s — KB written per second
- %Busy — 100 * (disk active time/elapsed time)
- %Wait — 100 * (nonempty wait queue time/elapsed time)

symlv example

The following examples show the command and system output when retrieving logical volume manager statistics.

To display statistics about all logical volumes of VxVM on SunOS every 30 seconds for one hour, enter:

```
symlv stats -i 30 -c 120 -type SUN_VXVM
```

The following is an explanation of the output headings:

```
Volume Group Type : SunOS VxVM

H:M:S VgName LvName RW/s R/s W/s KbRW/s KbR/s KbW/s %Busy %Wait
```

Where the first column provides the time of day, and the other columns (from left to right) are as follows:

- VgName — Volume group name
- LvName — Logical volume name
- RW/s — Read and write requests per second
- R/s — Read requests per second
- W/s — Write requests per second
- KbRW/s — KB read and written per second
- KbR/s — KB read per second
- KbW/s — KB written per second

- %Busy — 100 * (logical volume active time / elapsed time)
- %Wait — 100 * (non-empty wait queue time / elapsed time)

Note: Statistics are not available for AIX_LVM, LINUX_LVM, and DYNIX_SVM.

symrdb example

The following example displays statistics about a session of a specified Oracle database every 30 seconds for one hour:

```
symrdb stats -type ORACLE -i 30 -c 120 -target SESSION
```

The following is an explanation of the output headings:

```
H:M:S      sessionID   memory
```

Where the first column provides the time of day, and the other columns (from left to right) are as follows:

- session ID — database session ID
- memory — memory usage per second

Note: All the statistics values are per second.

Database metrics

The following tables list the information that returns from each database metric. There is a separate table for each database type.

Table 18 describes the available metrics for the Oracle database.

Table 18 Metric options for Oracle database types

Oracle metric option	Returns statistics about
INSTANCE	• Instance name • Background checkpoint started • Background checkpoint completed • Database connection count • Total memory usage • Database block buffers • Consistent gets • Consistent changes • Database session • Database file • All of the above
SESSION	• Session ID • Total memory usage
FILE	• Filename • Datafile physical read • Datafile physical write • Datafile block read • Datafile block write • Physical read time • Physical write time
ALL	All of the Oracle metric options

Table 19 describes the available metrics for the SQL Server database.

Table 19 Metric options for SQL Server databases

SQL Server metric option	Returns statistics about
INSTANCE	• Number of physical database page reads issued • Number of physical database page writes issued • Number of pages flushed by checkpoint or other operations that require all dirty pages to be flushed • Number of latch requests that could not be granted immediately and had to wait before being granted • Average latch wait time for latch requests that had to wait, in milliseconds • Total amount of dynamic memory the server is using for the dynamic SQL cache in KB • Read/write throughput for a backup device, in KB
DATABASE	• Name of the database • Reads performed through the log manager cache • Read/write throughput for backup/restore of a database • Total number of log bytes flushed • Number of commits waiting on log flush • Total wait time for log flush, in milliseconds
OBJECT	• Object name • Number of lock requests that could not be satisfied immediately and required the caller to wait before being granted the lock • Total wait time for locks in the last second in milliseconds • The average amount of wait time for each lock request that resulted in a wait in milliseconds • Number of lock requests that resulted in a deadlock
FILE	• Filename • Number of reads issued on the file • Number of writes issued on the file • KB of read issued on the file • KB of write issued on the file • Total amount of time that users waited for the I/Os to complete on the file in milliseconds
ALL	All of the SQL Server metric options

Table 20 describes the available metrics for the Sybase database.

Table 20 Metric options for Sybase databases

Sybase metric option	Returns statistics about:
SERVER	• Name of the Adaptive Server that is being monitored • Memory allocated for the page cache (in bytes) • Memory allocated for Adaptive Server • CPU busy time • Number of locks
OBJECT	• Name of the database • Name of the database object • Number of data page reads, whether satisfied from cache or from a database device • Number of data page reads that could not be satisfied from the data cache • Number of data pages written to a database device • Number of combined logical page reads and page writes • Number of locks that were granted after waiting for another lock to be released • Number of locks that were granted immediately
DEVICE	• Device name • Number of reads made from a database device • Number of writes made to a database device • Number of times access to a device was granted • Number of times that access to a device had to wait
ALL	All of the Sybase metric options

Table 21 describes the available metrics for the IBM DB2/UDB database.

Table 21 Metric options for IBMUDB databases

IBMUDB metric option	Returns statistics about
DATABASE	• Database name • Buffer pool logical data page reads • Buffer pool physical data page reads • Buffer pool logical index page reads • Buffer pool physical index page reads • Buffer pool data page writes • Buffer pool async data page reads • Buffer pool number async read requests • Buffer pool asynch data page writes • Buffer pool async index page reads • Buffer pool async index page writes • Buffer pool async read time • Buffer pool async write time • Log page read • Log page write
TABLE	• Database name • Table name • Table type • Number of changes to the table • Number of reads from the table • Number of accesses to overflow record
TABLESPACE	• Database name • Tablespace name • Buffer pool logical data page reads • Buffer pool physical data page reads • Buffer pool data page writes • Buffer pool logical index page reads • Buffer pool physical index page reads • Buffer pool index page writes • Buffer pool sync data page reads • Number of async read requests • Buffer pool async data page writes • Buffer pool async index page reads • Buffer pool async index page writes • Buffer pool async read time • Buffer pool async write time
ALL	All of the IBMUDB metric options

PART 2
Operational Examples

Part 2 contains the following:

Chapter 7, “SRM Examples”

Contains examples of using the Solutions Enabler CLI commands to perform Storage Resource Management operations.

CHAPTER 7
SRM Examples

This chapter contains examples of using the Solutions Enabler CLI commands to perform Storage Resource Management operations.

- Example 1: Displaying relational database objects 118
- Example 2: Mapping files and other disk storage objects 125
- Example 3: Displaying volume groups and logical volumes..... 137
- Example 4: Deporting and importing a volume group 142
- Example 5: Mapping files 144
- Example 6: Displaying a logical volume 147

Example 1: Displaying relational database objects

This example examines database objects for three different types of databases: Oracle, IBM UDB, and Informix.

Examining Oracle database objects

The hardware/software setup for this example consists of a Solaris host connected to a local Symmetrix array. Oracle version 8.1.7.0.0 software is installed on the host.

For Solutions Enabler to access a database, set the SYMCLI_RDB_CONNECT environment variable to the username and password of the system administrator's account. This first export command sets the variable to a username of "system" and a password of "manager."

```
export SYMCLI_RDB_CONNECT=system/manager
```

Setting ORACLE_HOME specifies the location of the Oracle binaries. Setting ORACLE_SID specifies the database name (64817).

```
export ORACLE_HOME=/db1/solaris/2.6/oracle/8.1.7

export ORACLE_SID=64817
```

The `symrdb list` command lists the current Oracle database (64817) and allows testing of basic database connectivity.

```
symrdb list -type oracle


DATABASE NAMES (ORACLE 8.1.7.0.0):

  Database Name
  -------------

  64817
```

The `symrdb list file` command identifies all the files of the current Oracle database.

```
symrdb list file -type oracle


DATABASE FILE NAMES (ORACLE 8.1.7.0.0):

  DB File Name                              Type       Status
  --------------                            ------     --------

  /usr/oracle/oradata/64817/control01.ctl  Control    Online
  /usr/oracle/oradata/64817/rbs01.dbf      Data       Online
  /usr/oracle/oradata/64817/system01.dbf   Data       Online
  /usr/oracle/oradata/64817/temp01.dbf     Data       Online
  /usr/oracle/oradata/64817/redo01.log     Log        Online
  /usr/oracle/oradata/64817/redo02.log     Log        Online
  /usr/oracle/oradata/64817/redo03.log     Log        Online
```

The `symrdb list table` command displays all tables in the tablespace named SYSTEM. The ellipsis (...) indicates where output was omitted for brevity.

```
symrdb list table -type oracle -tbs SYSTEM
```

```
TABLE SPACE NAME     : SYSTEM
DATABASE TABLE NAMES (ORACLE 8.1.7.0.0):

  Table Name                     Owner                          Type
  -------------                  ------                         --------

  OL$                            OUTLN                          Simple Table
  OL$HINTS                       OUTLN                          Simple Table
  ACCESS$                        SYS                            Simple Table
  AQ$_MESSAGE_TYPES              SYS                            Simple Table
  AQ$_PENDING_MESSAGES           SYS                            Simple Table
  AQ$_PROPAGATION_STATUS         SYS                            Simple Table
  AQ$_QUEUE_STATISTICS           SYS                            Simple Table
  AQ$_QUEUE_TABLE_AFFINITIES     SYS                            Simple Table
  AQ$_SCHEDULES                  SYS                            Simple Table
  ARGUMENT$                      SYS                            Simple Table
  ASSOCIATION$                   SYS                            Simple Table
  ATTRCOL$                       SYS                            Cluster
  ATTRIBUTE$                     SYS                            Cluster
  AUD$                           SYS                            Simple Table
  AUDIT$                         SYS                            Simple Table
  AUDIT_ACTIONS                  SYS                            Simple Table
  BOOTSTRAP$                     SYS                            Simple Table
  CCOL$                          SYS                            Cluster
  CDEF$                          SYS                            Cluster
  CLU$                           SYS                            Cluster
  COL$                           SYS                            Cluster
  COLLECTION$                    SYS                            Cluster
  COLTYPE$                       SYS                            Cluster
  COM$                           SYS                            Simple Table
  CON$                           SYS                            Simple Table
  CONTEXT$                       SYS                            Simple Table
  DBMS_ALERT_INFO                SYS                            Simple Table
  DBMS_LOCK_ALLOCATED            SYS                            Simple Table
  ..............................................................................
  AQ$_QUEUES                     SYSTEM                         Simple Table
  AQ$_QUEUE_TABLES               SYSTEM                         Simple Table
  AQ$_SCHEDULES                  SYSTEM                         Simple Table
  DEF$_AQCALL                    SYSTEM                         Simple Table
  DEF$_AQERROR                   SYSTEM                         Simple Table
  DEF$_CALLDEST                  SYSTEM                         Simple Table
  DEF$_DEFAULTDEST               SYSTEM                         Simple Table
  DEF$_DESTINATION               SYSTEM                         Simple Table
  DEF$_ERROR                     SYSTEM                         Simple Table
  DEF$_LOB                       SYSTEM                         Simple Table
  DEF$_ORIGIN                    SYSTEM                         Simple Table
  DEF$_PROPAGATOR                SYSTEM                         Simple Table
  DEF$_PUSHED_TRANSACTIONS       SYSTEM                         Simple Table
  DEF$_TEMP$LOB                  SYSTEM                         Simple Table
  SQLPLUS_PRODUCT_PROFILE        SYSTEM                         Simple Table
```

Use the `symrdb list tbs` command to display tablespaces, their allocated size, and their free space. Megabytes (`mb`) is the default.

```
symrdb -type oracle list tbs
```

```
TABLE SPACE NAMES (ORACLE 8.1.7.0.0):

  Table Space  Name              Type       Status     Size (mb)  Free (mb)
  -------------------            ------     --------   --------------------
  LOTSAFILES1                    Permanent  Online             3          1
  LOTSAFILES10                   Permanent  Online             3          1
  LOTSAFILES11                   Permanent  Online             3          1
  LOTSAFILES12                   Permanent  Online             3          1
  LOTSAFILES13                   Permanent  Online             3          1
  LOTSAFILES14                   Permanent  Online             3          1
  LOTSAFILES15                   Permanent  Online             3          1
  LOTSAFILES16                   Permanent  Online             3          1
  LOTSAFILES17                   Permanent  Online             3          1
  LOTSAFILES18                   Permanent  Online             3          1
  LOTSAFILES19                   Permanent  Online             3          1
  LOTSAFILES2                    Permanent  Online             3          1
  LOTSAFILES20                   Permanent  Online             3          1
  LOTSAFILES21                   Permanent  Online             3          1
  LOTSAFILES22                   Permanent  Online             0          0
  LOTSAFILES3                    Permanent  Online             3          1
  LOTSAFILES4                    Permanent  Online             3          1
  LOTSAFILES5                    Permanent  Online             3          1
  LOTSAFILES6                    Permanent  Online             3          1
  LOTSAFILES7                    Permanent  Online             3          1
  LOTSAFILES8                    Permanent  Online             3          1
  LOTSAFILES9                    Permanent  Online             3          1
  SYSTEM                         Permanent  Online            75         23
  USERS                          Permanent  Online             1          0
```

Examining IBM DB2/UDB database objects

The hardware/software setup for this example consists of a Solaris host connected to a local Symmetrix array. IBM UDB version 7.1.0 software is installed on the host.

For Solutions Enabler to access a database, set the SYMCLI_RDB_CONNECT environment variable to the username and password of the system administrator's account. This first `export` command sets the variable to a username of `db2v7in1` and a password of `redsox`.

```
export SYMCLI_RDB_CONNECT=db2v7in1/redsox
```

When running a DB2/UDB instance with Solaris, set LD_LIBRARY_PATH as follows. Also, set the variable DB2INSTANCE to the name of the database instance (for example, `db2v7in1`).

```
export LD_LIBRARY_PATH=/usr/db2v7in1/sqllib/lib
```

```
export DB2INSTANCE=db2v7in1
```

The `symrdb list` command lists the current databases (QU1 and SAMPLE) within the IBM UDB database instance defined above (`db2v7in1`).

```
symrdb list -type ibmudb

DATABASE NAMES (IBMUDB 7.1.0):

  Database Name
  -------------

  QU1
  SAMPLE
```

The `symrdb list tbs` command displays tablespaces in the database named SAMPLE. Note that the database (`-db`) name is case sensitive.

```
symrdb list tbs -type ibmudb -db SAMPLE
DATABASE          : SAMPLE
TABLE SPACE NAMES (IBMUDB 7.1.0):

  Table Space  Name              Type       Status
  -------------------            ------     --------

  SYSCATSPACE                    Permanent  Online
  TEMPSPACE1                     Temporary  Online
  USERSPACE1                     Permanent  Online
```

Examining Informix database objects

The hardware/software setup for this example consists of a Solaris host connected to a local Symmetrix array. Informix version 7.3.1 software is installed on the host.

For Solutions Enabler to access a database, set the SYMCLI_RDB_CONNECT environment variable to the username and password of the system administrator's account. The export command sets the variable to a username of "informix" and a password of "Informix."

```
export SYMCLI_RDB_CONNECT=informix/Informix
```

The export INFORMIXDIR command specifies the location of the Informix binaries. The export ONCONFIG command specifies the Informix configuration file. The export INFORMIXSERVER command specifies the server name (that is, the test64731 database instance).

```
export INFORMIXDIR=/db1/solaris/2.6/informix/7.3.1

export ONCONFIG=onconfig.test64731

export INFORMIXSERVER=test64731
```

The `symrdb list` command lists current databases for the Informix server defined above (`test64731`).

```
symrdb list -type informix

DATABASE NAMES (INFORMIX 7.31.UD1):

  Database Name
  -------------

  sysmaster
  sysutils
```

The `symrdb show file` command locates extent data on a sysmaster database file named `llog.dbs`, including the physical devices on which the file data extents reside and the mirror configuration at the operating system level. "Database File Block Size" means that this version of Informix reads a minimum of four blocks on accessing the file. This file is configured as a striped mirror with two columns and a stripe size of 256 blocks. Each column maps to a different physical disk.

```
symrdb show -type informix -db sysmaster -blocks file /
/demofs/informix/test64731/llog.dbs


DATABASE            : sysmaster
DATABASE FILE NAMES (INFORMIX 7.31.UD1):

Database File Name : /demofs/informix/test64731/llog.dbs

  Database File Type              : Log
  Database File Status            : Online
  Database File Size              : 4000b
  Database File Data Offset       : 0b
  Database File Block Size        : 4b

    Absolute Path                : /demofs/informix/test64731/llog.dbs
    Resolved Object Type         : SunOS UFS File
    Resolved Object Size         : 4000b
    Number of Trailing Bytes     : 0
    Extent byte offset to data   : 0
    Number of Physical Devices   : 2
    File System Mount Point      : /demofs
    File System Device Name      : /dev/vx/dsk/doug/Striped

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration

         Mirror Stripe Columns    : 2
         Mirror Stripe Size       : 256b
```

As this example illustrates below, it is not necessary for a file's extents to align with the boundaries of a striped logical volume. The first three extents on Dev 00BC are "backwards" in the stripe. The fourth and fifth extents on Dev 00BC begin 1008 blocks later (from Offset 35728 to 36736). Finally, the first two extents are contiguous but in reversed order and, therefore, are not collapsed for this "expanded" list.

```
Mirror Physical Extents (20):
{
-----------------------------------------------------------------------

      Size  Array Dev    Offset PPdevName                   Offset
-----------------------------------------------------------------------
       80b  00541 00BC   35728b /dev/vx/rdmp/c2t0d0s2       35728b (S)
       48b  00541 00BC   35680b /dev/vx/rdmp/c2t0d0s2       35680b (S)
       16b  00541 00BC   35648b /dev/vx/rdmp/c2t0d0s2       35648b (S)
       48b  00541 00BC   36736b /dev/vx/rdmp/c2t0d0s2       36736b (S)
      128b  00541 00BC   36800b /dev/vx/rdmp/c2t0d0s2       36800b (S)
      256b  00541 00BD   35712b /dev/vx/rdmp/c2t0d1s2       35712b (S)
      256b  00541 00BC   36928b /dev/vx/rdmp/c2t0d0s2       36928b (S)
      256b  00541 00BD   35968b /dev/vx/rdmp/c2t0d1s2       35968b (S)
      256b  00541 00BC   37184b /dev/vx/rdmp/c2t0d0s2       37184b (S)
      256b  00541 00BD   36224b /dev/vx/rdmp/c2t0d1s2       36224b (S)
      ..........................................................................
      256b  00541 00BD   36992b /dev/vx/rdmp/c2t0d1s2       36992b (S)
      256b  00541 00BC   38208b /dev/vx/rdmp/c2t0d0s2       38208b (S)
      256b  00541 00BD   37248b /dev/vx/rdmp/c2t0d1s2       37248b (S)
      256b  00541 00BC   38464b /dev/vx/rdmp/c2t0d0s2       38464b (S)
       96b  00541 00BD   37504b /dev/vx/rdmp/c2t0d1s2       37504b (S)
}
Mirror Physical Devices (2):
{
-----------------------------------------------------------------------

Array Dev   PPdevName                PdevName
-----------------------------------------------------------------------
00541 00BC  /dev/vx/rdmp/c2t0d0s2   /dev/vx/rdmp/c2t0d0s2      (S)
00541 00BD  /dev/vx/rdmp/c2t0d1s2   /dev/vx/rdmp/c2t0d1s2      (S)
}
```

Examining Microsoft Exchange database objects

Microsoft Exchange database objects can be listed. The hardware/software setup for this example consists of a Windows 2000 host connected to a local Symmetrix array. Microsoft Exchange version 6.0.4712.2 software is installed on the host.

Microsoft Exchange does not require setting the SYMCLI_RDB_CONNECT variable or any database environmental variables in order to accessing the database.

The following `symrdb list file` command displays database files in an Exchange storage group named `SG2`.

```
symrdb -type exchange list file -tbs SG2

TABLE SPACE NAME    : SG2
DATABASE FILE NAMES (EXCHANGE 6.0.4712.2):

  DB File Name                              Type       Status
  --------------                            ------     --------

  T:\SG2PrivA.edb                           Data       Online
  T:\SG2PrivA.stm                           Data       Online
  T:\SG2PrivB.edb                           Data       Online
  T:\SG2PrivB.stm                           Data       Online
```

```
L:\E01.log                                  Log        Online
L:\E0100009.log                             Log        Online
L:\E010000A.log                             Log        Online
L:\res1.log                                 Log        Online
L:\res2.log                                 Log        Online
```

Example 2: Mapping files and other disk storage objects

The hardware setup for this example consists of a Sun Solaris host connected to a Symmetrix array. The files displayed are within a Sun UFS file system. The SunOS VxVM logical volume manager manages a logical volume included in the example. In this example and all examples sections, an object is mapped to the Symmetrix device level (in some cases, to the Symmetrix metadevice level).

The `symhostfs list` command with the `-file` option displays all of the files in a specified directory (`/fs_test1`). The display includes file size and access information.

```
symhostfs list -file /fs_test1

Directory Name  : /fs_test1
FileSystem Type : SunOS UFS File

  File Name                                Size    Perms     Owner  Group
  ---------                                ----  ----------  -----  -----

  file1                                      1m  -rw-r--r--      0      1
  file1k                                     1k  -rw-r--r--      0      1
```

The `symhostfs show` command expands the extents of a file named `file1` and displays that extent data in blocks (1500) along with other characteristics of the file. "Device Name" in the display indicates the physical device on which the file is stored.

```
symhostfs show /fs_test1/file1 -expand -blocks


  File Name                       : /fs_test1/file1
  File Type                       : SunOS UFS File
  File Size                       : 1500b
  Number of Trailing Bytes        : 0
  Extent byte offset to data      : 0

  File Mode                       : 100644
  File Permission                 : -rw-r--r--
  File Owner ID                   : 0
  File Group ID                   : 1

  Number of Symbolic Links        : 1

  Last Access Time                : Fri 12-Apr-2002 16:16
  Last Modification Time          : Fri 12-Apr-2002 16:16
  Last Status Change Time         : Fri 12-Apr-2002 16:16

  Device Name                     : /dev/vx/rdsk/fsvg/Striped
  Inode Number                    : 4
  File System Fragment Size       : 2b
```

"Offset in File" shows the ascending sequence of the logical extents in blocks (b). "Offset in Device" relates to the host device or logical volume and has no predictable sequence. The ellipsis (...) indicates where output was omitted for brevity. "Metadata," while associated with the file, is not technically part of the file and thus is "N/A" in the column "Offset in File."

```
Number of Extents in File       : (95)
    {
    ----------------------------------------
    Extent    Offset in    Extent      Offset
    Type      Device       Size        in File
    ----------------------------------------
    Data          1680b       16b           0b
    Data          1696b       16b          16b
    Data          1712b       16b          32b
    Data          1728b       16b          48b
    Data          2000b       16b          64b
    Data          1776b       16b          80b
    Data          1792b       16b          96b
    Data          1824b       16b         112b
    Data          2208b       16b         128b
    Data          2224b       16b         144b
    Data          2176b       16b         160b
    Data          2320b       16b         176b
    Metadata    100368b       16b          N/A
    Data        100432b       16b         192b
    Data        100384b       16b         208b
    ........................................................
    Data        103824b       16b        1472b
    Data        103840b       12b        1488b
    }
```

SRM locates the physical extents on the physical devices where the file data resides and displays the mirror configuration at the operating system level, not the storage system level. (Data management software on the host has no knowledge of storage system mirrors - for example, Symmetrix mirrors M1, M2, M3, and M4. However, each layer on the host has the option to mirror some or all of its extents on the layer just below it.) In this case, an LVM on the host configured the data object as one mirror, meaning that data blocks for this object are written to one place. The LVM also configured the object as a striped mirror with four columns and a stripe size of 32 blocks. Each column maps to a different physical disk.

```
Absolute Path                 : /fs_test1/file1
Resolved Object Type          : SunOS UFS File
Resolved Object Size          : 1500b
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 4
File System Mount Point       : /fs_test1
File System Device Name       : /dev/vx/dsk/fsvg/Striped

Number of Mirrors for object (1):
   {
1) Mirror Configuration

     Mirror Stripe Columns    : 4
     Mirror Stripe Size       : 32b
```

The four Symmetrix device names (00A0, 00A4, 00A8, and 0009) correspond to host physical device names (PPdevName): /dev/vx/rdmp/c4t2d0s2, /c4t2d1s2, /c4t2d2s2, and /c5t0d1s2. The Symmetrix devices were configured arbitrarily as three metadevices and one simple Symmetrix device to illustrate that physical disks do not have to be the same configuration type, and may actually be reconfigured as storage needs expand. The "Dev Offset" value here is the same as the "PPdevName Offset" value because the disks are not partitioned. When disks are partitioned, these offsets will be different. Column "Array" provides the ID of the storage array on which the disk is located, in this case, Symmetrix arrays 00541 and 03122.

```
Mirror Physical Extents (94):
{
-----------------------------------------------------------------------

      Size  Array Dev    Offset PPdevName                    Offset
-----------------------------------------------------------------------
       16b  00541 00A0    3312b /dev/vx/rdmp/c4t2d0s2         3312b (S)
       16b  00541 00A4    3296b /dev/vx/rdmp/c4t2d1s2         3296b (S)
       16b  00541 00A4    3312b /dev/vx/rdmp/c4t2d1s2         3312b (S)
       16b  00541 00A8    3296b /dev/vx/rdmp/c4t2d2s2         3296b (S)
       16b  00541 00A8    3376b /dev/vx/rdmp/c4t2d2s2         3376b (S)
       16b  03122 0009    3312b /dev/vx/rdmp/c5t0d1s2         3312b (S)
       16b  00541 00A0    3328b /dev/vx/rdmp/c4t2d0s2         3328b (S)
       16b  00541 00A4    3328b /dev/vx/rdmp/c4t2d1s2         3328b (S)
       16b  00541 00A4    3424b /dev/vx/rdmp/c4t2d1s2         3424b (S)
       16b  00541 00A4    3440b /dev/vx/rdmp/c4t2d1s2         3440b (S)
       16b  03122 0009   28800b /dev/vx/rdmp/c5t0d1s2        28800b (S)
       16b  03122 0009   28816b /dev/vx/rdmp/c5t0d1s2        28816b (S)
       16b  00541 00A0   28832b /dev/vx/rdmp/c4t2d0s2        28832b (S)
       16b  00541 00A0   28848b /dev/vx/rdmp/c4t2d0s2        28848b (S)
       12b  00541 00A4   28832b /dev/vx/rdmp/c4t2d1s2        28832b (S)
}

Mirror Physical Devices (4):
{
-----------------------------------------------------------------------

Array Dev   PPdevName                PdevName
-----------------------------------------------------------------------
00541 00A0  /dev/vx/rdmp/c4t2d0s2   /dev/vx/rdmp/c4t2d0s2      (M)
00541 00A4  /dev/vx/rdmp/c4t2d1s2   /dev/vx/rdmp/c4t2d1s2      (M)
00541 00A8  /dev/vx/rdmp/c4t2d2s2   /dev/vx/rdmp/c4t2d2s2      (M)
03122 0009  /dev/vx/rdmp/c5t0d1s2   /dev/vx/rdmp/c5t0d1s2      (V)
}
}

Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

The symrslv dir command examines a host directory named /usr within the root file system (/). Solaris, at installation time, mounted the root file system on slice 0 (s0) of an internal disk. The size of the directory is two blocks (2b), and its single collapsed extent begins at offset 936896b of /dev/rdsk/c0t0d0s0. By default, the directory's extents

are collapsed. Because SRM is examining a local device here rather than a Symmetrix or CLARiiON device, SRM cannot return Array, Dev, and Dev Offset. Therefore, these columns are not applicable (N/A) with this output display.

```
symrslv dir /usr -blocks

Absolute Path                 : /usr
Resolved Object Type          : Directory
Resolved Object Size          : 2b
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 1
File System Mount Point       : /
File System Device Name       : /dev/dsk/c0t0d0s0

Number of Mirrors for object (1):
{
1) Mirror Configuration
Mirror Physical Extents (1):
{
----------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                   Offset
----------------------------------------------------------------------
2b    N/A  N/A      N/A /dev/rdsk/c0t0d0s0         936896b
}
Mirror Physical Devices (1):
{
----------------------------------------------------------------------

Array Dev   PPdevName               PdevName
----------------------------------------------------------------------
N/A   N/A   /dev/rdsk/c0t0d0s0      /dev/rdsk/c0t0d0s2
}
       }

Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

The `symrslv file` command expands the extents of the file named `file1` and displays its 94 physical extents in blocks (1500 total). Unlike the previous `symhostfs show` command, `symrslv` focuses strictly on the extent data at the Symmetrix device layer. The `symrslv` command drills down directly to the Symmetrix devices (Dev) on which the file is stored and locates the extents there.

```
symrslv file /fs_test1/file1 -expand -blocks

    Absolute Path                 : /fs_test1/file1
    Resolved Object Type          : SunOS UFS File
    Resolved Object Size          : 1500b
    Number of Trailing Bytes      : 0
    Extent byte offset to data    : 0
    Number of Physical Devices    : 4
    File System Mount Point       : /fs_test1
    File System Device Name       : /dev/vx/dsk/fsvg/Striped

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration
```

```
            Mirror Stripe Columns    : 4
            Mirror Stripe Size       : 32b

            Mirror Physical Extents (94):
            {
  ----------------------------------------------------------------------

        Size  Array Dev    Offset PPdevName                   Offset
  ----------------------------------------------------------------------
         16b  00541 00A0    3312b /dev/vx/rdmp/c4t2d0s2        3312b (S)
         16b  00541 00A4    3296b /dev/vx/rdmp/c4t2d1s2        3296b (S)
         16b  00541 00A4    3312b /dev/vx/rdmp/c4t2d1s2        3312b (S)
         16b  00541 00A8    3296b /dev/vx/rdmp/c4t2d2s2        3296b (S)
         16b  00541 00A8    3376b /dev/vx/rdmp/c4t2d2s2        3376b (S)
         16b  03122 0009    3312b /dev/vx/rdmp/c5t0d1s2        3312b (S)
         16b  00541 00A0    3328b /dev/vx/rdmp/c4t2d0s2        3328b (S)
         16b  00541 00A4    3328b /dev/vx/rdmp/c4t2d1s2        3328b (S)
         16b  00541 00A4    3424b /dev/vx/rdmp/c4t2d1s2        3424b (S)
         16b  00541 00A4    3440b /dev/vx/rdmp/c4t2d1s2        3440b (S)
         ......................................................................
         16b  03122 0009   28800b /dev/vx/rdmp/c5t0d1s2       28800b (S)
         16b  03122 0009   28816b /dev/vx/rdmp/c5t0d1s2       28816b (S)
         16b  00541 00A0   28832b /dev/vx/rdmp/c4t2d0s2       28832b (S)
         16b  00541 00A0   28848b /dev/vx/rdmp/c4t2d0s2       28848b (S)
         12b  00541 00A4   28832b /dev/vx/rdmp/c4t2d1s2       28832b (S)
  }

  Mirror Physical Devices (4):
  {
  ----------------------------------------------------------------------

  Array Dev   PPdevName                PdevName
  ----------------------------------------------------------------------
  00541 00A0  /dev/vx/rdmp/c4t2d0s2   /dev/vx/rdmp/c4t2d0s2      (M)
  00541 00A4  /dev/vx/rdmp/c4t2d1s2   /dev/vx/rdmp/c4t2d1s2      (M)
  00541 00A8  /dev/vx/rdmp/c4t2d2s2   /dev/vx/rdmp/c4t2d2s2      (M)
  03122 0009  /dev/vx/rdmp/c5t0d1s2   /dev/vx/rdmp/c5t0d1s2      (S)
}
   Legend for the Attribute of Devices:

   (C): CLARiiON Device.
   (S): Symmetrix Device.
   (M): Symmetrix Device Meta Head.
   (m): Symmetrix Device Meta member.
   (V): Virtual Disk on VMFS.
```

The `symrslv file` command with the `-collapse` option logically collapses the file's extent data, which means that the file can be reconstructed if a subsequent expand operation is performed. Extents whose size is now greater than 16b were formed from contiguous extents that were collapsed (for example, a 32b extent represents two contiguous 16b extents that were collapsed to form one extent).

symrslv file /fs_test1/file1 -collapse -blocks

```
    Absolute Path                 : /fs_test1/file1
    Resolved Object Type          : SunOS UFS File
    Resolved Object Size          : 1500b
    Number of Trailing Bytes      : 0
    Extent byte offset to data    : 0
    Number of Physical Devices    : 4
    File System Mount Point       : /fs_test1
    File System Device Name       : /dev/vx/dsk/fsvg/Striped

    Number of Mirrors for object (1):
```

```
                    {
                 1) Mirror Configuration

                      Mirror Stripe Columns    : 4
                      Mirror Stripe Size       : 32b

                      Mirror Physical Extents (27):
            {
            ----------------------------------------------------------------------

                  Size  Array Dev    Offset PPdevName                    Offset
            ----------------------------------------------------------------------
                   16b  00541 00A0    3312b /dev/vx/rdmp/c4t2d0s2         3312b (S)
                   32b  00541 00A4    3296b /dev/vx/rdmp/c4t2d1s2         3296b (S)
                   16b  00541 00A8    3296b /dev/vx/rdmp/c4t2d2s2         3296b (S)
                   16b  00541 00A8    3376b /dev/vx/rdmp/c4t2d2s2         3376b (S)
                   16b  03122 0009    3312b /dev/vx/rdmp/c5t0d1s2         3312b (V)
                   16b  00541 00A0    3328b /dev/vx/rdmp/c4t2d0s2         3328b (S)
                   16b  00541 00A4    3328b /dev/vx/rdmp/c4t2d1s2         3328b (S)
                   32b  00541 00A4    3424b /dev/vx/rdmp/c4t2d1s2         3424b (S)
                   16b  00541 00A0    3424b /dev/vx/rdmp/c4t2d0s2         3424b (S)
                   16b  00541 00A0    3472b /dev/vx/rdmp/c4t2d0s2         3472b (S)
                   16b  00541 00A8   27984b /dev/vx/rdmp/c4t2d2s2        27984b (S)
                   32b  00541 00A4   27968b /dev/vx/rdmp/c4t2d1s2        27968b (S)
                   16b  00541 00A8   27968b /dev/vx/rdmp/c4t2d2s2        27968b (S)
                   32b  00541 00A4   28384b /dev/vx/rdmp/c4t2d1s2        28384b (S)
                   16b  00541 00A8   28384b /dev/vx/rdmp/c4t2d2s2        28384b (S)
                   16b  00541 00A0   27968b /dev/vx/rdmp/c4t2d0s2        27968b (S)
                   16b  00541 00A8   28496b /dev/vx/rdmp/c4t2d2s2        28496b (S)
                   32b  00541 00A4   28512b /dev/vx/rdmp/c4t2d1s2        28512b (S)
                   32b  00541 00A0   28544b /dev/vx/rdmp/c4t2d0s2        28544b (S)
                   32b  03122 0009   28480b /dev/vx/rdmp/c5t0d1s2        28480b (S)
                   16b  00541 00A8   28512b /dev/vx/rdmp/c4t2d2s2        28512b (S)
                   16b  00541 00A8   28560b /dev/vx/rdmp/c4t2d2s2        28560b (S)
                   32b  03122 0009   28544b /dev/vx/rdmp/c5t0d1s2        28544b (S)
                  256b  00541 00A8   28576b /dev/vx/rdmp/c4t2d2s2        28576b (V)
                  256b  03122 0009   28576b /dev/vx/rdmp/c5t0d1s2        28576b (S)
                  256b  00541 00A0   28608b /dev/vx/rdmp/c4t2d0s2        28608b (S)
                  236b  00541 00A4   28608b /dev/vx/rdmp/c4t2d1s2        28608b (S)
            }

            Mirror Physical Devices (4):
            {
            ----------------------------------------------------------------------

            Array Dev   PPdevName                PdevName
            ----------------------------------------------------------------------
            00541 00A0  /dev/vx/rdmp/c4t2d0s2   /dev/vx/rdmp/c4t2d0s2      (M)
            00541 00A4  /dev/vx/rdmp/c4t2d1s2   /dev/vx/rdmp/c4t2d1s2      (M)
            00541 00A8  /dev/vx/rdmp/c4t2d2s2   /dev/vx/rdmp/c4t2d2s2      (M)
            03122 0009  /dev/vx/rdmp/c5t0d1s2   /dev/vx/rdmp/c5t0d1s2      (V)
            }
          }

Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

The `symrslv file` command with the `-phys_collapse` option physically collapses the same logically collapsed file from 27 extents to 22 extents, but the file can no longer be reconstructed if a subsequent expand operation is performed. For example, the first 32b extent (Dev `00A0`) displayed represents two 16b extents that were not contiguous extents of the file but are contiguous on the disk. Thus, these two extents were not logically collapsible (in the previous command), but they are physically collapsible as long as the file does not need to be reconstructed.

symrslv file /fs_test1/file1 -phys_collapse -blocks

```
    Absolute Path                 : /fs_test1/file1
    Resolved Object Type          : SunOS UFS File
    Resolved Object Size          : 1500b
    Number of Trailing Bytes      : 0
    Extent byte offset to data    : 0
    Number of Physical Devices    : 4
    File System Mount Point       : /fs_test1
    File System Device Name       : /dev/vx/dsk/fsvg/Striped

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration

         Mirror Stripe Columns    : 4
         Mirror Stripe Size       : 32b

         Mirror Physical Extents (22):
         {
-----------------------------------------------------------------------

      Size  Array Dev    Offset PPdevName                   Offset
-----------------------------------------------------------------------
       32b  00541 00A0    3312b /dev/vx/rdmp/c4t2d0s2        3312b (S)
       48b  00541 00A4    3296b /dev/vx/rdmp/c4t2d1s2        3296b (S)
       16b  00541 00A8    3296b /dev/vx/rdmp/c4t2d2s2        3296b (S)
       16b  00541 00A8    3376b /dev/vx/rdmp/c4t2d2s2        3376b (S)
       16b  03122 0009    3312b /dev/vx/rdmp/c5t0d1s2        3312b (S)
       32b  00541 00A4    3424b /dev/vx/rdmp/c4t2d1s2        3424b (S)
       16b  00541 00A0    3424b /dev/vx/rdmp/c4t2d0s2        3424b (S)
       16b  00541 00A0    3472b /dev/vx/rdmp/c4t2d0s2        3472b (S)
       16b  00541 00A8   27984b /dev/vx/rdmp/c4t2d2s2       27984b (S)
       32b  00541 00A4   27968b /dev/vx/rdmp/c4t2d1s2       27968b (S)
       16b  00541 00A8   27968b /dev/vx/rdmp/c4t2d2s2       27968b (S)
       32b  00541 00A4   28384b /dev/vx/rdmp/c4t2d1s2       28384b (S)
       16b  00541 00A8   28384b /dev/vx/rdmp/c4t2d2s2       28384b (S)
       16b  00541 00A0   27968b /dev/vx/rdmp/c4t2d0s2       27968b (S)
       32b  00541 00A8   28496b /dev/vx/rdmp/c4t2d2s2       28496b (S)
       32b  00541 00A4   28512b /dev/vx/rdmp/c4t2d1s2       28512b (S)
       32b  00541 00A0   28544b /dev/vx/rdmp/c4t2d0s2       28544b (V)
       32b  03122 0009   28480b /dev/vx/rdmp/c5t0d1s2       28480b (S)
      272b  00541 00A8   28560b /dev/vx/rdmp/c4t2d2s2       28560b (S)
      288b  03122 0009   28544b /dev/vx/rdmp/c5t0d1s2       28544b (S)
      256b  00541 00A0   28608b /dev/vx/rdmp/c4t2d0s2       28608b (S)
      236b  00541 00A4   28608b /dev/vx/rdmp/c4t2d1s2       28608b (V)
}
```

```
  Mirror Physical Devices (4):
  {
  ----------------------------------------------------------------------

  Array Dev   PPdevName                PdevName
  ----------------------------------------------------------------------
  00541 00A0  /dev/vx/rdmp/c4t2d0s2   /dev/vx/rdmp/c4t2d0s2      (M)
  00541 00A4  /dev/vx/rdmp/c4t2d1s2   /dev/vx/rdmp/c4t2d1s2      (M)
  00541 00A8  /dev/vx/rdmp/c4t2d2s2   /dev/vx/rdmp/c4t2d2s2      (M)
  03122 0009  /dev/vx/rdmp/c5t0d1s2   /dev/vx/rdmp/c5t0d1s2      (V)
  }
}

     Legend for the Attribute of Devices:

     (C): CLARiiON Device.
     (S): Symmetrix Device.
     (M): Symmetrix Device Meta Head.
     (m): Symmetrix Device Meta member.
     (V): Virtual Disk on VMFS.
```

The `symrslv fs` command with the `-no_extents` option displays the attributes of a file system (`/fs_test1`) but omits any extent data.

```
symrslv fs /fs_test1 -no_extents

    Absolute Path                 : /fs_test1
    Resolved Object Type          : File System
    Resolved Object Attributes    : Clustered
    Resolved Object Size          : 4430m
    Number of Trailing Bytes      : 0
    Extent byte offset to data    : 0
    Number of Physical Devices    : 5
    File System Mount Point       : /fs_test1
    File System Device Name       : /dev/vx/dsk/fsvg/Striped

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration

         Mirror Stripe Columns    : 4
         Mirror Stripe Size       : 16k

         Mirror Physical Extents (0):

         Mirror Physical Devices (5):
         {
  ----------------------------------------------------------------------

  Array Dev   PPdevName                PdevName
  ----------------------------------------------------------------------
  00541 00A0  /dev/vx/rdmp/c4t2d0s2   /dev/vx/rdmp/c4t2d0s2      (M)
  00541 00A4  /dev/vx/rdmp/c4t2d1s2   /dev/vx/rdmp/c4t2d1s2      (M)
  00541 00A8  /dev/vx/rdmp/c4t2d2s2   /dev/vx/rdmp/c4t2d2s2      (M)
  03122 0009  /dev/vx/rdmp/c5t0d1s2   /dev/vx/rdmp/c5t0d1s2      (S)
  03122 000A  /dev/vx/rdmp/c5t0d2s2   /dev/vx/rdmp/c5t0d2s2      (S)
  }
}
     Legend for the Attribute of Devices:

     (C): CLARiiON Device.
     (S): Symmetrix Device.
     (M): Symmetrix Device Meta Head.
     (m): Symmetrix Device Meta member.
     (V): Virtual Disk on VMFS.
```

Note: Clusterd file systems are only supported on Solaris and Red Hat (global file systems) clustered environments.

The `symrslv lv` command examines a logical volume named `striped` within a volume group (`-g`) named `fsvg`. The `-pdev_extents` option shows information about pdev-level extents only - without displaying any underlying metadevice configuration. The `-stripe_column` option adds the stripe column number to the display of physical extents, a useful option when dealing with a striped device. Note that stripe column 3 spans two devices (`0009` and `000A`) whose combined size equals the column size of each of the other devices. Column-3 devices are located on a different array (`03122`) from the others.

```
symrslv lv striped -pdev_extents -stripe_column -g fsvg -blocks


    Absolute Path                 : /dev/vx/rdsk/fsvg/Striped
    Resolved Object Type          : SunOS VxVM Logical Volume
    Resolved Object Attributes    : Clustered
    Resolved Object Size          : 9216000b
    Number of Trailing Bytes      : 0
    Extent byte offset to data    : 0
    Number of Physical Devices    : 5

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration

         Mirror Stripe Columns    : 4
         Mirror Stripe Size       : 32b

         Mirror Physical Extents (5):
         {
  ----------------------------------------------------------------------
  Stripe
  Column    Size  Array Dev    Offset PPdevName              Offset
  ----------------------------------------------------------------------
      0 2304000b  00541 00A0    2880b /dev/vx/rdmp/c4t2d0s2   2880b (S)
      1 2304000b  00541 00A4    2880b /dev/vx/rdmp/c4t2d1s2   2880b (S)
      2 2304000b  00541 00A8    2880b /dev/vx/rdmp/c4t2d2s2   2880b (S)
      3 1961280b  03122 0009    2880b /dev/vx/rdmp/c5t0d1s2   2880b (S)
      3  342720b  03122 000A    3840b /dev/vx/rdmp/c5t0d2s2   3840b (S)
  }

  Mirror Physical Devices (5):
  {
  ----------------------------------------------------------------------

  Array Dev   PPdevName               PdevName
  ----------------------------------------------------------------------
  00541 00A0  /dev/vx/rdmp/c4t2d0s2   /dev/vx/rdmp/c4t2d0s2      (M)
  00541 00A4  /dev/vx/rdmp/c4t2d1s2   /dev/vx/rdmp/c4t2d1s2      (M)
  00541 00A8  /dev/vx/rdmp/c4t2d2s2   /dev/vx/rdmp/c4t2d2s2      (M)
  03122 0009  /dev/vx/rdmp/c5t0d1s2   /dev/vx/rdmp/c5t0d1s2      (S)
  03122 000A  /dev/vx/rdmp/c5t0d2s2   /dev/vx/rdmp/c5t0d2s2      (S)
  }
}

    Legend for the Attribute of Devices:

    (C): CLARiiON Device.
    (S): Symmetrix Device.
    (M): Symmetrix Device Meta Head.
    (m): Symmetrix Device Meta member.
    (V): Virtual Disk on VMFS.
```

The `symrslv lv` command examines the same logical volume (named `striped`) as the previous command, but this time displays (by default) the underlying metadevice configuration for each extent. An (`M`) at the end of a row of extent data indicates that this extent is located on a metadevice.

symrslv lv striped -g fsvg -blocks

```
    Absolute Path                 : /dev/vx/rdsk/fsvg/Striped
    Resolved Object Type          : SunOS VxVM Logical Volume
    Resolved Object Attributes    : Clustered
    Resolved Object Size          : 9216000b
    Number of Trailing Bytes      : 0
    Extent byte offset to data    : 0
    Number of Physical Devices    : 5

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration

         Mirror Stripe Columns    : 4
         Mirror Stripe Size       : 32b

         Mirror Physical Extents (5):
         {
  ----------------------------------------------------------------------

        Size  Array Dev    Offset PPdevName                   Offset
  ----------------------------------------------------------------------
    2304000b  00541 00A0    2880b /dev/vx/rdmp/c4t2d0s2        2880b (M)
    2304000b  00541 00A4    2880b /dev/vx/rdmp/c4t2d1s2        2880b (M)
    2304000b  00541 00A8    2880b /dev/vx/rdmp/c4t2d2s2        2880b (M)
    1961280b  03122 0009    2880b /dev/vx/rdmp/c5t0d1s2        2880b (S)
     342720b  03122 000A    3840b /dev/vx/rdmp/c5t0d2s2        3840b (S)
  }

  Mirror Physical Devices (5):
  {
  ----------------------------------------------------------------------

  Array Dev   PPdevName               PdevName
  ----------------------------------------------------------------------
  00541 00A0  /dev/vx/rdmp/c4t2d0s2   /dev/vx/rdmp/c4t2d0s2      (M)
  00541 00A4  /dev/vx/rdmp/c4t2d1s2   /dev/vx/rdmp/c4t2d1s2      (M)
  00541 00A8  /dev/vx/rdmp/c4t2d2s2   /dev/vx/rdmp/c4t2d2s2      (M)
  03122 0009  /dev/vx/rdmp/c5t0d1s2   /dev/vx/rdmp/c5t0d1s2      (S)
  03122 000A  /dev/vx/rdmp/c5t0d2s2   /dev/vx/rdmp/c5t0d2s2      (S)
  }
}
    Legend for the Attribute of Devices:

    (C): CLARiiON Device.
    (S): Symmetrix Device.
    (M): Symmetrix Device Meta Head.
    (m): Symmetrix Device Meta member.
    (V): Virtual Disk on VMFS.
```

The `symrslv pd` command examines extents of one of the physical devices from the previous display. As shown in that display, device `/dev/rdsk/c4t2d0s2` is the metahead (M) of a Symmetrix metadevice that includes other metamembers (m). This display identifies those metamembers as Symmetrix devices `00A1`, `00A2`, and `00A3`.

symrslv pd /dev/rdsk/c4t2d0s2 -blocks

```
    Absolute Path                 : /dev/rdsk/c4t2d0s2
    Resolved Object Type          : Physical Device
    Resolved Object Size          : 35349120b
    Number of Trailing Bytes      : 0
    Extent byte offset to data    : 0
    Number of Physical Devices    : 1

    Number of Mirrors for object (1):
       {
    1) Mirror Configuration

         Mirror Physical Extents (4):
         {
  ----------------------------------------------------------------------

        Size  Array Dev    Offset PPdevName                   Offset
  ----------------------------------------------------------------------
    8837760b  00541 00A0       0b /dev/rdsk/c4t2d0s2              0b (M)
    8837760b  00541 00A1       0b /dev/rdsk/c4t2d0s2        8837760b (m)
    8837760b  00541 00A2       0b /dev/rdsk/c4t2d0s2       17675520b (m)
    8835840b  00541 00A3       0b /dev/rdsk/c4t2d0s2       26513280b (m)
  }

  Mirror Physical Devices (4):
  {
  ----------------------------------------------------------------------

  Array Dev   PPdevName               PdevName
  ----------------------------------------------------------------------
  00541 00A0  /dev/rdsk/c4t2d0s2      /dev/rdsk/c4t2d0s2         (M)
  00541 00A1  /dev/rdsk/c4t2d0s2      /dev/rdsk/c4t2d0s2         (m)
  00541 00A2  /dev/rdsk/c4t2d0s2      /dev/rdsk/c4t2d0s2         (m)
  00541 00A3  /dev/rdsk/c4t2d0s2      /dev/rdsk/c4t2d0s2         (m)
  }
}

    Legend for the Attribute of Devices:

    (C): CLARiiON Device.
    (S): Symmetrix Device.
    (M): Symmetrix Device Meta Head.
    (m): Symmetrix Device Meta member.
    (V): Virtual Disk on VMFS.
```

The `symrslv file` command is used to display the extents of a Celerra HighRoad MPFS file.

symrslv file /mpfstest/system

```
    Absolute Path                 : /mpfstest/system
    Resolved Object Type          : Celerra HighRoad MPFS File
    Resolved Object Size          : 2k
    Number of Trailing Bytes      : 257
    Extent byte offset to data    : 0
    Number of Physical Devices    : 1
    File System Mount Point       : /mpfstest
    File System Device Name       : api242:/nmk_fs
```

```
            Number of Mirrors for object (1):
            {
            1) Mirror Configuration

            Mirror Physical Extents (1):
            {
-----------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                   Offset
-----------------------------------------------------------------------
8k  00229 03A3    7881m /dev/rdsk/c3t1d3s2           7881m (S)
}

Mirror Physical Devices (1):
{
-----------------------------------------------------------------------

Array Dev   PPdevName               PdevName
-----------------------------------------------------------------------
00229 03A3  /dev/rdsk/c3t1d3s2      /dev/rdsk/c3t1d3s2             (S)
}
}

            Legend for the Attribute of Devices:

            (C): CLARiiON Device.
            (S): Symmetrix Device.
            (M): Symmetrix Device Meta Head.
            (m): Symmetrix Device Meta member.
            (V): Virtual Disk on VMFS.
```

Example 3: Displaying volume groups and logical volumes

The hardware setup for this example consists of a Sun Solaris host connected to a Symmetrix array. The SunOS VxVM logical volume manager manages the volume groups and logical volumes.

The `symvg list` command displays logical volume groups defined for this host. The N/A columns are not applicable to a SunOS VxVM logical volume manager (although HP-UX and AIX LVMs use them).

symvg list

```
VOLUME GROUPS (SunOS VxVM):

                                          PE    Max     Max     Num     Num
Name            State    Attribute      Size  Devices Volumes Devices Volumes

rootdg          Enabled  N/A             N/A      N/A     N/A       1       0
bcv_fs_lvm_sin* Enabled  N/A             N/A      N/A     N/A       1       1
fsvg            Enabled  N/A             N/A      N/A     N/A       6       2
nasty           Enabled  N/A             N/A      N/A     N/A       4       7
testvg          Enabled  N/A             N/A      N/A     N/A       3       1
```

The `symvg show` command displays information on the volume group named `nasty`, including the set of four physical devices that the LVM allocated for this volume group. The display includes each device's physical device name, Symmetrix device name, status (RW), and capacity in megabytes. The physical devices in this display are on Symmetrix array `00541`, and three of the devices are metadevices(M).

symvg show nasty

```
    Volume Group Name : nasty
    Volume Group Type : SunOS VxVM

     Volume Group State             : Enabled

     Volume Group Attributes        : Clustered | Read Only

     Group's Physical Extent Size   : N/A

     Max Number of Devices in Group : N/A
     Max Number of Volumes in Group : N/A

     Number of Devices in Group     :   4
     Number of Volumes in Group     :   7

     Physical Device Members (4):
         {
         -------------------------------------------------------
                                          Sym                Cap
         PdevName                   SymID  Dev  Att. Sts     (MB)
         -------------------------------------------------------
         /dev/vx/rdmp/c4t1d6s2      00541  009B      RW      4315
         /dev/vx/rdmp/c4t1d7s2      00541  009C (M)  RW     17261
         /dev/vx/rdmp/c4t2d3s2      00541  00AC (M)  RW     17261
         /dev/vx/rdmp/c4t2d4s2      00541  00B0 (M)  RW     17258
         }
```

Note: Clustered volume groups are only supported in Solaris and Red Hat clustered environments.

The `symlv list` command lists the logical volumes of the volume group (`nasty`) and shows each volume's configuration at the operating system level, state, condition, number of mirrors, and number of logical extents.

```
symlv -g nasty list

Volume Group Name: nasty
Volume Group Type: SunOS VxVM

                                                    Num      Num Log
         Name        Configuration   State     Cond    Mirrors  Extents
-------------------- -------------  -------  --------  -------  -------

MirrorStripe         Striped Mir    Enabled  Sync            2      N/A
RAID5                RAID5          Enabled  Sync            1      N/A
Simple               Simple         Enabled  Sync            1      N/A
StripeMirror         Striped Mir    Enabled  Sync            2      N/A
Striped              Striped        Enabled  Sync            1      N/A
Wide                 Striped        Enabled  Sync            1      N/A
Wide2                Striped        Enabled  Sync            1      N/A
```

The `symlv show` command examines a logical volume named Simple, displaying its mirror extent size in blocks. SRM locates the physical extents on the physical devices where the logical volume resides and displays the mirror configuration at the operating system level. In this case, an LVM on the host configured this logical volume as one mirror, meaning that data blocks for this volume are written to one place. The LVM also configured the volume as a "simple" mirror.

```
symlv -g nasty -blocks show Simple

  Logical Volume Name            : Simple
  Logical Volume Pathname        : /dev/vx/rdsk/nasty/Simple
  Volume State                   : Enabled
  Volume Configuration           : Simple
  Volume Condition               : Sync
  Volume Allocation State        : N/A
  Volume Attributes              : Clustered | Read Only

  Logical Volume Size            : 204800b

  Number of Logical Volume Mirrors (1):
     {
  1) Mirror Configuration        : Simple
     Mirror State                : Enabled
     Mirror Flags                : None
     Mirror Condition            : Sync

     Number of device Partitions : 1
     Number of Physical devices  : 1
     Number of Storage devices   : 1
```

Note: Clustered volumes are only supported in Solaris and Red Hat clustered environments.

The display shows one 205440-block extent on Symmetrix device `009B` (partitioned physical device name `/dev/vx/rdmp/c4t1d6s2`). The condition of the extent is Sync (all physical extents in all mirrors of the logical volume are synchronized). Other possible extent conditions are listed in Table on page 139.

```
Mirror Physical Extents (1):
{
-----------------------------------------------------------------------

Condition     Size  Array Dev    Offset PPdevName                 Offset
-----------------------------------------------------------------------
Sync       205440b  00541 009B 7331520b /dev/vx/rdmp/c4t1d6s2   7331520b
}

Mirror Physical Devices (1):
{
-----------------------------------------------------------------------

Array Dev   PPdevName                PdevName
-----------------------------------------------------------------------
00541 009B  /dev/vx/rdmp/c4t1d6s2   /dev/vx/rdmp/c4t1d6s2            (S)
}
}Legend for the Attribute of Devices:...
```

Table 22 Extent condition descriptions

Extent condition	Description
Stale	The logical volume contains a mirror with some physical extents that are not consistent with corresponding extents in another mirror.
Sync	All physical extents in all mirrors of the logical volume are synchronized.
Degraded RAID 5	For VxVM, DiskADM, and LDM only. One of the devices in the RAID 5 set is unavailable. Inaccessible data is being generated using the parity check formula.
Degraded Volume	A mirror is off line.
Empty	There is no data on the volume.
Unusable RAID 5	For VxVM, DiskADM, and LDM only. The RAID 5 volume is not usable. This condition implies that access to two subdisks has failed.
Syncing	A member of the volume is being regenerated.
NeedSync	A mirror is stale.
NoDev	A physical device is unavailable.
Degraded	In a simple or striped logical volume, there is at least one good copy of data available.
Initializing	The logical volume is initializing.

This `symlv show` command displays a logical volume named `MirrorStripe`, which the LVM has configured as striped and mirrored (Striped Mir) with two logical volume mirrors.

```
symlv -g nasty -blocks -collapse show MirrorStripe

     Logical Volume Name            : MirrorStripe
     Logical Volume Pathname        : /dev/vx/rdsk/nasty/MirrorStripe
     Volume State                   : Enabled
```

```
        Volume Configuration           : Striped Mir
        Volume Condition               : Sync
        Volume Allocation State        : N/A
        Volume Attributes              : DRL enabled

        Logical Volume Size            : 204800b

        Number of Logical Volume Mirrors (2):
```

This part of the display describes one (1) of the logical volume mirrors. This mirror maps to two physical devices (`/dev/vx/rdmp/c4t1d6s2` and `/dev/vx/rdmp/c4t1d7s2`), which are configured at the storage level to two storage devices (Symmetrix devices `009B` and `009D`). The extent marked by an (m) is located on a device that is a member of a Symmetrix metadevice.

```
        {
      1) Mirror Configuration        : Striped
         Mirror State                : Enabled
         Mirror Flags                : None
         Mirror Condition            : Sync
         Number of Striped Devices   : 2
         Stripe Size                 : 128b

         Number of device Partitions : 2
         Number of Physical devices  : 2
         Number of Storage devices   : 2

         Mirror Physical Extents (2):
  {
  ---------------------------------------------------------------------------

  Condition     Size  Array Dev    Offset PPdevName                 Offset
  ---------------------------------------------------------------------------
  Sync       102720b  00541 009B 7228800b /dev/vx/rdmp/c4t1d6s2   7228800b(m)
  Sync       102720b  00541 009D 2992320b /dev/vx/rdmp/c4t1d7s2   11830080b
  }

Mirror Physical Devices (2):
  {
  ---------------------------------------------------------------------------

  Array Dev   PPdevName               PdevName
  ---------------------------------------------------------------------------
  00541 009B  /dev/vx/rdmp/c4t1d6s2   /dev/vx/rdmp/c4t1d6s2            (S)
  00541 009D  /dev/vx/rdmp/c4t1d7s2   /dev/vx/rdmp/c4t1d7s2            (m)
  }
```

This part of the display describes the second (2) mirror. This mirror maps to two physical devices (`/dev/vx/rdmp/c4t2d3s2` and `/dev/vx/rdmp/c4t2d4s2`) that are configured at the storage level as Symmetrix metadevices, each with four metamembers. Physical extents exist on all eight of the storage devices.

```
      2) Mirror Configuration        : Striped
         Mirror State                : Enabled
         Mirror Flags                : Striped META
         Mirror Condition            : Sync
         Number of Striped Devices   : 2
         Stripe Size                 : 128b

         Number of device Partitions : 2
         Number of Physical devices  : 2
         Number of Storage devices   : 8

         Mirror Physical Extents (10):
```

```
            {
  ---------------------------------------------------------------------

  Condition     Size  Array Dev    Offset PPdevName                 Offset
  ---------------------------------------------------------------------
  Sync          960b  00541 00AD 2957760b /dev/vx/rdmp/c4t2d3s2   11830080b(m)
  Sync          960b  00541 00B0 2959680b /dev/vx/rdmp/c4t2d4s2   11830080b(M)
  Sync        26880b  00541 00AE 2956800b /dev/vx/rdmp/c4t2d3s2   11831040b(m)
  Sync        26880b  00541 00B1 2956800b /dev/vx/rdmp/c4t2d4s2   11831040b(m)
  Sync        24960b  00541 00AF 2956800b /dev/vx/rdmp/c4t2d3s2   11832960b(m)
  Sync        26880b  00541 00B2 2956800b /dev/vx/rdmp/c4t2d4s2   11834880b(m)
  Sync        24960b  00541 00AC 2958720b /dev/vx/rdmp/c4t2d3s2   11834880b(M)
  Sync        24960b  00541 00B3 2956800b /dev/vx/rdmp/c4t2d4s2   11838720b(m)
  Sync        24960b  00541 00AD 2958720b /dev/vx/rdmp/c4t2d3s2   11836800b(m)
  Sync        23040b  00541 00B0 2960640b /dev/vx/rdmp/c4t2d4s2   11842560b(M)
  }

Mirror Physical Devices (8):
  {
  ---------------------------------------------------------------------

  Array Dev   PPdevName               PdevName
  ---------------------------------------------------------------------
  00541 00AC  /dev/vx/rdmp/c4t2d3s2   /dev/vx/rdmp/c4t2d3s2            (M)
  00541 00AD  /dev/vx/rdmp/c4t2d3s2   /dev/vx/rdmp/c4t2d3s2            (m)
  00541 00AE  /dev/vx/rdmp/c4t2d3s2   /dev/vx/rdmp/c4t2d3s2            (m)
  00541 00AF  /dev/vx/rdmp/c4t2d3s2   /dev/vx/rdmp/c4t2d3s2            (m)
  00541 00B0  /dev/vx/rdmp/c4t2d4s2   /dev/vx/rdmp/c4t2d4s2            (M)
  00541 00B1  /dev/vx/rdmp/c4t2d4s2   /dev/vx/rdmp/c4t2d4s2            (m)
  00541 00B2  /dev/vx/rdmp/c4t2d4s2   /dev/vx/rdmp/c4t2d4s2            (m)
  00541 00B3  /dev/vx/rdmp/c4t2d4s2   /dev/vx/rdmp/c4t2d4s2            (m)...
```

Example 4: Deporting and importing a volume group

Volume group metadata can be deported from a system to persistent storage and imported later to another host (for example, a backup server). The hardware setup for this example consists of a Sun Solaris host connected to a Symmetrix array. The SunOS VxVM logical volume manager manages the volume groups and logical volumes.

The `symvg show` command displays information on the volume group named `testvg`, including the set of three physical devices that the LVM allocated for this volume group. The display includes each device's physical device name, Symmetrix device name, status (RW), and capacity in megabytes. The physical devices in this display are on Symmetrix (S) array `03122`.

```
symvg show testvg

       Volume Group Name : testvg
       Volume Group Type : SunOS VxVM

       Volume Group State             : Enabled

       Volume Group Attributes        : N/A

       Group's Physical Extent Size   : N/A

       Max Number of Devices in Group : N/A
       Max Number of Volumes in Group : N/A

       Number of Devices in Group     :   3
       Number of Volumes in Group     :   1

       Physical Device Members (3):
           {
           -------------------------------------------------------
                                                                Cap
           PdevName                  Array  Dev  Att. Sts      (MB)
           -------------------------------------------------------
           /dev/vx/rdmp/c5t1d0s2     03122  0028 (S)  RW        960
           /dev/vx/rdmp/c5t1d2s2     03122  002A (S)  RW        960
           /dev/vx/rdmp/c5t1d7s2     03122  002F (S)  RW        960
           }

   Legend for the Attribute of Devices:

   (C): CLARiiON Device.
   (S): Symmetrix Device.
   (M): Symmetrix Device Meta Head.
   (m): Symmetrix Device Meta member.
   (V): Virtual Disk on VMFS.
```

The `symvg deport` command deports the metadata of volume group `testvg` from the system and stores that information in persistent storage. Some platforms may require `-mapfile` option and a file name for a deport operation, which will also be needed for a subsequent import operation.

```
symvg deport testvg

The Control Operation Completed Successfully
```

Another `symvg show` command verifies that volume group `testvg` is no longer available in the system.

```
symvg show testvg

Either no volume groups or the specified volume group does not exist
```

The `symvg import` command imports volume group `testvg` into the system from where the command is issued.

```
symvg import testvg

The Control Operation Completed Successfully
```

Another `symvg show` command verifies that the volume group was successfully imported and that information about the volume group is available again.

```
symvg show testvg

    Volume Group Name : testvg

    Volume Group Type : SunOS VxVM

    Volume Group State             : Enabled

    Volume Group Attributes        : N/A

    Group's Physical Extent Size   : N/A

    Max Number of Devices in Group : N/A
    Max Number of Volumes in Group : N/A

    Number of Devices in Group     :   3
    Number of Volumes in Group     :   1

    Physical Device Members (3):
        {
        -------------------------------------------------------
                                                            Cap
        PdevName                  Array  Dev  Att. Sts     (MB)
        -------------------------------------------------------
        /dev/vx/rdmp/c5t1d0s2     03122  0028 (S)  RW       960
        /dev/vx/rdmp/c5t1d2s2     03122  002A (S)  RW       960
        /dev/vx/rdmp/c5t1d7s2     03122  002F (S)  RW       960
        }

Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

Example 5: Mapping files

CLARiiON storage devices

The hardware setup for this example consists of a Windows 2000 host connected to a CLARiiON array. The files displayed are within a Windows NTFS file system.

The `symrslv file` command expands the extents of the file named `clartest` and displays all 3201 of its physical extents. The ellipsis (...) indicates where output was omitted for brevity. This command focuses on the extent data at the CLARiiON device layer. The `symrslv` command drills down directly to the CLARiiON devices (Dev) on which the file is stored and locates the extents there.

```
symrslv  file p:\clartest -expand

   Absolute Path                 : P:\CLARTEST
   Resolved Object Type          : Windows NTFS File
   Resolved Object Size          : 200m
   Number of Trailing Bytes      : 0
   Extent byte offset to data    : 0
   Number of Physical Devices    : 2
   File System Mount Point       : P:\
   File System Device Name       : \\.\P:

   Number of Mirrors for object (1):
   {
   1) Mirror Configuration

   Mirror Stripe Columns    : 2
   Mirror Stripe Size       : 64k

   Mirror Physical Extents (3201):
   {

   ------------------------------------------------------------------

   Size  Array Dev    Offset PPdevName                   Offset
   ------------------------------------------------------------------
   60k  00316 0018     127m \\.\PHYSICALDRIVE24           127m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)
   64k  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
   64k  00316 0018     128m \\.\PHYSICALDRIVE24           128m (C)

   ..................................................................

   64k  00316 0018     227m \\.\PHYSICALDRIVE24           227m (C)
   64k  00316 0017     227m \\.\PHYSICALDRIVE23           227m (C)
   64k  00316 0018     227m \\.\PHYSICALDRIVE24           227m (C)
```

```
64k  00316 0017     227m \\.\PHYSICALDRIVE23           227m (C)
64k  00316 0018     227m \\.\PHYSICALDRIVE24           227m (C)
64k  00316 0017     227m \\.\PHYSICALDRIVE23           227m (C)
64k  00316 0018     227m \\.\PHYSICALDRIVE24           227m (C)
}

Mirror Physical Devices (2):
{
-------------------------------------------------------------------

Array Dev   PPdevName               PdevName
-------------------------------------------------------------------
00316 0018  \\.\PHYSICALDRIVE24    \\.\PHYSICALDRIVE24         (C)
00316 0017  \\.\PHYSICALDRIVE23    \\.\PHYSICALDRIVE23         (C)
}
}

Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

The `symrslv file` command with the `-collapse` option logically collapses the same file's extent data, which means that the file can be reconstructed if a subsequent expand operation is performed. Extents whose size is now 100 megabytes were formed from collapsing contiguous 64-kilobyte extents shown in the previous expanded display for the file `clartest`.

symrslv file p:\clartest -collapse

```
Absolute Path                 : P:\CLARTEST
Resolved Object Type          : Windows NTFS File
Resolved Object Size          : 200m
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 2
File System Mount Point       : P:\
File System Device Name       : \\.\P:

Number of Mirrors for object (1):
{
1) Mirror Configuration

Mirror Stripe Columns    : 2
Mirror Stripe Size       : 64k

Mirror Physical Extents (3):
{
-------------------------------------------------------------------

Size  Array Dev    Offset PPdevName                  Offset
-------------------------------------------------------------------
 60k  00316 0018     127m \\.\PHYSICALDRIVE24           127m (C)
100m  00316 0017     128m \\.\PHYSICALDRIVE23           128m (C)
100m  00316 0018     128m \\.\PHYSICALDRIVE24           128m (V)
         }

Mirror Physical Devices (2):
{

-------------------------------------------------------------------
```

```
Array Dev   PPdevName                PdevName
-------------------------------------------------------------------
00316 0018  \\.\PHYSICALDRIVE24    \\.\PHYSICALDRIVE24         (C)
00316 0017  \\.\PHYSICALDRIVE23    \\.\PHYSICALDRIVE23         (C)
}
}

Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

Example 6: Displaying a logical volume

The hardware setup for this example consists of a Windows 2000 host connected to a CLARiiON storage array.

The `symlv show` command examines a logical volume named `P:` in the volume group `clardg`, displaying the volume's mirror extent size as 2039 megabytes. SRM locates the physical extents on the physical devices where the logical volume resides and displays the mirror configuration at the operating system level. An LVM on the host configured this logical volume as one "striped" mirror, with data blocks for this volume written to two devices. The display shows two 1020-megabyte extents on CLARiiON devices `0017` and `0018`. The extent condition is Sync (see Table on page 139 for other possible extent conditions).

`symlv -g clardg show P:`

```
  Logical Volume Name            : P:
  Logical Volume Pathname        : \\.\P:
  Volume State                   : Enabled
  Volume Configuration           : Striped
  Volume Condition               : Sync
  Volume Allocation State        : N/A
  Volume Attributes              : N/A

  Logical Volume Size            : 2039m

  Number of Logical Volume Mirrors (1):
     {
  1) Mirror Configuration        : Striped
     Mirror State                : Enabled
     Mirror Flags                : None
     Mirror Condition            : Sync

     Number of Striped Devices   : 2
     Stripe Size                 : 64k

     Number of device Partitions : 2
     Number of Physical devices  : 2
     Number of Storage devices   : 2

     Mirror Physical Extents (2):
       {
  ---------------------------------------------------------------------

  Condition     Size  Array Dev    Offset PPdevName                Offset
  ---------------------------------------------------------------------
(C)  Sync         1020m  00316 0017      31k \\.\PHYSICALDRIVE23          31k
(C)  Sync         1020m  00316 0018      31k \\.\PHYSICALDRIVE24          31k
     }

Mirror Physical Devices (2):
  {
  ---------------------------------------------------------------------

  Array Dev   PPdevName               PdevName
  ---------------------------------------------------------------------
  00316 0017  \\.\PHYSICALDRIVE23     \\.\PHYSICALDRIVE23              (C)
  00316 0018  \\.\PHYSICALDRIVE24     \\.\PHYSICALDRIVE24              (C)
  }
}
```

```
Legend for the Attribute of Devices:

(C): CLARiiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```