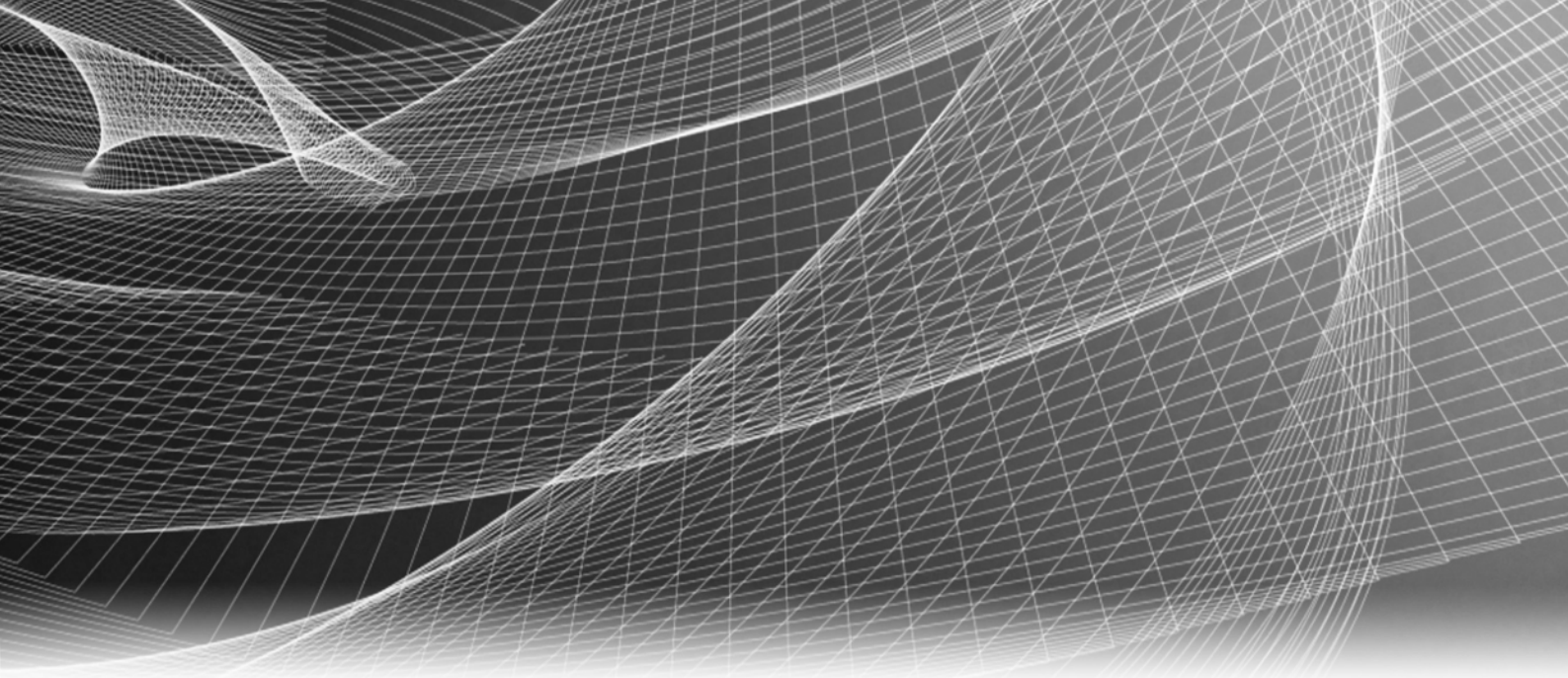




EMC[®] Solutions Enabler SRM

Version 8.1

CLI User Guide

REV 01

Copyright © 2002-2015 EMC Corporation. All rights reserved. Published in the USA.

Published September 2015

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.

EMC², 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

Summary of SRM commands.....	18
Data object commands	18
Database commands	19
Host file system command	20
Logical volume commands.....	20
Performance statistics commands.....	21
Daemon service	22
Daemon authorization.....	22
Defining the SRM operations available to users.....	23
Setting operating system level permissions.....	24

Chapter 2 Data Objects

Conventions.....	26
Data object terminology and architecture	27
Partitioned physical devices and partitions	27
Data objects.....	27
Extents.....	27
Data blocks.....	28
Examining device partitions	28
Options.....	28
Examples	29
Examining device data objects	31
Options.....	31
Example.....	32
Examining logical volume data objects.....	33
Options.....	33
Example.....	33
Examining file system data objects	35
Options.....	35
Examples	36

Chapter 3 Databases

Conventions.....	40
Database architecture and terminology	41
Oracle architecture.....	42
SQL Server architecture.....	45
Sybase architecture	47
IBM Informix architecture	49
IBM DB2 architecture	51
Microsoft Exchange Server architecture	53
SharePoint Portal Server.....	54
Generic SafeWrite	54

Establishing the database connection.....	55
Setting UNIX environment variables	55
Connecting to the database	55
Database startup options.....	56
Database shutdown options	58
Using database daemons.....	60
What are daemons?	60
Why use daemons?	60
Running database daemons.....	61
Listing database instances.....	62
Examining database files	63
Examining database file attributes	63
Listing tablespace files.....	65
Translating database devices to VMAX groups	65
Database devices to device groups	65
Database devices to composite groups	66
Examining tablespaces	67
Examining tablespace attributes	67
Listing tablespace files.....	69
Listing tablespace tables	69
Listing segments in a tablespace	70
Translating tablespace devices to VMAX groups	71
Tablespace devices to device groups	71
Tablespace devices to composite groups	72
Examining schemas	73
Listing schema files	73
Listing schema tables	73
Examining schema attributes	74
Listing segments in a schema	75
Examining tables	75
Examining table attributes	76
Examining segments.....	76
Examining segment attributes	77
Invoking database I/O control	78
Freezing a database	78
Thawing a database	78
Hot backup control.....	78
Checkpoint.....	79
Archiving a log	79
Taking a snapshot of a SQL Server database	79
Using the EMC Oracle ASM library	80
The ASM syntax for disk discovery.....	80
Usage of CLI asmdscvr	81

Chapter 4

File Systems

Conventions.....	84
Overview.....	85
Operating systems and file systems	85
Terminology	85
Finding file systems	86
Options.....	86
Examples	86
Examining a file system.....	87
Options.....	87
Example.....	87

Finding directories and files	88
Options	88
Examples	88
Examining directories.....	89
Listing files	89
Options	89
Examples	90
Examining files	90
Options	91
Example.....	91

Chapter 5

Logical Volumes

Conventions.....	94
Overview.....	95
LVM terminology	95
Mirror configurations.....	97
Logical volume configurations.....	100
Mirror conditions.....	101
Viewing volume groups	102
Listing the volume groups	102
Volume group details	102
Volume group control operations	105
Creating a volume group	105
Adding and removing devices	105
Destroying a volume group.....	105
Recovering a volume group	105
Importing and deporting operations	106
Control options	106
Translating volume groups to VMAX groups	107
Options	107
Examples	107
Viewing logical volumes.....	109
Listing logical volumes.....	109
Logical volume details.....	109
Condition, state, and attributes of a logical volume.....	111
Control operations for logical volumes	112
Creating a logical volume	112
Deleting a logical volume	113
Adding and removing mirrors	113
Extending and reducing logical volumes.....	113
Viewing extents.....	114
Extent conditions	115
Expanded list	115
No extents in list	116

Chapter 6

Statistics

Overview.....	118
Monitoring data over time	118
symhost stats	119
CPU example.....	119
Memory example.....	120
Disk example	120
symlv stats.....	121

symrdb stats	122
Oracle	122
SQL Server	123
Sybase	124
IBM DB2	125
Example	125

Part 2 Operational Examples

Chapter 7 SRM Examples

Example 1: Displaying relational database objects	130
Examining Oracle database objects	130
Examining IBM DB2 database objects	134
Examining IBM Informix database objects	135
Examining Microsoft Exchange database objects	137
Example 2: Mapping files and other disk storage objects	138
symhosts list example	138
symhosts show example	138
symrslv dir example	141
symrslv file example 1	142
symrslv file example 2	143
symrslv file example 3	145
symrslv fs example	146
symrslv lv example 1	147
symrslv lv example 2	148
symrslv pd example	149
Example 3: Displaying volume groups and logical volumes	150
symvg list example	150
symvg show example	150
symlv list example	151
symlv show example 1	151
symlv show example 2	152
Example 4: Deporting and importing a volume group	155
symvg show example	155
deport and import example	156
Example 5: Mapping files	157
symrslv file example 1	157
symrslv file example 2	158
Example 6: Displaying a logical volume	160

FIGURES

	Title	Page
1	Objects, extents, and data blocks	27
2	Oracle database architecture	42
3	Segments, extents, and data blocks	44
4	SQL Server database structure	45
5	Sybase database architecture	47
6	Informix database architecture	49
7	IBM DB2 database architecture	51
8	Microsoft Exchange database architecture	53
9	System view of the daemon process	60
10	Database daemon libraries for communication	62
11	Volume group	96
12	Simple mirror configuration	97
13	Concatenated mirror configuration	98
14	Striped mirror configuration	98
15	RAID 5 Veritas and Windows LDM configuration	99

TABLES

	Title	Page
1	Data object mapping 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	Conventions for parameters to SYMCLI commands for data objects	26
8	Options for the sympart command	28
9	Options for the symrslv pd command	31
10	Options for the symrslv lv command	33
11	Options for the symrslv command for data objects	35
12	Conventions for parameters to SYMCLI commands for databases	40
13	Oracle database terminology	43
14	SQL Server database terminology	46
15	Sybase database terminology	47
16	IBM Informix database terminology	49
17	IBM DB2 database terminology	52
18	UNIX database environment variables	55
19	Database daemon names	62
20	Options for the symrdb show command for database files	63
21	Device options for the symrdb rdb2dg command	65
22	Options for the symrdb show command for tablespaces	67
23	Device options for the symrdb tbs2dg command	71
24	Device options for the symrdb tbs2cg command	72
25	Options for the symrdb show command for schema attributes	74
26	Options for the symrdb show command for table attributes	76
27	Conventions for parameters to SYMCLI commands for file systems	84
28	File system terminology	85
29	Options for the symhostfs show command	87
30	Options for the symhostfs show command for a file	91
31	Conventions for parameters to SYMCLI commands for logical volumes	94
32	Logical volume configuration types	100
33	Logical volume mirror configurations	100
34	Mirror condition descriptions	101
35	Volume group types	103
36	Volume group control options	106
37	Logical volume conditions	111
38	Options for logical volume control operations	112
39	Logical volume extent condition	115
40	SRM statistics commands	118
41	Command line options for monitoring a resource over time	118
42	Data returned by the syhost stats command	119
43	Volume group types	121
44	Metric options for Oracle database types	122
45	Metric options for SQL Server databases	123
46	Metric options for Sybase databases	124
47	Metric options for IBM DB2 databases	125
48	Environment variables to access a Oracle database	130
49	Environment variables to access an IBM DB2 database	134
50	Environment variables to access a Oracle database	135

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 about Solutions Enabler:

- ◆ *EMC Solutions Enabler Release Notes* — Detail new features and any known limitations.
- ◆ *EMC Solutions Enabler Installation Guide* — Provides host-specific installation instructions.
- ◆ *EMC Solutions Enabler CLI Command Reference* — Documents the SYMCLI commands, daemons, error codes and option file parameters provided with the Solutions Enabler man pages.
- ◆ *EMC Solutions Enabler Array Management CLI User Guide* — Describes how to configure array control, management, and migration operations using SYMCLI commands.
- ◆ *EMC Solutions Enabler SRDF Family CLI User Guide* — Describes how to configure and manage SRDF environments using SYMCLI commands.
- ◆ *EMC Solutions Enabler TimeFinder Family CLI User Guide* — Describes how to configure and manage TimeFinder environments using SYMCLI commands.
- ◆ *EMC VMAX Family Security Configuration Guide* — Describes how to configure VMAX Family security settings.

The following provide additional information:

- ◆ *EMC VMAX3™ Family Documentation Set* — Contains documentation related to the VMAX 100K, 200K, and 400K arrays.
- ◆ *EMC VMAX Family Documentation Set* — Contains documentation related to the VMAX 10K, 20K, and 40K arrays.
- ◆ *EMC VMAX3 Family with HYPERMAX OS Release Notes* — Detail new features and any known limitations.
- ◆ *EMC VMAX Family Viewer for Desktop and iPad®* — Illustrates system hardware system configurations offered for VMAX and VMAX3 arrays.
- ◆ *E-Lab™ Interoperability Navigator (ELN)* — Provides a web-based interoperability and solution search portal. You can find the ELN at <https://elabnavigator.EMC.com>.
- ◆ *Solve Desktop* — Provides procedures for common tasks and supported SRDF features. To download the Solve desktop tool go to EMC Online Support at <https://support.EMC.com> and search for Solve Desktop. Download the Desktop and load the VMAX Family and DMX procedure generator.

Conventions used in this document

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
<i>Italic</i>	Used in all text (including procedures) for: - Full titles of publications referenced in text - Emphasis, for example, a new term - Variables
Monospace	Used for: - System output, such as an error message or script - URLs, complete paths, filenames, prompts, and syntax when shown outside of running text
Monospace bold	Used for specific user input, such as commands
<i>Monospace italic</i>	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.

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Send your opinions of this document to:

VMAXContentFeedback@emc.com

PART 1

Concepts and Procedures

This guide has into two parts: Part 1 provides concepts and procedures of the Storage Resource Management (SRM) facility 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 tasks of mapping and examining various objects on a host system, relational databases, and the attached storage arrays.

[Chapter 2, “Data Objects”](#)

Describes the host system SRM. It shows how to examine the extents of datafiles and objects using SYMCLI commands.

[Chapter 3, “Databases”](#)

Describes the host system database SRM for various relational database systems, and shows how to examine the databases and their structure using SYMCLI commands.

[Chapter 4, “File Systems”](#)

Describes the host file system SRM and shows how to examine the file systems and their structure using SYMCLI commands.

[Chapter 5, “Logical Volumes”](#)

Describes the host system logical volume SRM and shows how to examine logical volumes and their groups using SYMCLI commands.

[Chapter 6, “Statistics”](#)

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

CHAPTER 1

SRM Overview

This chapter provides an overview of the Solutions Enabler SRM component focusing on the tasks of managing various storage objects on the host system.

- ◆ [Summary of SRM commands.....](#) 18
- ◆ [Daemon service](#) 22

Summary of SRM commands

The SRM component extends the command set provided by the Symmetrix® Command Line Interface (SYMCLI) to include commands relating to storage resources. These commands display and manipulate the attributes of various objects in a host system, including relational databases and EMC storage arrays.

There are five categories of SYMCLI commands for SRM:

- ◆ Data objects
- ◆ Databases
- ◆ File systems
- ◆ Logical volumes and volume groups
- ◆ Performance statistics

Note: For a list of the platforms and versions for Solutions Enabler, refer to the E-Lab™ Interoperability Navigator, available at <http://elabnavigator.EMC.com>.

Support for the HP-UX platform will be deprecated in a future release of SRM.

Data object commands

The data object commands display the mapping of storage devices and the characteristics of datafiles and objects. [Table 1](#) summarizes the SYMCLI commands for data objects.

Table 1 Data object mapping commands

Command	Argument	Description
symrslv	pd	Display logical-to-physical mapping information for any physical device.
	lv	Display logical-to-physical mapping information for a logical volume.
	file	Display logical-to-physical mapping information for a file.
	dir	Display logical-to-physical mapping information for a directory.
	fs	Display logical-to-physical mapping information for a file system.
	object	Display information about a generic object. In this case, and object is a directory, filename, file system mount point, logical volume name, or a physical device name.
	identity	Display the identity of an object without resolving it.
sympart	show	Display detailed information about a partition for a specified host device.
	list	Display partition information for all the devices connected to a host.

[Chapter 2 “Data Objects”](#) shows how to use the data object commands. For more information about the `symrslv` and `sympart` commands, refer to the *EMC Solutions Enabler CLI Command Reference*.

Database commands

The database object commands list the attributes that describe the structures, files, tablespaces, and user schemas of a database. [Table 2](#) summarizes the SYMCLI commands for databases.

Table 2 Database object commands

Command	Argument	Description
symrdb	list	Display various physical and logical database objects, including: - The available instances of relational databases - Tablespaces, tables, files, or schemas of a database - Files, segments, or tables of a database tablespace or schema
	show	Display 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	Display performance statistics for databases.
	rdb2dg rdb2cg	Translate the devices of a specified database into a device group or a composite group.
	tbs2dg tbs2cg	Translate the devices of a specified tablespace into a device group or a composite group.
	startup	Provide startup options for a database manager instance.
	shutdown	Provide shutdown options for a database manager instance.
symioctl	freeze	Freeze all I/O access to specified databases.
	thaw	Resume I/O access to specified databases.
	begin backup	Place objects into hot backup mode.
	end backup	Take objects out of hot backup mode.
	checkpoint	Issue a checkpoint on a database.
	archive log	Archive the current log.
	begin snapshot	Start a snapshot backup on SQL Server.
	end snapshot	Stop snapshot metadata and resumes writes on SQL Server.
	restore snapshot	Restore previously saved snapshot metadata on SQL Server.
	abort snapshot	Terminate the snapshot without saving metadata and resumes writes on SQL Server.

[Chapter 3 “Databases”](#) shows how to use the database commands. For more information about the `symrdb` and `symioctl` commands, refer to the *EMC Solutions Enabler CLI Command Reference*.

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 of the attributes that describe file systems, directories and files, and their mappings to physical devices and extents. [Table 3](#) summarizes the host file system command.

Table 3 Host file system command

Command	Argument	Description
symhostfs	list	Display a list of file systems, files, or directories.
	show	Display detailed information about a file system or file system object.

[Chapter 4 “File Systems”](#) shows how to use the host file system command. For more information about the **symhostfs** command, refer to the *EMC Solutions Enabler CLI Command Reference*.

Logical volume commands

The logical volume commands map logical volumes and display the underlying storage devices. A 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. [Table 4](#) summarizes the SYMCLI commands for logical volumes.

Table 4 Logical volume commands

Command	Option	Description
symlv	list	Display a list of logical volumes in a specified volume group.
	show	Display detail information (including extent data) about a logical volume.
	add	Add mirror images to a logical volume of the specified type.
	create	Create a logical volume of the specified type.
	delete	Delete a logical volume of the specified type.
	extend	Extend (grows) a logical volume of the specified type.
	reduce	Reduce (shrinks) a logical volume of the specified type.
	remove	Remove mirrors of a logical volume of the specified type.
	stats	Display performance statistics about logical volumes.

Table 4 Logical volume commands (continued)

Command	Option	Description
symvg	adddev	Extend a volume group by adding the specified devices to the volume group.
	create	Create a volume group using the specified devices.
	deport	Deport a specified volume group so it can be imported later.
	destroy	Destroy a volume group.
	import	Import a specified volume group.
	list	Display a list of volume groups defined on your host system by the logical volume manager.
	recover	Recover a failed volume group.
	rescan	Rescan all the volume groups.
	rmdev	Reduce a volume group by removing the specified devices from the volume group.
	show	Display more detail information about a volume group.
	vg2cg	Translate volume groups to composite groups.
	vg2dg	Translate volume groups to device groups.

[Chapter 5 “Logical Volumes”](#) shows how to use the logical volume commands. For more information about the **symlv** and **symvg** commands, refer to the *EMC Solutions Enabler CLI Command Reference*.

Performance statistics commands

The performance statistics commands display statistics about a host’s CPU, disk, memory, logical volumes, and databases. [Table 5](#) summarizes the SYMCLI commands for performance statistics.

Table 5 Performance statistics commands

Command	Argument	Description
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.

[Chapter 6 “Statistics”](#) shows how to use the statistics commands. For more information about the **symhost**, **symlv**, and **symrdb** commands, refer to the *EMC Solutions Enabler CLI Command Reference*.

Daemon service

The SRM daemon is a process or a service that allows certain non-root users and non-administrators to perform SRM operations. For API calls on UNIX systems, the daemon, named `storsrmd`, 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).

For API calls on Windows systems, however, the daemon needs to be running prior to a call that requires root or administrator privileges. Therefore, on Windows systems the SRM daemon (named `storsrmd`) service can be configured to start automatically at boot time, or an authorized user (root or administrator) can start it manually before running any SRM API calls.

Authorized users control daemons using the `stordaeomon` command line utility and can start the SRM daemon as follows:

```
stordaeomon start storsrmd
```

Note: The SRM daemon is available for 64-bit versions of IBM® AIX®, Oracle® Solaris®, HP-UX®, Linux®, and Microsoft® Windows® systems. The SRM daemon supports local client connections only. 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 from another host.

Daemon authorization

Access to the SRM functionality is controlled by limiting permission to the SRM daemon through the daemon authorization file. This file, named `daemon_users`, contains a list of the users authorized to use the daemon and is in the following location:

UNIX	<code>/var/symapi/config/daemon_users</code>
Windows	<code>c:\Program Files\EMC\SYMAPI\config\daemon_users</code>

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

Users meeting any of the following criteria can control and use the SRM daemon:

- ◆ Authorized user: 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 allow users named `smith` and `jones` to use the SRM daemon:

```
smith    storsrmd
jones    storsrmd
```

Defining the SRM operations available to users

You can define the set of SRM control operations that a user can carry out by adding the appropriate control operation keywords to the user's entry in the `daemon_users` file. For example:

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

In this example users `smith` and `jones` can allocate files and extend files. To specify more than one keyword, separate them with commas as the example shows.

[Table 6](#) lists the SRM control operation keywords that can be added for a user.

Table 6 SRM control operations

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

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 Objects

This chapter explains how to display the extents of data files and display other objects using SYMCLI commands.

◆ Conventions.....	26
◆ Data object terminology and architecture	27
◆ Examining device partitions	28
◆ Examining device data objects	31
◆ Examining logical volume data objects.....	33
◆ Examining file system data objects	35

Conventions

This chapter includes summaries of SYMCLI commands that examine physical devices, logical volumes, partitions, file systems, directories, and files. For detailed descriptions of those commands refer to the *EMC Solutions Enabler CLI Command Reference*.

These commands use the following conventions for their parameters:

Table 7 Conventions for parameters to SYMCLI commands for data objects

Parameter	Description
<i>DirName</i>	The name of a directory path.
<i>FileName</i>	The name of a file.
<i>FsName</i>	The name of a file system.
<i>LvolName</i>	The name of a logical volume.
<i>PdevName</i>	The name of a physical device.
<i>VgName</i>	The name of a volume group.

Data object terminology and architecture

The data object commands display the extents of datafiles and objects and the partition information of a physical device. These commands are available on HP-UX, IBM AIX, Oracle Solaris, Microsoft Windows, and Linux.

This section describes data object terminology and SRM architecture:

- ◆ Partitioned physical devices and partitions
- ◆ Data objects
- ◆ Extents
- ◆ Data blocks

Partitioned physical devices and partitions

Partitioned physical devices are made up of portions of a large SCSI device, or an entire SCSI device. Dividing the disk into portions creates more manageable areas with each portion known as a partition. Each partition has a name so that it can be treated as if it were a separate device.

Data objects

A data object is an entity such as a file, a logical volume, a file system, or a physical device that stores data. A data object consists of a set of extents (see [Figure 1](#)). Within a VMAX® array, an object can have extents on multiple devices.

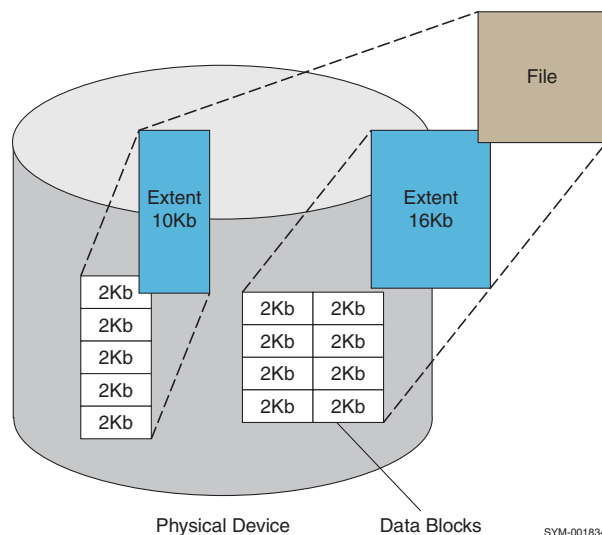


Figure 1 Objects, extents, and data blocks

Extents

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

Data blocks

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 provides information about partitions for host devices.

The command displays detailed information about the structure 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.

You can view partition offset and size information in blocks, kilobytes, or megabytes, by using the **-blocks**, **-kb** and **-mb** options respectively.

Use the following command to list the partitions for a host physical device:

```
sympart show PdevName
```

Options

You can use the following options to tailor the information that the **sympart** command produces:

Table 8 Options for the sympart command

Option	Description
-count <i>n</i>	List the first <i>n</i> partitions only. Replace <i>n</i> with a positive integer. for example: -count 3
-device_type <i>deviceType</i>	List only the partitions with the specified type. Use one of the following values: SYMMETRIX CLARIION
-label <i>partLabel</i>	List only those partitions with the specified partition label. Use one of the following values: MBR GBT VTOC EFI

Table 8 Options for the `sympart` command (continued)

Option	Description
-type <i>partType</i>	List only those partitions with the specified partition type. Use one of the following values: 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	List all partitions except those that meet a specified criterion.
-blocks -kb -mb	Display the offset and size of partitions in 512-byte blocks, kilobytes, or megabytes respectively. If you omit all of these options, the command displays offsets and sizes in megabytes.

Examples

The following lists the partitions for the device `c0t0d0s0`:

```
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/rdisk/c0t0d0s0  Root filesystem      None           0m          32803m
/dev/rdisk/c0t0d0s1  Swap partition       U              32803m      2049m
/dev/rdisk/c0t0d0s2  Full disk            None           0m          34852m
/dev/rdisk/c0t0d0s3  Unassigned partition None           0b          0b
/dev/rdisk/c0t0d0s4  Unassigned partition None           0b          0b
/dev/rdisk/c0t0d0s5  Unassigned partition None           0b          0b
/dev/rdisk/c0t0d0s6  Unassigned partition None           0b          0b
/dev/rdisk/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
```

The following displays partition information for all of the devices connected to a host:

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 following lists all devices except those with a FAT32 partition:

```
sympart list -exclude -type PART_FAT32
```

Examining device data objects

The **symrslv pd** command displays information about data objects down to their physical extents.

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

The **symrslv** command can also work with VMware® and AIX VIO virtualized clients. This means the command can identify array devices behind the virtual disk. The host, with HBAs and a VMAX® or VMAX3 array, creates virtualized environments, or guests. The host pools together disks and can then create virtual disks for guests.

The **symrslv pd** command is used to determine which VMAX or VMAX3 devices need examination.

Use the following command to examine a host physical device:

```
symrslv pd PdevName
```

Options

You can use the following options to tailor the information that the **symrslv pd** command provides:

Table 9 Options for the **symrslv pd** command

Option	Description
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.
-phys_collapse	Display physically contiguous extents as one large extent. The list of extents that the command displays are not logically consistent.
-pdev_extents	Display extents at the physical device level. The command does not expand extents at the level of the underlying metadvice.

Example

The following displays the logical to physical mapping for the device:

/dev/dsk/c5t6d0s2:

symrslv pd /dev/rdisk/c5t6d0s2

```
Absolute Path          : /dev/rdisk/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/rdisk/c5t6d0s2	0b	(S)

Mirror Physical Devices (1):

```
{
-----
Array  Dev  PPdevName          PdevName          Attributes
-----
03122  0082  /dev/rdisk/c5t6d0s2          /dev/rdisk/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.
```

Examining logical volume data objects

The **symrslv lv** command displays the data mapping attributes and the extents of a specified logical volume.

Use the following command to examine a logical volume:

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

Options

You can use the following options to tailor the output from the **symrslv lv** command.

Table 10 Options for the **symrslv lv** command

Option	Description
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.
-phys_collapse	Display physically contiguous extents as one large extent. The list of extents that the command displays are not logically consistent.
-pdev_extents	Display extents at the physical device level. The command does not expand extents at the level of the underlying metadvice.

Example

The following displays information about logical volume `/dev/BigLV`:

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

```
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                      (M)
-----
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.
```

The output shows that the physical device underlying the logical volume is a VMAX metadvice. It also lists the attributes for the logical volume, mirror physical extents, and mirror physical devices.

Examining file system data objects

You can also use the **symrslv** command to list the data mapping attributes and the extents of a data object specified as a file system or a regular datafile.

Use the following command to display a file system:

```
symrslv fs FsName
```

Use the following command to display a datafile:

```
symrslv file FileName
```

Use the following command to display a directory:

```
symrslv dir DirName
```

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

Options

You can use the following options to tailor the output from the **symrslv** command.

Table 11 Options for the **symrslv** command for data objects

Option	Description
-blocks	Display size information 512-byte blocks.
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.
-phys_collapse	Display physically contiguous extents as one large extent. The list of extents that the command displays are not logically consistent.
-pdev_extents	Display extents at the physical device level. The command does not expand extents at the level of the underlying metadvice.

Chapter 4, “File Systems” has more information on file systems.

Examples

File system

The following displays the attributes of the file system /fs_test1:

```
symmrslv 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.

```

File

The following displays the attributes of the file /fs_test1/file1:

```
symmrslv file /fs_test1/file1 -expand -blocks
```

```

Absolute Path           : /fs_test1/file1
Resolved Object Type    : Solaris 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.

```

Directory

The following displays the attributes of the directory /usr:

```
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/rdisk/c0t0d0s0      936896b
}
Mirror Physical Devices (1):
{
-----

Array Dev      PPdevName                PdevName
-----
N/A    N/A    /dev/rdisk/c0t0d0s0      /dev/rdisk/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.

```

CHAPTER 3

Databases

This chapter shows how to display and manage the elements of databases using SYMCLI commands.

◆ Conventions.....	40
◆ Database architecture and terminology	41
◆ Establishing the database connection.....	55
◆ Using database daemons.....	60
◆ Listing database instances	62
◆ Examining database files	63
◆ Translating database devices to VMAX groups	65
◆ Examining tablespaces	67
◆ Translating tablespace devices to VMAX groups	71
◆ Examining schemas	73
◆ Examining tables	75
◆ Examining segments	76
◆ Invoking database I/O control	78
◆ Using the EMC Oracle ASM library	80

Conventions

This chapter includes summaries of SYMCLI commands that examine and translate databases, database files, database devices, tablespaces, schemas, and segments. For detailed descriptions of those commands refer to the *EMC Solutions Enabler CLI Command Reference*.

These commands use the following conventions for their parameters:

Table 12 Conventions for parameters to SYMCLI commands for databases

Parameter	Description
<i>CgName</i>	The name of a composite group.
<i>ComGrpType</i>	The type of a composite group. Use one of the following values: RDF1 RDF2 REGULAR
<i>DbName</i>	The name of a database.
<i>DbType</i>	The type of a database. Use one of the following values: Oracle SQLServer Sybase MVSDB2 IBMUDB Informix Exchange Exchange2007 SharePoint
<i>DevGrpType</i>	The type of a device group. Use one of the following values: RDF1 RDF2 REGULAR
<i>DgName</i>	The name of a device group.
<i>FileName</i>	The name of a file.
<i>Object</i>	The name of a database object.
<i>SaveFile</i>	The name of a file to hold database metadata.
<i>SchemaName</i>	The name of a database schema.
<i>SegmentName</i>	The name of a database segment.
<i>TblName</i>	The name of a database table.
<i>TblSpName</i>	The name of a tablespace in a database.

Database architecture and terminology

The database commands display database mapping and database characteristics. The commands list attributes that describe databases structures, files, tablespaces, and user schemas.

A relational database management system (RDBMS), also known as a relational database server or database server, is the basis for information management. It reliably manages a large amount of data in a multiuser environment allowing many users concurrent access to the same data without degradation of performance. A database server also prevents unauthorized access and provides 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. This collection of processes is the implementation of the database server. 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.

The following sections define the terminology relating to the following database products:

- ◆ Oracle
- ◆ Microsoft SQL Server®
- ◆ Sybase®
- ◆ IBM Informix®
- ◆ IBM DB2®
- ◆ Microsoft Exchange Server®
- ◆ SharePoint® Portal Server

In addition, there is information on the use of the Generic SafeWrite feature to help protect critical data and prevent torn pages.

For information about the most current databases and host platforms supported, refer to the *EMC Solutions Enabler CLI Command Reference*.

Note: Many database architectures include the terms *extent* and *data block*. These are not the same extents and data blocks associated with the data objects described in [Chapter 2](#). For instance, a data block in a database may consist of one or more data blocks of the underlying file system.

Oracle architecture

In Oracle terminology, a *schema* is a collection of objects that constitute a database. Schema objects are logical structures that directly refer to the database's data.

Typical Oracle 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.

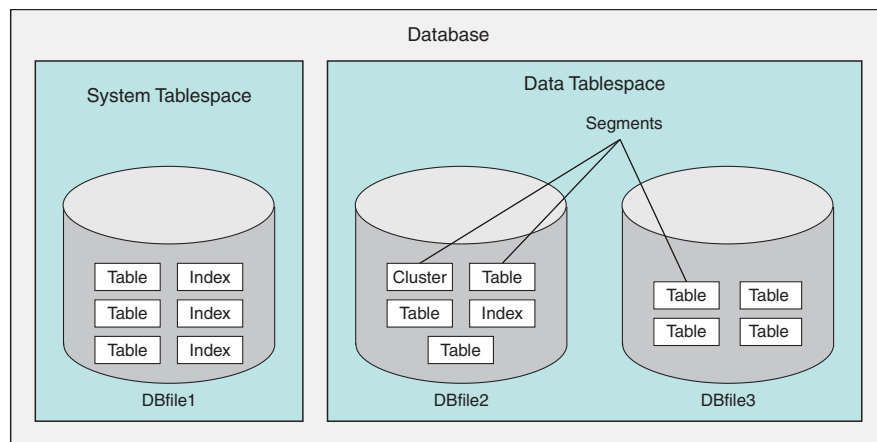


Figure 2 Oracle database architecture

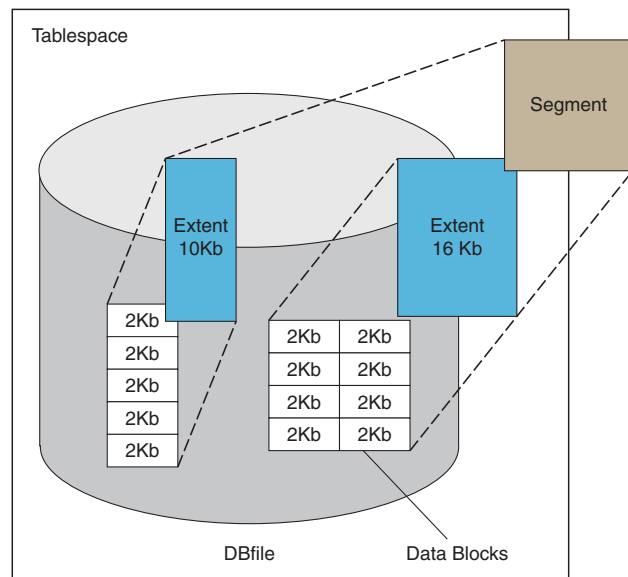
[Table 13](#) defines the terminology of the Oracle architecture.

Table 13 Oracle database terminology

Term	Definition
Table	A table is the basic object and unit of storage. A table is a data structure that has a name and consists of a set of columns and rows. Data occupies each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.
Partitioned table	Partitioning is a means of dividing a very large table into more manageable pieces, each called a partition. Each partitioned table is stored in a separate segment. Optionally, each partition can be stored in a separate tablespace, which has the following advantages: • Limits the impact of any damaged data • Enables back up and recovery of each partition independently • Balances the I/O load by locating partitions on different disk drives
Tablespace	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. Every Oracle database contains a tablespace named SYSTEM, which Oracle creates when creating the database. The SYSTEM tablespace contains the data dictionary tables for the entire database.
Segment	A segment consists of one or more extents that contain all the data blocks for a logical storage structure or object within a tablespace (see Figure 3). There are three types of segment: • User segment — this contains data such as a table, table partition, table cluster, index, or index cluster. • Temporary segment — this contains temporary workspace to hold interim values while processing a SQL query processing. • Undo segment — this contains records of previous transactions and enables the rolling back of a transaction or the recovery of a transaction that was terminated. For example, one or more extents are allocated to each table and to each associated index in a database. 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.
Extent	An extent is a logical storage structure that holds a contiguous set of data blocks for the management of a particular database file. One or more extents make up a segment.
Data block	A data block is a logical storage structure that is the smallest unit of storage and I/O used by the database. The block size is specified when creating a database and is typically 4 KB or 8 KB. A number of contiguous data blocks make up an extent.
Index	An index is an optional structure associated with a table that improves the data retrieval performance. 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. Like large tables, large indexes can be partitioned to make them more manageable. Indexes are created on one or more columns of a table.

Table 13 Oracle database terminology (continued)

Term	Definition
Cluster	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.
View	A view is a user-defined subset of the data from one or more tables. A view consists of the result of a SQL query, that is stored in the database, and given a name. This enables the content of the view to be treated as a table. Conceptually a view is a stored query.
Database link	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.
Data dictionary	A data dictionary contains a set of tables and views that contain read-only reference data about the database. Items that the data dictionary holds include: • Information about the logical and physical structure of the database • A list of valid database users • Table integrity constraints • Space allocation parameters for a schema object Each database has a data dictionary.



SYM-001836

Figure 3 Segments, extents, and data blocks

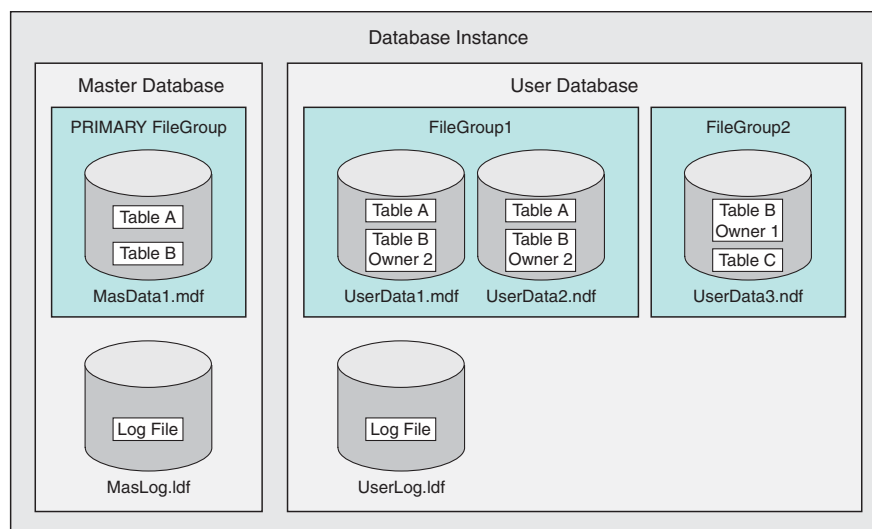
SQL Server architecture

An SQL Server database is a collection of objects directly referring to the data in the database. An instance of SQL Server can manage multiple databases. Typically, in any SQL Server database instance, there are four system databases and one or more user databases.

Each database has a specific owner. The owner's account controls all permissions and ownership of objects in the database. 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.

Typical SQL Server database objects are:

- ◆ Tables
- ◆ Indexes
- ◆ Views



SYM-001837

Figure 4 SQL Server database structure

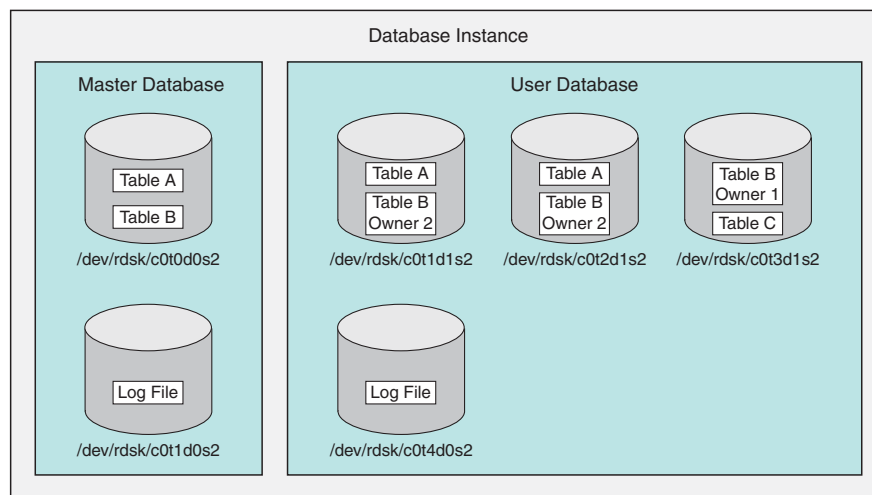
Table 14 defines the terminology of the SQL Server architecture.

Table 14 SQL Server database terminology

Term	Definition
Table	A table is the basic object and unit of storage. A table is a data structure that has a table name and consists of a set of columns and rows. Data occupies each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.
Index	An index is an optional structure associated with a table that improves 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.
View	A view is a user-defined subset of the data from one or more tables. A view consists of the result of a SQL query, that is stored in the database, and given a name. This enables the content of the view to be treated as a table. Conceptually a view is a stored query.
Filegroup	A filegroup is a named storage pool that physically allocates space for the database files. As shown in Figure 4, 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. Filegroups are divided into logical units of storage called extents.
Extent	An extent is a logical storage structure that holds a contiguous set of pages, allocated by the SQL Server processes for the management of a particular database file. An extent consists of eight contiguous pages. A database allocates one or more extents to each table and each index. Any extent can span the data files within a filegroup.
Page	A page is a logical storage structure that is the smallest unit of storage in a database. Each page consists of 8 KB.

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

Sybase architecture



SYM-001838

Figure 5 Sybase database architecture

A database owner is associated with a user within a Sybase database instance. The owner's account controls all permissions and ownership of objects in the database. 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.

[Table 15](#) defines the terminology of the Sybase architecture.

Table 15 Sybase database terminology

Term	Definition
Table	A table is the basic object and unit of storage. A table is a data structure that has a name and consists of a set of columns and rows. Data occupies each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.
Index	An index is an optional structure associated with a table that improves the data retrieval performance. 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.
View	A view is a user-defined subset of the data from one or more tables. A view consists of the result of a SQL query, that is stored in the database, and given a name. This enables the content of the view to be treated as a table. Conceptually a view is a stored query.

Table 15 Sybase database terminology (continued)

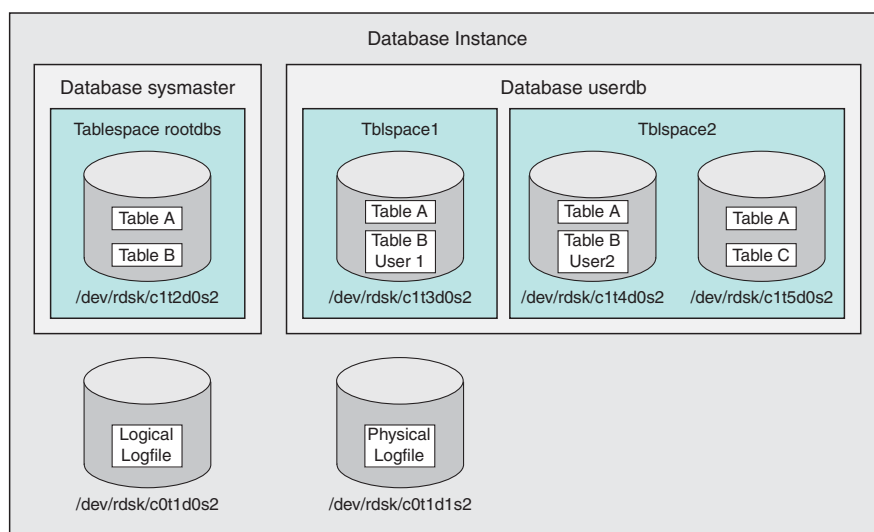
Term	Definition
Instance	An instance is a collection of datafiles, known as devices. There can be one or more devices defined for each Sybase instance. Databases are created and defined on pieces of one or more devices. The administrator decides which devices to use for a particular database at database create time. As shown in Figure 5, 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.
Extent	An extent is a logical storage structure that holds a contiguous set of data blocks, allocated by the Sybase processes for the management of a particular database file. An extent consists of a set of eight pages.
Page	A page is a logical storage structure that is the smallest unit of storage and I/O used by the database. A page contains 2 KB, 4 KB, 8 KB, or 16 KB depending on the options chosen when installing the Sybase software.

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

IBM Informix architecture

In IBM Informix, a database is a collection of logical objects (such as tables, indexes, locator objects, views, and functions), tablespaces, and logical and physical logs. There can be many databases in an IBM Informix server instance each occupying space in one or more tablespaces. A database has a specific owner that can grant or revoke permissions for other users.

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



SYM-001839

Figure 6 Informix database architecture

[Table 16](#) defines the terminology of the IBM Informix architecture.

Table 16 IBM Informix database terminology

Term	Definition
Table	A table is the basic object and unit of storage. A table is a data structure that has a name and consists of a set of columns and rows. Data occupies each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.
Fragmented table	Fragmented tables are typically used when a table has grown too large for a single object. Fragmentation is a means of dividing a very large table into more manageable pieces.
Tablespace	A tablespace is a named storage pool that allocates for the database files. The tablespace consists of one or more physical chunks of storage.
Chunk	A chunk is the largest unit of physical disk storage that a database server uses. A chunk can hold up to 4 TB of data and a database server can have up to 32,766 chunks. The database administrator determines the size of a chunk. Tablespaces are made up of one or more chunks, as Figure 6 shows.

Table 16 IBM Informix database terminology (continued)

Term	Definition
Extent	Chunks are divided into units of storage called extents. An extent is a logical storage structure that holds a contiguous set of pages, that contain all or part of a database object, such as a table. An extent can span chunks within the data space.
Page	The page is a physical unit of disk storage. Typically a page contains 2 KB or 4 KB depending on the operating system that hosts the database server. A number of contiguous data blocks make up an extent.
Index	An index is an optional structure associated with a table that improves the data retrieval performance. 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. Like large tables, large indexes can be fragmented to make them more manageable.
View	A view is a user-defined subset of the data from one or more tables. A view consists of the result of a SQL query, that is stored in the database, and given a name. This enables the content of the view to be treated as a table. Conceptually a view is a stored query.

IBM DB2 architecture

In IBM DB2, an instance (also called a database manager) comprises DB2 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.

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

DB2 logical database objects include:

- ◆ Tables
- ◆ Indexes
- ◆ Views
- ◆ Functions
- ◆ Large objects (LOB)

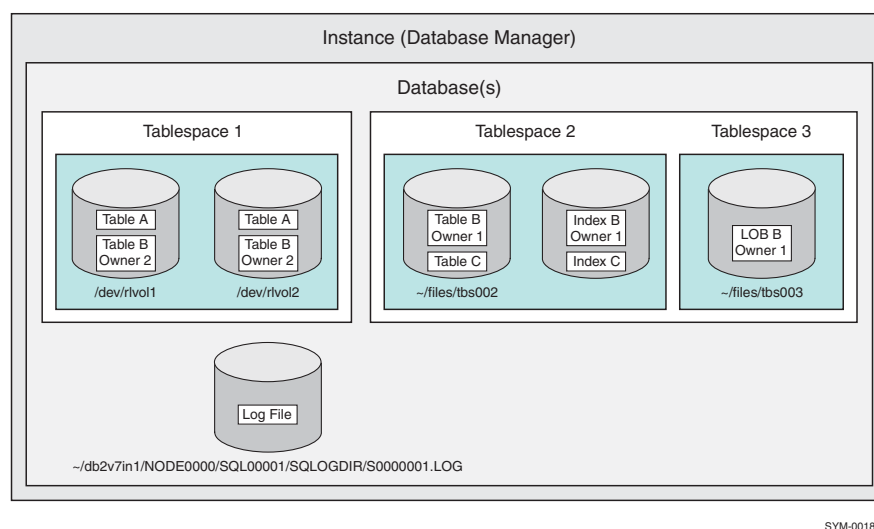


Figure 7 IBM DB2 database architecture

[Table 17](#) defines the terminology of the IBM DB2 architecture.

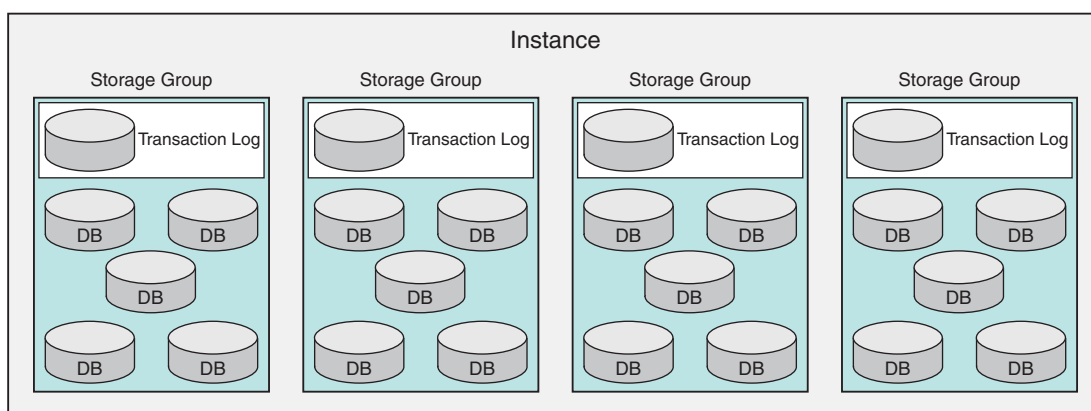
Table 17 IBM DB2 database terminology

Term	Definition
Tablespace	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. Each database contains at least three tablespaces: • Catalog (SYSCATSPACE) • User (USERSPACE1) • Temporary (TEMPSPACE1) Additional table spaces are typically created to contain tables, indexes, and LOBs.
Table	A table is the basic object and unit of storage. A table is a data structure that has a name and consists of a set of columns and rows. Data occupies each cell formed by a row/column intersection. A row is a collection of column information corresponding to a single record.
Index	An index is an optional structure associated with a table that improves the data retrieval performance. 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.
View	A view is a user-defined subset of the data from one or more tables. A view consists of the result of a SQL query, that is stored in the database, and given a name. This enables the content of the view to be treated as a table. Conceptually a view is a stored query.
Extent	An extent is a logical storage structure that holds a contiguous set of data blocks, that contain all or part of a database object, such as a table. DB2 allocates extents among the data containers that make up the database in round robin fashion.
Data block	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.

Microsoft Exchange Server architecture

As [Figure 8](#) shows, an Exchange Server is a single instance of Exchange consisting of storage groups and databases. Each Exchange Server has a maximum of four storage groups. Within SYMCLI, an Exchange storage group is referred to as an Exchange tablespace.

Each Exchange storage group consists of 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. Data is written to the transaction logs first, then placed in memory, and finally committed to the datafiles on disk. Each storage group also has a checkpoint file that keeps track of transactions that have been committed to their databases, and those that have not been committed.



SYM-001841

Figure 8 Microsoft Exchange database architecture

Exchange Server has 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 reside in the Microsoft Windows Active Directory®. They are attributes of a Windows user object.

An Exchange public store is similar to the mailbox store, except that all of the data is organized logically into a folder hierarchy instead of mailboxes.

Different databases in different storage groups can have the same name. To make Exchange database names unique within SYMCLI, use names in the form *storagegroup\database*. For example, if storage groups *SG1* and *SG2* each have a database *DB1*, the database names in SYMCLI are *SG1\DB1* and *SG2\DB1*, respectively. Since 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) are invalid.

An Exchange database consists of two datafiles, a binary datafile, and a streaming file. The binary datafile uses a binary tree structure. The streaming file consists of flat, 64 KB runs. The binary datafile is the traditional datafile that stores the majority of typical email data. The streaming file is intended to store native Internet content in large, sequential chunks. The sequential layout of the streaming file allows for faster retrieval of 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.

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.

In Exchange, there is no concept of 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 by Exchange itself. In Exchange, data blocks always use 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. SharePoint Portal Server runs on Exchange Server and so its architecture 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, the Generic SafeWrite feature helps to protect critical applications from incurring an incomplete write, and subsequent torn page, due to a failure with a component connected to the VMAX or VMAX3 Front-End Channel Adapter. Generic SafeWrite is most often used to protect against corruption from HBA and link failures, including server crashes. Essentially, it helps protect against fractured writes that can occur before the data reaches the storage array.

A RDBMS, such as Microsoft Exchange, structures data within database files using data blocks. The size of a block varies from one database product to another. For example, the block size for Microsoft Exchange is 4 KB and for Oracle it is usually 8 KB, although that can be configured. If an incomplete block is written to a database file, the database becomes corrupted. That corruption is known as a *torn page*.

Most RDBMSs detect torn pages after the corruption has been written, when they read that area of the database, which could be long after when the corruption occurred. In general, the only way to recover from a torn page is to restore the database 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.

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 and Oracle. Generic SafeWrite is not intended for use with applications where torn pages are not a concern, such as fileshares or FTP servers.

Generic SafeWrite is enabled by default for Engenuity 5773 and later. For more information on enabling and using the feature, refer to the *EMC Solutions Enabler Array Management CLI User Guide*.

Establishing the database connection

Setting UNIX environment variables

To be able to start a CSYMCLI session for database commands, some environment variables need defining on the host system. [Table 18](#) lists the variables required for all databases and any additional ones that are product-specific.

Table 18 UNIX database environment variables

Database	Variables to set
All databases	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	DB2INSTANCE

- Set for Solaris and points to the DB client libraries.
- Set for Solaris and points to the DB client libraries (only if you need to override LD_LIBRARY_PATH).
- Set for HP-UX and point to the DB client libraries.
- Set for AIX and point to the DB client libraries.
- 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.
- Must point to the correct Sybase server, That is, 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 exist between the database and the application that issues 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 non-root user on Solaris, manually start the Sybase database daemon from the root user before calling any Sybase database functions.

If the connection is remote, supply a network service name 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 enable startup and shutdown of the specified database server manager instance. With the SYMCLI client/server capability, the database manager instance startup or shutdown can be done from the client side. This removes the limitation of most databases, that only allow startup and shutdown of the database manager instance on the database server side.

Use the `symrdb startup -type DbType` to start Sybase, SQL Server, DB2, and Oracle databases.

Sybase

The following options are available for Sybase databases:

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

where:

runserver_file is 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.

SQL Server

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 is the name of the instance start.

master_file_path is the path name of the master database file.

error_log_path is the path name of the error log file.

master_log_path is the path name of the master database log file.

virtual_addr_space is the amount of virtual address space in megabytes.

trace_number specifies the trace number.

DB2

The following startup options are available for DB2 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 starts the node in STANDALONE mode.

NODE defines that the containers for the temporary tablespaces are the same as those for the specified node.

CATALOG defines that the containers for the temporary tablespaces are the same as those for the catalog node of each database.

profile is the name of the profile.

node is the node number.

hostname is the system name.

port is the port number.

netname is the net name.

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

computer is 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 use while starting up the instance.

DbName specifies the name of the database to mount or open. Refer to the Oracle documentation for the definition of database names.

FORCE forces the current Oracle instance to shut down (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 mode.

OPEN_READ_WRITE_RECOVER specifies READ WRITE to open the database in read/write mode and perform that media recovery, if necessary, before starting the instance.

Database shutdown options

Use the `symrdb shutdown -type DbType` command to shut down Sybase, SQL Server, DB2, and Oracle 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 of the backup server in the server's system table.

wait shuts down the server in an orderly fashion.

nowait shuts down the server immediately.

SQL Server

The following shutdown options are available for SQLServer databases:

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

where:

instance specifies the name of the instance to shut down.

DB2

The following shutdown options are available for the DB2 database:

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

where:

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 is the name of the profile.

node is the node number.

Oracle

The following shutdown options are available for Oracle databases:

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

where:

ABORT performs the fastest possible shutdown. The system does not wait for calls to complete or for users to disconnect.

IMMEDIATE does not wait for current calls to complete, prohibits further connects, closes the database, and dismounts the database. Finally, the system shuts down the instance. It does not wait for connected users to disconnect. However, this option does not require instance recovery on next startup.

NORMAL waits for currently connected users to disconnect from the database, prohibits further connects, and closes the database and dismounts the database. Finally, the system shuts down the instance. This option does not require instance recovery on next startup. **NORMAL** is the default option.

TRANSACTIONAL shuts down an instance while minimizing interruption to clients. No client can start a new transaction on the instance.

LOCAL carries out a transactional shutdown on the local instance only.

Using database daemons

Solutions Enabler provides a number of database utilities called *daemons*. This section describes what daemons are, explains why they are useful, and provides an overview of 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 and databases.

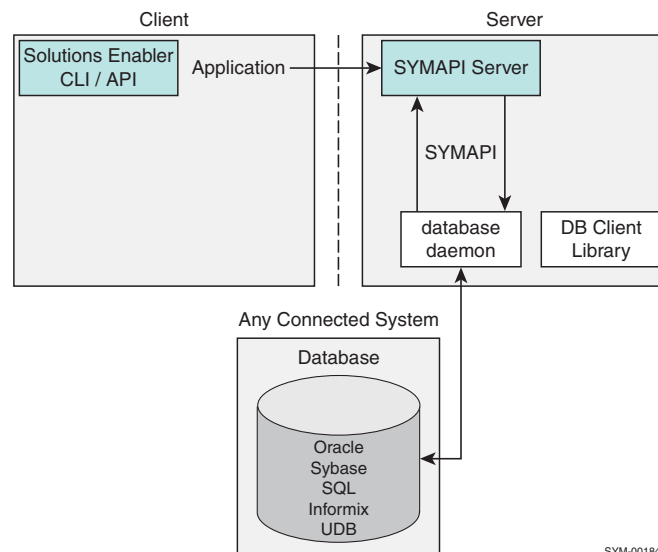


Figure 9 System view of the daemon process

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 Microsoft Windows platform, a database daemon runs as a Windows service. After the service is installed, it appears in the service list dialog box and can be accessed by through the control panel. The service does not restart at boot time unless you modify the startup options through the service list dialog.

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

You can start or stop a daemon manually. 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, restart the daemon to reconnect to the server.

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

```
stordaeomon start stororad
```

By default, the `stordaeomon` command waits 30 seconds to verify that the daemon is running. To override this use the `-wait` option. For example:

```
stordaeomon start stororad -wait 0
```

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

Always set the `LD_LIBRARY_PATH` environment variable, or its equivalent counterpart environment variable, for all database daemons except Oracle and SQL Server. See [Table 18 on page 55](#) 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 62](#).

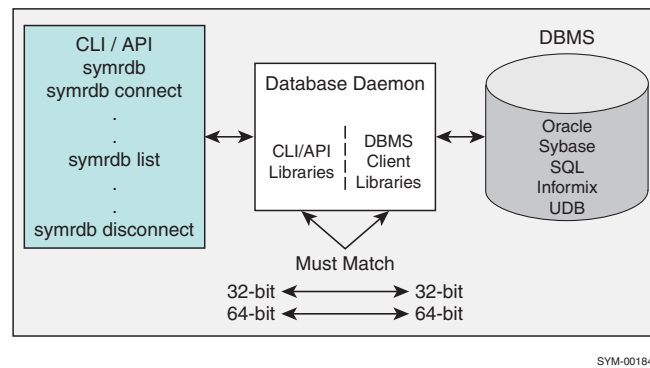


Figure 10 Database daemon libraries for communication

Note: The database daemons `stororad` and `storsysbs12.5d` require the optional libraries `/usr/symcli/shlib` and DBMS client libraries at product installation time.

Table 19 lists the database daemons.

Table 19 Database daemon names

Daemon name	Database type
<code>stororad</code>	Oracle
<code>storubd</code>	DB2
<code>storifmxd</code>	IBM Informix
<code>storifmx64d</code>	IBM Informix (64-bit)
<code>storsql</code>	SQL
<code>storsybs12d</code>	Sybase Version 12
<code>storsybs12.5d</code>	Sybase Version 12.5

The *EMC Solutions Enabler Installation Guide* contains information on installing and using daemons.

Listing database instances

Use the following command to list the available databases (database instances):

```
symrdb list -type DbType
```

For Oracle databases, the command lists the current Oracle database instance. For example:

```
symrdb list -type Oracle
```

```
DATABASE NAMES (ORACLE 10.1.0.1.0):
```

```
Database Name
-----
```

```
64806
```

Examining database files

Use the following command to identify all the files in a database:

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

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

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

Examining database file attributes

Use the following command to list the attributes and the extent information of a file in a database:

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

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

Options

You can use the following options to tailor the information that the command produces:

Table 20 Options for the symrdb show command for database files

Option	Description
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.

Example

The following displays information about the file `rbs01.dbf` in the Oracle database named `64806`:

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

```
DATABASE           : 64806
DATABASE FILE NAMES (ORACLE 10.1.0.1.0):
```

```
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    : Solaris 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/rdisk/c0t1d0s0      11654m
32k   N/A  N/A        N/A  /dev/rdisk/c0t1d0s0      11654m
1m    N/A  N/A        N/A  /dev/rdisk/c0t1d0s0      12409m
120k  N/A  N/A        N/A  /dev/rdisk/c0t1d0s0      12410m
...

```

```
Mirror Physical Devices (1):
```

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

Listing tablespace files

Use the following command to list all the files in a tablespace and database:

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

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

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

Translating database devices to VMAX groups

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

Database devices to device groups

Use the following command to create a device group from a database:

```
symrdb -db DbName rdb2dg DgName -dgtype DevGrpType
```

Options

You can use the following options to define which devices in the database to include in the device group:

Table 21 Device options for the symrdb rdb2dg command

Option	Description
-bcv	Include only BCV devices in the device group.
-nobcv	Include only standard devices in the device group.
-vdev	Include VDEVs in the device group.

If you omit these options, the command includes standard and BCV devices in the device group.

Examples

The following creates an RDF1 device group named `Prod1dg` with only the standard devices from the database named `Prod1db`:

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

The following creates a REGULAR device group named `Prod2dg` with only the R2 BCV devices from the database named `Acc2db`:

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

Database devices to composite groups

Use the following command to create a composite group from a database:

```
symrdb -db DbName rdb2cg CgName -cgType ComGrpType
```

Options

You can use the `-vdev` option to add virtual device to a composite group.

Examples

The following creates a composite group named `ProdRcg` with the R1 standard and R1 BCV devices from the database named `ADB4db`:

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

The following creates a composite group named `myrdfcg` with only the R1 standard devices from the database named `Accdb`:

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

Examining tablespaces

Use the following command to list all the tablespaces that a database uses:

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

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 example, the following lists the tablespaces in the Oracle database 64806:

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

```
DATABASE          : 64806
TABLE SPACE NAMES (ORACLE 10.1.0.1.0):
```

Table Space Name	Type	Status	Size (mb)	Free (mb)
RBS	Permanent	Online	30	26
SYSTEM	Permanent	Online	64	46

Examining tablespace attributes

Use the following command to list the attributes and extent information of a tablespace of a specified database:

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

Options

You can use the following options to tailor the information that the command produces:

Table 22 Options for the symrdb show command for tablespaces

Option	Description
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.

Example

The following lists the attributes of the tablespace `SYSTEM` in the Oracle database 64806:

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

```
DATABASE          : 64806
TABLE SPACE NAMES (ORACLE 10.1.0.1.0):
```

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 : Solaris 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/rdisk/c0t1d0s0          11654m
32k   N/A  N/A          N/A  /dev/rdisk/c0t1d0s0          11654m
80k   N/A  N/A          N/A  /dev/rdisk/c0t1d0s0          12004m
1m    N/A  N/A          N/A  /dev/rdisk/c0t1d0s0          12004m
...
}
```

Mirror Physical Devices (1):

{

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

Listing tablespace files

Use the following command to list all the files in a tablespace and database:

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

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

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

Listing tablespace tables

Use the following command to list the tables in a tablespace:

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

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

For example the following lists the tables in the tablespace `SYSTEM`:

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

```
TABLE SPACE NAME      : SYSTEM
DATABASE TABLE NAMES (ORACLE 10.1.0.1.0):
```

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

Listing segments in a tablespace

Use the following command to list the segments of a tablespace. A segment (storage unit) exists for every data table, partition, and index.

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

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

For example, the following lists the segments of tablespace `SYSTEM`, in the Oracle database 64806:

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

```
DATABASE           : 64806
TABLE SPACE NAME    : SYSTEM
DATABASE SEGMENT NAMES (ORACLE 10.1.0.1.0):
```

Segment Name -----	Owner -----	Type -----
1.527	SYS	Cache
C_COBJ#	SYS	Cluster
C_FILE#_BLOCK#	SYS	Cluster
SYS_C00512	SYS	Index
SYS_IL00000000227C00002\$	SYS	LOB_Index
SYS_LOB00000000235C00002\$	SYS	LOB_Segment
SYS_LOB00000000239C00002\$	SYS	LOB_Segment
SYSTEM	SYS	Rollback
ACCESS\$	SYS	Table

Translating tablespace devices to VMAX groups

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

Tablespace devices to device groups

Use the following command to create a device group from a tablespace:

```
symrdb -tbs TblSpName tbs2dg DgName -dgtype DevGrpType
```

Options

You can use the following options to define which devices in the database to include in the device group:

Table 23 Device options for the symrdb tbs2dg command

Option	Description
-bcv	Include only BCV devices in the device group.
-nobcv	Include only standard devices in the device group.
-vdev	Include VDEVs in the device group.

If you omit these options, by default the command includes standard and BCV devices in the device group.

Examples

The following creates an RDF1 device group named `Prod1dg` with only the standard devices from the tablespace named `Prod1tbs`:

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

The following creates a REGULAR device group named `Prod2dg` with only the R2 BCV devices from the tablespace named `Acc2tbs`:

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

Tablespace devices to composite groups

Use the following command to create a composite group from a tablespace:

```
symrdb -tbs TblSpName tbs2cg CgName -cgtype ComGrpType
```

Options

You can use the following options to define which devices in the database to include in the device group:

Table 24 Device options for the symrdb tbs2cg command

Option	Description
-bcv	Include only BCV devices in the device group.
-nobcv	Include only standard devices in the device group.
-vdev	Include VDEVs in the device group.

If you omit these options, by default the command includes standard and BCV devices in the device group.

Examples

The following creates a composite group named `ProdRcg` with the R1 standard and R1 BCV devices from the tablespace named `ADB4tbs`:

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

The following creates a composite group named `myrdfcg` with only the R1 standard devices from the tablespace named `Acctbs`:

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

Examining schemas

Use the following command to list all the schemas in a database:

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

For example, The following lists the schemas for Oracle database 64806:

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

```
DATABASE      : 64806
SCHEMA NAMES (ORACLE 10.0.1.0.0):
```

```
Schema Name
-----
```

```
SYS
SYSTEM
DBSNMP
```

Listing schema files

Use the following command to list all the files in a schema:

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

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

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

For example, the following lists the files in the SYSTEM schema for the Oracle database 64806:

```
symrdb list -type Oracle FILE -schema SYSTEM -db 64806
```

Listing schema tables

Use the following command to list the tables in a schema:

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

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

- ◆ Database version
- ◆ Names of the tables
- ◆ Owner of each table
- ◆ Type of each table: partitioned, non-partitioned, clustered, index-organized, nested, or N/A

For example, the following lists the schema tables of the SYSTEM schema in the Oracle database 64806:

```
symrdb list -type Oracle TABLE -schema SYSTEM -db 64806
```

Examining schema attributes

Use the following command to list the attributes and extent information for a schema of a database:

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

Options

You can use the following options to tailor the information that the command produces:

Table 25 Options for the symrdb show command for schema attributes

Option	Description
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.

Example

The following lists the attributes of the schema `SYSTEM` in the Oracle database 64806:

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

```
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     : Solaris 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/rdisk/c0t1d0s0        11654m
32k   N/A  N/A       N/A  /dev/rdisk/c0t1d0s0        11654m
80k   N/A  N/A       N/A  /dev/rdisk/c0t1d0s0        12004m
...
}
```

```
Mirror Physical Devices (1):
```

```
{
-----
Array Dev  PPdevName                PdevName
```

```

-----
N/A    N/A    /dev/rdsk/c0t1d0s0    /dev/rdsk/c0t1d0s2
...
}

```

Listing segments in a schema

Use the following command to list the segments in a database schema. A segment (storage unit) exists for every data table, partition, and index in a schema.

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

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

For example, the following lists the segments of the schema `SYSTEM` in the Oracle database `64806`:

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

```

DATABASE           : 64806
SCHEMA NAME        : SYSTEM
DATABASE SEGMENT NAMES (ORACLE 10.1.0.1.0):

```

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

Examining tables

Use the following command to list the tables of a tablespace:

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

Use the following command to list the tables of a schema:

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

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

- ◆ Database version
- ◆ Names of the tables
- ◆ Type of each table: partitioned, non-partitioned, clustered, index-organized, nested, or N/A
- ◆ Owner of each table
- ◆ Index-organized tablename
- ◆ Nested parent table information
- ◆ Cluster name information

Examining table attributes

Use the following command to list the attributes of a table in a tablespace:

```
symrdb show -type DbType TABLE TblName -tbs TblSpName -db DbName
```

Use the following command to list the attributes of a table in a database schema:

```
symrdb show -type DbType TABLE TblName -schema SchemaName
```

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

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

You can use the following options to tailor the information that the command produces:

Table 26 Options for the symrdb show command for table attributes

Option	Description
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.

Examining segments

Use the following command to list all the segments of a tablespace:

```
symrdb list -type DbType SEG -tbs TblSpName
```

Use the following command to list all the segments of a database schema:

```
symrdb list -type DbType SEG -schema SchemaName
```

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 example, the following displays the segments for tablespace SYSTEM in the Oracle database 64806:

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

```
DATABASE           : 64806
TABLE SPACE NAME    : SYSTEM
DATABASE SEGMENT NAMES (ORACLE 10.1.0.1.0):
```

Segment Name -----	Owner -----	Type -----
1.527	SYS	Cache
C_COBJ#	SYS	Cluster
SYS_C00512	SYS	Index
SYS_IL00000000227C00002\$	SYS	LOB_Index
SYS_LOB00000000235C00002\$	SYS	LOB_Segment
SYS_LOB00000000239C00002\$	SYS	LOB_Segment
SYSTEM	SYS	Rollback
ACCESS\$	SYS	Table
AQ\$_MESSAGE_TYPES	SYS	Table

Examining segment attributes

Use the following command to list the attributes of a segment of a tablespace:

```
symrdb show -type DbType SEG SegmentName -tbs TblSpName -db DbName
```

Use the following command to list the attributes of a segment of a schema:

```
symrdb show -type DbType SEG SegmentName -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

Invoking database I/O control

The `symioctl` command provides the following control operations for a database and database objects:

- ◆ Freezing a database
- ◆ Thawing a database
- ◆ Hot backup control (Oracle only)
- ◆ Checkpointing
- ◆ Archiving a log (Oracle only)
- ◆ Taking a snapshot of a SQL Server database

Freezing a database

Use the following command to freeze all I/O access to a database:

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

IBM DB2 and SQL Server allow some or all databases to be specified. Oracle and IBM Informix allow a user to freeze or thaw an entire database system only; for these databases exclude the list of database objects.

For example, the following freezes databases HR and Payroll:

```
symioctl freeze -type SQLSever 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: Avoid freezing the "master" database of a SQL Server instance as this can lead to unpredictable behavior.

Thawing a database

Use the following command to resume full I/O access to the database instance, reversing a previous freeze command:

```
symioctl thaw -type DbType
```

Note: You cannot freeze nor thaw an Oracle database over 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 be performed on a list of tablespace objects only. In addition, it must be performed before a freeze command and after a thaw command. For example, 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

Use the following command to issue a checkpoint on a database:

```
symioctl checkpoint -type DbType Object...
```

Oracle and IBM Informix checkpoint the entire database system; for these databases exclude the list of objects. SQL Server allows some or all databases to be specified. IBM DB2 does not support checkpointing.

Archiving a log

On an Oracle database, you can archive the current log. This forces the log writer process to stop writing to the current log file and switch to the next available online log file.

Taking a snapshot of a SQL Server database

A snapshot of an SQL Server database is a copy of the data and log files at a single point in time. The **symioctl** command provides an interface to the SQL Server snapshot feature. When combined with TimeFinder instant split, you can create copies of a database quickly for backup, reporting, testing, and initialization of warm standby servers.

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

Precede the TimeFinder split with the following command:

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

By default, the *savefile* is not overwritten. To replace it, either remove it or use the **-overwrite** option.

Follow the TimeFinder split with this command:

```
symioctl end snapshot Object
```

Note that all writes to the database are suspended during the begin/end snapshot sequence so use TimeFinder instant split to minimize transaction delays.

If a problem occurs that requires termination of the backup, use the following command:

```
symioctl abort snapshot Object
```

Follow the TimeFinder restore with this command:

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

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

The ASM syntax for disk discovery

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' | 'blksiz' |
StringAttribute:= '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 StringAttribute 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.

Examples

The following constructs a disk set of all configured devices known to ASM matching the pattern /dev/risk/* and a capacity of more than one gigabyte.

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

The following creates a diskgroup dg1 with external redundancy using device /dev/risk/c3t0d1s6.

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

Usage of CLI `asmddscvr`

The **asmddscvr** 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 **asmddscvr** do not need to be available through normal operating system interfaces.

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

The `asmddscvr` utility provides the following:

```
asmddscvr -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 help information

`-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

The following fetches all configured devices known to ASM matching the pattern `/dev/rdisk/*`:

```
asmddscvr -d "PATH=/dev/rdisk/*"
```

The following fetches all configured devices known to ASM with a capacity greater than two gigabytes:

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

The following fetches all configured devices known to ASM matching the pattern `/dev/rdisk/*` and of capacity greater than two gigabytes, enter:

```
asmddscvr -d "PATH = /dev/rdisk/* AND SIZE > 2G"
```


CHAPTER 4

File Systems

This chapter shows how to examine host the file systems and their structure using SYMCLI commands.

◆ Conventions.....	84
◆ Overview.....	85
◆ Finding file systems	86
◆ Examining a file system.....	87
◆ Finding directories and files	88
◆ Examining directories.....	89
◆ Listing files	89
◆ Examining files	90

Conventions

This chapter includes summaries of SYMCLI commands that examine file systems and their components. For detailed descriptions of those commands refer to the *EMC Solutions Enabler CLI Command Reference*.

These commands use the following conventions for their parameters:

Table 27 Conventions for parameters to SYMCLI commands for file systems

Parameter	Description
<i>DirName</i>	The name of a directory path.
<i>FileName</i>	The name of a file.
<i>Object</i>	The name of an object in a file system. That is, a directory or file.

Overview

The file system commands list the attributes of file systems, directories, and files. In addition, the commands list the mapping of these objects to physical devices and extents.

Operating systems and file systems

The file system commands operate on the following operating systems and file systems:

- ◆ 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)
- ◆ Oracle Solaris (ufs, VxFS, NFS)
- ◆ VMware file system (VMFS)
- ◆ Windows (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 User Guide* contains information on setting up VMware virtual disk mapping support.

Terminology

[Table 28](#) defines the terminology for file systems.

Table 28 File system terminology

Term	Definition
File system	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.
Extent	A block of storage space on a disk, which is reserved by the file system for the storage of a particular file or program.
i-node	An internal data structure for file systems of UNIX-based operating systems. An i-node contains the file type, owner, size, modified date, and physical device location of an individual file. Typically, a table of i-nodes is stored near the beginning of a file system.

Finding file systems

Use the following command to list the file systems that are on the host system:

```
symhostfs list
```

The command lists the following attributes for each file system:

- ◆ Device name where the file system is located
- ◆ Type of file system
- ◆ File system mount point

Options

You can use the **-nfs** option to list file systems with NFS mount points.

Examples

The following lists the file systems on the host system:

```
symhostfs list
```

Mounted File Systems

Device Name -----	Type -----	Mount Point -----
/dev/vg00/lvol3	HP VxFS	/
/dev/vg00/lvol11	HFS	/stand
/dev/vg00/lvol18	HP VxFS	/var
/dev/vg00/lvol17	HP VxFS	/usr
/dev/vg00/lvol14	HP VxFS	/tmp
api193:(pid787)	Unknown Type	/net

The following lists mounted files systems with NFS mount points:

```
symhostfs list -nfs
```

Mounted File Systems

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

Examining a file system

Use the following command to list the attributes of a specified file system:

```
symhostfs show Object
```

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

Options

You can use the following options to tailor the information that the command produces:

Table 29 Options for the `symhostfs show` command

Option	Description
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information.

Example

The following lists the attributes of a Linux file system with the mount point of `/stand` in a clustered environment. The file system is mounted on the logical device `lv011` on the physical device `c0t3d0`.

```
symhostfs show /stand
```

```
File System Mount Point      : /stand
File System Type             : EXT3
Mount Options                : Read Write | Clustered
File System Device Name      : /dev/vg00/lv011
Size                         : 82m
Number Free Blocks           : 117778
Fragment Size                : 1024 bytes
Block Size                   : 8192 bytes
Number Inodes                : 13440
Number Free Inodes           : 13403
Mount Time                   : Mon 18-Mar-2013 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/lv011
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/rdisk/c0t3d0              3m
```

```
Mirror Physical Devices (1):
```

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

Finding directories and files

Use the following command to list the subdirectories of any root or parent directory:

```
symhostfs list -dir DirName
```

Options

You can use the **-R** option to display recursively subdirectories in the directory tree.

Examples

The following lists the subdirectories in the `/stand` directory:

```
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 following lists all subdirectories recursively:

```
symhostfs list /stand -R
```

```
...
```

```
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

Use the following command to list the attributes of a directory:

```
symhostfs show DirName
```

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

For example, the following lists the attributes of the directory /stand/dlkm/system.d:

```
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	: Mon 18-Mar-2013 11:57
Last Modification Time	: Mon 03-Dec-2012 15:19
Last Status Change Time	: Mon 03-Dec-2012 15:19

Listing files

Use the following command to list all the files in a directory:

```
symhostfs list -file DirName
```

The command lists all files found in the specified directory. For each file, the output shows its name, size, permissions, owner, and group.

Options

You can use the **-R** option to display recursively subdirectories in the directory tree.

Examples

The following lists the files in the `/stand` directory:

```
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 following shows a portion of the output when the `-R` option is present:

```
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

Examining files

Use the following command to list the attributes of a file:

```
symhostfs show FileName
```

Note: Some journaled file systems 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.

Options

You can use the following options to tailor the information that the command produces.

Table 30 Options for the `symhostfs show` command for a file

Option	Description
-data	Include only data objects in the output. You cannot use this option together with the <code>-meta</code> option.
-meta	Include only metadata objects in the output. You cannot use this option together with the <code>-data</code> option.
-expand	Display multiple, contiguous extents as separate extents. By default, the command collapses contiguous extents into one large extent.
-no_extents	Display information about the object, excluding extent information

Example

The following lists the attributes of the file named `system` in the `/stand` directory:

```
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 18-Mar-2013 16:20
Last Modification Time    : Mon 03-Dec-2012 15:12
Last Status Change Time   : Mon 03-Dec-2012 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/rdisk/c0t3d0                3m
}

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

Array Dev  PPdevName                      PdevName
-----
N/A    N/A    /dev/rdisk/c0t3d0                /dev/rdisk/c0t3d0
}
}

```

CHAPTER 5

Logical Volumes

This chapter shows how to examine and manage logical volumes and their groups using SYMCLI commands.

◆ Conventions.....	94
◆ Overview.....	95
◆ Viewing volume groups	102
◆ Volume group control operations	105
◆ Translating volume groups to VMAX groups	107
◆ Viewing logical volumes	109
◆ Control operations for logical volumes	112
◆ Viewing extents.....	114

Conventions

This chapter includes summaries of SYMCLI commands that examine and manage logical volumes. For detailed descriptions of those commands refer to the *EMC Solutions Enabler CLI Command Reference*.

These commands use the following conventions for their parameters:

Table 31 Conventions for parameters to SYMCLI commands for logical volumes

Parameter	Description
<i>CgName</i>	The name of a composite group.
<i>DgName</i>	The name of a device group.
<i>GrpNum</i>	The SRDF RA group number.
<i>LVolName</i>	The name of a logical volume.
<i>Mirrors</i>	The number of mirrors to add to or remove from a logical volume.
<i>PartitionSize</i>	The size of a partition in megabytes.
<i>PdevName</i>	The name of a physical device.
<i>Size</i>	The size of a logical volume. By default this is in 512-byte blocks but you can use any of the following suffixes to specify the unit of capacity: • <i>b</i> (size in blocks) • <i>k</i> (size in kilobytes) • <i>m</i> (size in megabytes)
<i>SymmID</i>	The unique ID of a VMAX array.
<i>VgName</i>	The name of a volume group.
<i>VgType</i>	The type of a volume group. Table 35 on page 103 lists the values for this parameter.

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 volume group is a set of physical devices managed by a LVM. The physical devices form a pool of disk space that the LVM uses to define logical volumes, or virtual devices. In turn, the LVM makes these volumes available to applications and databases. Users and applications see a logical volume as another physical device.

The user controls how the LVM uses physical disk space to create a logical volume by specifying its characteristics—how many copies of the data to retain, 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 is allocated in units called 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 are retained.

A physical extent is a specific 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 creating a volume group and applies to all physical devices in the group. Exceptions are the logical volume managers (VxVM®, ptx/SVM, LSM, DiskADM, and LDM) that allow the size of physical extents, or subdisks to vary.

The physical extents associated with a logical volume can occupy one or more physical devices, and may or may not be contiguous.

[Figure 11 on page 96](#) illustrates a volume group that contains a logical volume (VOL1) whose logical extents are mapped to the physical devices PDEV1 and PDEV2.

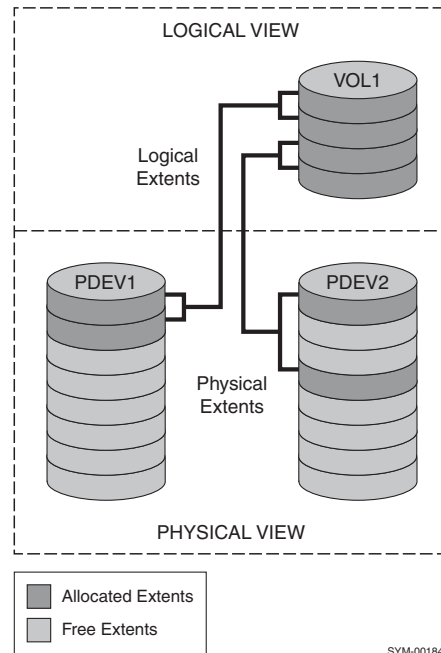


Figure 11 Volume group

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 such as NTFS, FAT, or CDFS, the volume mount point is similar to a UNIX file system mount point. When the associated volume is "raw" (not formatted), the volume mount point is similar to a UNIX logical volume path. When a single Windows volume is associated with a drive letter and multiple volume mount points, the volume mount point is similar to a UNIX symbolic link.

Windows 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 Configuration Manager allows users to create a volume without assigning any drive letters or volume mount points. If the Windows volume has a drive letter, the logical volume name is in the `v:` form. If the volume does not have a drive letter, 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 mirror

In a simple mirror configuration there are two or more copies of the logical volume's data that use physical extents on one physical device. In [Figure 12](#), all physical extents are on one physical device, PDEV1.

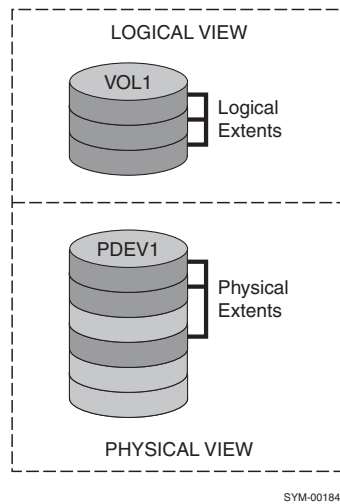


Figure 12 Simple mirror configuration

Concatenated mirror

A concatenated mirror configuration has two or more copies of the data that use physical extents on two or more physical devices.

In [Figure 13](#), the logical extents on VOL1 point to the physical extents on PDEV1 and PDEV2.

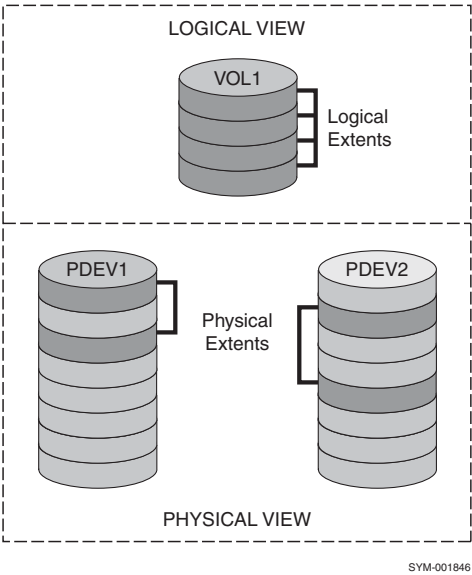


Figure 13 Concatenated mirror configuration

Striped mirror

In a striped mirror configuration, the data occupies 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.

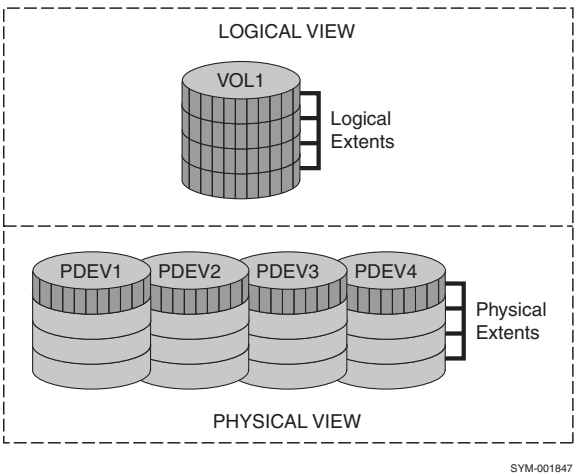


Figure 14 Striped mirror configuration

RAID 5 mirror

In a RAID 5 mirror configuration (for VxVM, Windows DiskADM, and LDM), the data occupies 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 improved. 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 is 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, DiskADM (`WINDISK.EXE`), the parity shifts across all four columns, starting from the left-most column.

For Veritas® VxVM and Windows 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.

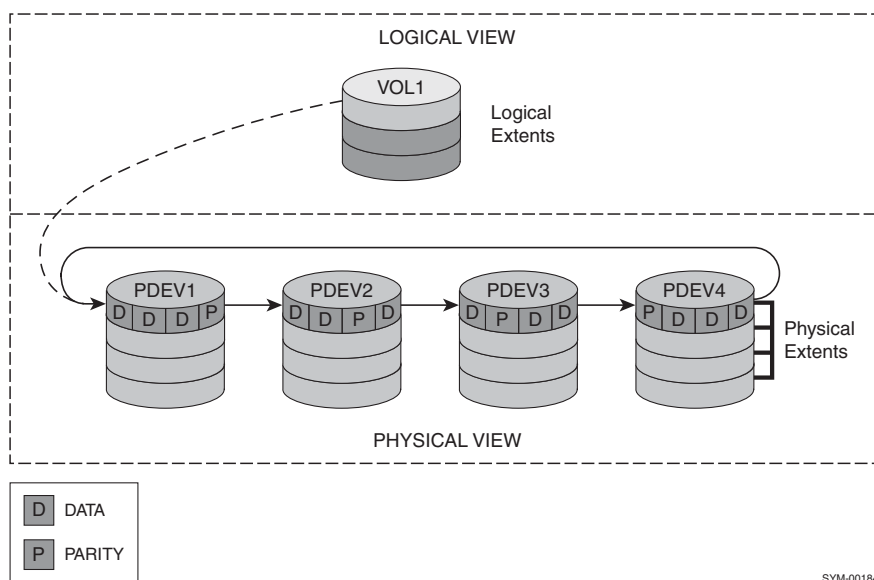


Figure 15 RAID 5 Veritas and Windows LDM configuration

Logical volume configurations

The logical volume configuration is determined by analyzing the configurations of its mirrors. Use the `sym1v show` command to view the configuration of a logical volume. [Table 32](#) lists the types of logical volume configuration.

Table 32 Logical volume configuration types

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

The mirror configuration type and the number of mirrors defines the logical volume configuration type as shown in [Table 33](#).

Table 33 Logical volume mirror configurations

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

Table 33 Logical volume mirror configurations (continued)

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

Mirror conditions

[Table 34](#) describes the types of mirror conditions.

Table 34 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

Use the `symvg` command to view volume groups and group detail on the host system.

Listing the volume groups

Use the following command to list all the volume group names on the host system:

```
symvg list
```

The command returns a list of volume group names, as the following example shows:

VOLUME GROUPS (HP-UX LVM) :

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

Use the following command to list the logical volume information for a specified volume group:

```
symvg show VgName
```

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

This command displays the following information about the group:

- ◆ 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

In addition, the command displays the following information about each device in the group:

- ◆ Physical device name
- ◆ VMAX array ID
- ◆ VMAX device name
- ◆ Device attributes
- ◆ Device status
- ◆ Device capacity in megabytes

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

Volume group types

[Table 35](#) lists the supported operating systems and their logical volume managers (volume group type).

Table 35 Volume group types

Operating system	Volume group type	VgType
HP-UX	HP-UX LVM HP-UX VxVM	HP_LVM HP_VXVM
IBM AIX	AIX VxVM AIX LVM	AIX_VXVM AIX_LVM
Oracle Solaris	Solaris VxVM Solstice Disk Suite ^a Oracle ASM Volume Manager ^b	SUN_VXVM SUN_SOLSTICE ORACLE_ASM
Windows	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. Add the `$SYMAPI_ASM_HOME/bin` to your user path to point to the path that contains the Oracle ASM Volume Manager binaries. Also, set `SYMCLI_ASM_CONNECT` to your username and password for database, `SYMAPI_ASM_HOME` to the location of the Oracle ASM binaries, and `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.

Example

The following lists details about the volume group `/dev/BigVG`:

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

```
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):
{
```

PdevName	SymID	Sym Dev	Att.	Sts	Cap (MB)
/dev/rdisk/c2t1d5	03003	0123	(M)	RW	21577

```
}
```

Volume group control operations

The `symvg` command provides the following control operations for volume groups:

`create` — Create a new volume group.

`destroy` — Delete a volume group.

`recover` — Recover a failed volume group.

`adddev` — Add devices to a volume group.

`rmdev` — Remove 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.

Creating a volume group

Use the following command to create a volume group:

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

Adding and removing devices

Use the following command to extend a volume group by adding devices:

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

To add more multiple devices separate their names with a space.

Use the following command to remove devices from a volume group:

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

To specify multiple devices separate their names with a space.

Destroying a volume group

Use the following command to delete a volume group:

```
symvg destroy VgName
```

Recovering a volume group

Use the following command to recover 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.

For example, to copy a volume group that spans several disks to another system:

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 the other host system. The import operation makes the volume group available to this new host system.

In addition to the `import` and `deport` operations, you can rescan volume groups. The `rescan` operation is currently supported only for the LDM volume groups and Veritas VxVM volume groups on Windows, Linux LVM volume groups, and HP LVM volume groups on HP-UX.

Control options

You can use the following options with the `import` and `deport` operations for volume groups.

Table 36 Volume group control options

Option	Action	Description
<code>-overwrite</code>	<code>deport</code>	Overwrite an existing mapfile (used with the <code>-mapfile</code> option).
<code>-cluster</code>	<code>import</code>	Imports Windows VxVM volume group as a cluster.
<code>-clear</code>	<code>import</code>	Import a volume group and clear the host ID on the volume group. This option is for Veritas volume managers only.

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

Translating volume groups to VMAX groups

The devices of an existing logical volume group can be translated and defined as a VMAX 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.

Options

You can use additional options to filter the devices added to the group. The **-sid** and **-RDFG** options can limit the devices to a specific VMAX array ID or SRDF RA group.

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

Use the **-force** option to do a partial add (that is, 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 fails because a device cannot be in more than one device or composite groups simultaneously.

Examples

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`, type:

```
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`, type:

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

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

```
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 `newcg` with the R1 and R1-BCV devices from the volume group named `thisvg`, type:

```
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`, type:

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

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

```
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.

Listing logical volumes

Use the following command to list all the logical volumes within a volume group:

```
symlv -g VgName list
```

The command displays the volume group name and the logical volume manager type, along with the name and attributes of each volume in the group.

For example, the following lists the logical volumes for volume group `/dev/BigVG`:

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

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

Name	Configuration	State	Cond	Num Mirrors	Num Log Extents
BigLV	Simple	Enabled	Sync	1	2500

Note: The above CLI command example is not available 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

Use the following command to list detailed information (including extent data) about a logical volume within a volume group:

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

For example, the following displays detailed information about logical volume `BigLV` in volume group `/dev/BigVG`:

```
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/rdisk/c2t1d5          1m (M)
      Sync           4m   03003 0123        5m /dev/rdisk/c2t1d5          5m (M)
      Sync           4m   03003 0123        9m /dev/rdisk/c2t1d5          9m (M)
      Sync           4m   03003 0123       13m /dev/rdisk/c2t1d5         13m (M)
      Sync           4m   03003 0123       17m /dev/rdisk/c2t1d5         17m (M)
... (etc. many more)
}

Mirror Physical Devices (3):
{
-----
      Array Dev      PPdevName                      PdevName                      (M)
-----
      03003 0123    /dev/rdisk/c2t1d5          /dev/rdisk/c2t1d5          (M)
      03003 0124    /dev/rdisk/c2t1d5          /dev/rdisk/c2t1d5          (m)
      03003 0125    /dev/rdisk/c2t1d5          /dev/rdisk/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 VMAX concatenated metadvice.

Condition, state, and attributes of a logical volume

The listings that the `symdev` command produces include information on the condition, state, and (in the case of a detailed listing) attributes of each logical volume. The following sections define the conditions and states that a logical volume can be in and list the attributes that a logical volume can have.

Logical volume conditions

A logical volume is in one of the nine states listed in [Table 37](#).

Table 37 Logical volume conditions

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 has a mirror where some physical extents are not consistent with corresponding extents in another mirror.
Sync	All physical extents in all mirrors of the logical volume are synchronized.
SyncInProgress	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 states that a logical volume can be in are:

- ◆ 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 attributes of a logical volume are:

- ◆ 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.

Control operations for logical volumes

The `symlv` command provides the following control operations for logical volumes:

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

`delete` — Delete a logical volume.

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

`remove` — Remove mirrors of a logical volume.

`extend` — Increase the size of a logical volume.

`reduce` — Decrease the size of a logical volume.

In addition, performance statistics for logical volumes is available as shown in [Chapter 6, “Statistics,”](#).

Creating a logical volume

Use the following command to create a logical volume:

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

Options

You can use the following options with the control commands for logical volumes

Table 38 Options for logical volume control operations

Option	Description
<code>-nmirror</code>	The number of mirrors in the logical volume.
<code>-mir</code>	The name of the mirror to remove.
<code>-striped RAID5</code>	The type of logical volume; striped or RAID 5.
<code>-ncols</code>	The number of stripe columns.
<code>-strsize</code>	The size of each stripe column in 512 byte blocks.
<code>-pd</code>	A list of the device names used for the operation.
<code>-type</code>	The volume group type (see Table 35 on page 103). The default <i>VgType</i> is assumed if no <i>VgType</i> is specified.

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

Example

The following creates a simple volume called `testlv` of size 500 MB in the volume group named `testvg`:

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

Deleting a logical volume

Use the following command to delete a logical volume:

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

For example, the following deletes a volume called `testlv` from the volume group named `testvg`:

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

Adding and removing mirrors

Use the following command to add a mirror image to a logical volume:

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

You can use the `-striped|RAID5`, `-ncols`, `-strsize`, and `-pd` options with this command (see [Table 38](#)).

Use the following command to remove a mirror from a logical volume:

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

You can also use the option `-mir` to the name the mirror to remove.

Extending and reducing logical volumes

Use the following command to extend (grow) a logical volume:

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

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

Use the following command to reduce the size of a logical volume:

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

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

Viewing extents

Use the following command to list the physical extents of each mirror in a logical volume:

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

If the VMAX device containing the physical extent is a metadvice, and the extent spans multiple metadvice members, the extent is broken into multiple extent definitions. Each definition describes the portion of the extent that resides on a unique Symmetrix metadvice member.

If the physical extent is part of a logical volume that does not reside on a VMAX or VMAX 3 device, the *Array* and *Dev* fields columns contain "N/A".

The output for the extents indicates the attributes of the device on which the extent resides. The meaning of the attributes are:

- ◆ (C) CLARiiON device
- ◆ (S) VMAX device
- ◆ (M) VMAX device metahead
- ◆ (m) VMAX device metamember

For example:

```
symmlv -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)
(...)
}
```

Extent conditions

Table 39 describes the possible conditions of a logical volume extent.

Table 39 Logical volume extent condition

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.

Expanded list

You can use the `-expand` option to obtain a detailed 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.

When using the `-expand` option for a mirror that is striped, the extent list includes a extent description of 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

You can use the `-no_extents` option to show the attributes of a logical volume mirror without details of the logical volume's extents.

```
symmlv -g VgName show LvolName -no_extents
```

For example:

```
symmlv -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/rdisk/c2t1d5    /dev/rdisk/c2t1d5    (M)
03003 0124   /dev/rdisk/c2t1d5    /dev/rdisk/c2t1d5    (m)
03003 0125   /dev/rdisk/c2t1d5    /dev/rdisk/c2t1d5    (m)
...
}
```

CHAPTER 6

Statistics

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

◆ Overview.....	118
◆ Monitoring data over time	118
◆ symhost stats	119
◆ symlv stats.....	121
◆ symrdb stats.....	122

Overview

The Solutions Enabler SYMCLI commands includes commands that provide statistical and performance information for various host-based objects. For example, you can obtain data relating to the host's CPU, memory, disks, logical volume manager, and databases. For detailed descriptions of those commands refer to the *EMXC Solutions Enabler CLI Command Reference*.

[Table 40](#) lists the commands that provide statistical and performance data.

Table 40 SRM statistics commands

Command	Argument	Actions
symhost	stats	Displays performance statistics.
symlv	stats	Displays performance statistics about logical volumes.
symrdb	stats	Displays performance statistics about the specified database.

Statistical and performance data are available for the following host operating systems:

- ◆ AIX
- ◆ HP-UX
- ◆ Linux
- ◆ Oracle Solaris
- ◆ Windows

Monitoring data over time

Each of the statistics commands provides two options that enable you to monitor a host resource over a period of time.

Table 41 Command line options for monitoring a resource over time

Option	Description
-c	Determines the number of times to run the command to display statistics.
-i	Determines the interval between each time the command runs, in seconds.

Note: if you supply the **-i** option without the **-c** option the command displays statistics indefinitely.

symhost stats

Use the **symhost stats** command to display any of the following metrics from the host:

- ◆ CPU
- ◆ Memory
- ◆ Disk
- ◆ CPU, disk, and memory (All)

You can use the **-type** option to specify the metric you require. [Table 42](#) shows the data returned for each of these metrics.

Table 42 Data returned by the syhost stats command

Value of the -type option	Data returned
CPU	CPU number/identifier Percentage busy in user mode Percentage busy in system mode Percentage idle time waiting for I/O to complete Number of interrupts per second Number of system calls per second Number of process context switches per second
MEMORY	Number of page in requests per second Number of pages paged in per second Number of page out requests per second Number of pages paged out per second Number of swap in requests per second Number of pages swapped in per second Number of swap out requests per second Number of pages swapped out per second
DISK	Disk name Number of read and write requests per second Number of read requests per second Number of write requests per second Number of kilobytes read and written per second Number of kilobytes read per second Number of kilobytes written per second Percentage of time the disk has been busy Percentage of time the wait queue has not been empty
ALL	This produces all of the items for CPU, memory, and disk.

CPU example

The following displays statistics about all host processors every 30 seconds for one hour:

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

```
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
```

Memory example

The following display statistics about memory usage on the host every 30 seconds for one hour:

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

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

Disk example

The following displays information about disks on the host every 30 seconds for one hour:

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

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

symlv stats

Use the **symlv stats** command to display the following statistics about logical volumes:

- ◆ Volume group name
- ◆ Logical volume name
- ◆ Number of read and write requests per second
- ◆ Number of read requests per second
- ◆ Number of write requests per second
- ◆ Number of kilobytes read and written per second
- ◆ Number of kilobytes read per second
- ◆ Number of kilobytes written per second
- ◆ Percentage of time the disk has been busy
- ◆ Percentage of time the wait queue was not empty

You can use the **-type** option to specify the type of volume group that you want information about. Statistics are available for the following types of volume groups:

Table 43 Volume group types

Operating system	Volume group type	Value of -type
HP-UX	HP-UX LVM HP-UX VxVM	HP_LVM HP_VXVM
IBM AIX	AIX VxVM	AIX_VXVM
Oracle Solaris	Solaris VxVM Solstice Disk Suite ^a	SUN_VXVM SUN_SOLSTICE
Windows	LDM VxVM	WIN_LDM WIN_VXVM
Linux	VxVM GFS Pool Linux LVM	LINUX_VXVM LINUX_POOL LINUX_LVM

- a. On the Solaris platform with SUN_SOLSTICE volume manager, a volume group named solstice cannot be created.

For example, the following displays the statistics about all logical volumes of VxVM on Oracle Solaris every 30 seconds for one hour:

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

symrdb stats

Use the **symrdb stats** command to display statistical data about any of the following types of database:

- ◆ Oracle
- ◆ SQL Server
- ◆ Sybase
- ◆ IBM DB2

You can use the **-type** option to specify the type of database and the **-target** option to specify the set of data you require. The following sections list the values you can use for the **-target** option for each type of database.

Oracle

[Table 44](#) lists the values that the **-target** option can have for an Oracle database and the statistics that each value generates:

Table 44 Metric options for Oracle database types

Value of -target	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

SQL Server

Table 45 lists the values that the **-target** option can have for a SQL Server database and the statistics that each value generates.

Table 45 Metric options for SQL Server databases

Value of -target	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/writethroughputforbackup/restoreofadatabase • 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

Sybase

Table 46 lists the values that the **-target** option can have for a Sybase database and the statistics that each value generates.

Table 46 Metric options for Sybase databases

Value of -target	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

IBM DB2

Table 47 lists the values that the **-target** option can have for a IBM DB2 database and the statistics that each value generates.

Table 47 Metric options for IBM DB2 databases

Value of -target	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 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 • 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 IBM DB2 metric options

Example

The following displays statistics about a session of an Oracle database every 30 seconds for one hour:

```
symrdb stats -type ORACLE -i 30 -c 120 -target SESSION
```


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 130
- ◆ Example 2: Mapping files and other disk storage objects 138
- ◆ Example 3: Displaying volume groups and logical volumes..... 150
- ◆ Example 4: Exporting and importing a volume group 155
- ◆ Example 5: Mapping files 157
- ◆ Example 6: Displaying a logical volume 160

Example 1: Displaying relational database objects

This example examines database objects for three different types of databases (Oracle, IBM DB2, and IBM Informix) and Microsoft Exchange

Examining Oracle database objects

This example uses an Oracle Solaris host running Oracle version 10.1 and connected to a local VMAX array.

Environment variables

The following environment variables enable Solutions Enabler to access the database:

Table 48 Environment variables to access a Oracle database

Environment Variable	Description	Example Value
SYMCLI_RDB_CONNECT	This variable defines the username and password of the system administrator. The variable is used to connect to the database.	system/manager
ORACLE_HOME	This variable defines the location of the Oracle binaries.	/opt/oracle/app/oracle
ORACLE_SID	The name of the Oracle database.	64817

symrdb list example

The **symrdb list** command lists the current Oracle database (64817) and allows testing of basic database connectivity.

```
symrdb list -type oracle
DATABASE NAMES (ORACLE 10.1.0.1.0):

Database Name
-----

64817
```

symrdb show example

The **symrdb show** command lists the basic attributes of a database.

```
symrdb show -type ORACLE
DB Name      State      Status      Size (mb)  Free (mb)
-----
64817        ONLINE     N/A         1315       75
```

symrdb list file example

The **symrdb list file** command identifies all the files of the current Oracle database.

```
symrdb list file -type oracle  
DATABASE FILE NAMES (ORACLE 10.1.0.1.0):
```

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

symrdb list table example

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 10.1.0.1.0):

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

```

symrdb list tbs example

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 10.1.0.1.0):

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 database objects

This example uses a Solaris host running IBM DB2 version 10.1 and connected to a local VMAX array.

Environment variables

The following environment variables enable Solutions Enabler to access the database:

Table 49 Environment variables to access an IBM DB2 database

Environment Variable	Description	Example Value
SYMCLI_RDB_CONNECT	This variable defines the username and password of the system administrator. The variable is used to connect to the database.	system/manager
LD_LIBRARY_PATH	This variable defines the location of the database client libraries.	/usr/db2v10in1/sql/lib/lib
DB2INSTANCE	The name of the DB2 database instance.	db2v10in1

symrdb list example

The **symrdb list** command lists the current databases (QU1 and SAMPLE) within the IBM DB2 database instance defined above (db2v10in1).

```
symrdb list -type ibmdb2
DATABASE NAMES (IBMDB2 10.1.0):
```

```
Database Name
-----
```

```
QU1
SAMPLE
```

symrdb list tbs example

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 ibmdb2 -db SAMPLE
DATABASE          : SAMPLE
TABLE SPACE NAMES (IBMDB2 10.1.0):
```

```
Table Space Name          Type          Status
-----
SYSCATSPACE               Permanent    Online
TEMPSPACE1                Temporary    Online
USERSPACE1                Permanent    Online
```

Examining IBM Informix database objects

This example uses an Oracle Solaris host running IBM Informix version 11.7 and connected to a local VMAX array.

Environment variables

The following environment variables enable Solutions Enabler to access the database:

Table 50 Environment variables to access a Oracle database

Environment Variable	Description	Example Value
SYMCLI_RDB_CONNECT	This variable defines the username and password of the system administrator. The variable is used to connect to the database.	informix/informix
INFORMIXDIR	This variable defines the location of the Informix binaries.	/usr/informix/11.7
ONCONFIG	This variable defines the name of the Informix configuration file	onconfig.test64117
INFORMIXSERVER	This variable defines the name of the default instance of the Informix database server.	test64117

symrdb list example

The **symrdb list** command lists current databases for the Informix server defined above (test64117).

```
symrdb list -type informix
```

```
DATABASE NAMES (INFORMIX 11.7.UD1):
```

```
Database Name
-----
```

```
sysmaster
sysutils
```

symrdb show file example

The **symrdb show file** command locates extent data on a sysmaster database file named `llog.dbs`. The data includes the physical devices where 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 when 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/test64117/llog.dbs
```

```
DATABASE           : sysmaster  
DATABASE FILE NAMES (INFORMIX 11.7.UD1):
```

```
Database File Name : /demofs/informix/test64117/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/test64117/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

```

The next part of this example shows that 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

This example uses a Windows host running Microsoft Exchange 2010 and connected to a local VMAX array.

Microsoft Exchange does not require setting the SYMCLI_RDB_CONNECT variable or any database environmental variables in order to access 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 2010):

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

This example uses an Oracle Solaris host connected to a VMAX array. The files are within a Solaris UFS file system and a Veritas VxVM logical volume manager manages a logical volume. In this example and all examples sections, an object is mapped to the VMAX device level (in some cases, to the VMAX metadvice level).

symhostfs list example

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 : Solaris UFS File
```

File Name	Size	Perms	Owner	Group
-----	----	-----	----	-----
file1	1m	-rw-r--r--	0	1
file1k	1k	-rw-r--r--	0	1

symhostfs show example

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 where the file is stored.

```
symhostfs show /fs_test1/file1 -expand -blocks
```

```
File Name           : /fs_test1/file1
File Type           : Solaris 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    : Thu 12-Apr-2012 16:16
Last Modification Time : Thu 12-Apr-2012 16:16
Last Status Change Time : Thu 12-Apr-2012 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 shows as not applicable (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, VMAX 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          : Solaris 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 VMAX device names (00A0, 00A4, 00A8, and 0009) correspond to host physical device names (PPdevName): /dev/vx/rdmp/c4t2d0s2, /c4t2d1s2, /c4t2d2s2, and /c5t0d1s2. The VMAX devices were configured arbitrarily as three metadevices and one simple VMAX 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 are different. Column "Array" provides the ID of the storage array on which the disk is located, in this case, VMAX 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): CLARiON Device.
 (S): Symmetrix Device.
 (M): Symmetrix Device Meta Head.
 (m): Symmetrix Device Meta member.
 (V): Virtual Disk on VMFS.

symrslv dir example

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 VMAX or VNX device, SRM cannot return Array, Dev, and Dev Offset so those columns show as not available (N/A) in the command output.

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.

```

symrslv file example 1

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 on the extent data at the VMAX device layer. The `symrslv` command goes directly to the VMAX devices (Dev) that stores the file and locates the extents there.

```

symrslv file /fs_test1/file1 -expand -blocks
Absolute Path                : /fs_test1/file1
Resolved Object Type         : Solaris 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.

```

symrslv file example 2

The **symrslv file** command with the **-collapse** option logically collapses the file's extent data. This 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 collapsed to form one extent.

```
symrslv file /fs_test1/file1 -collapse -blocks
```

```

Absolute Path           : /fs_test1/file1
Resolved Object Type    : Solaris 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.
```

symrslv file example 3

The **symrslv file** command with the **-phys_collapse** option physically collapses the same logically collapsed file from 27 extents to 22 extents. However, 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 cannot be logically collapsed, 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     : Solaris 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.

```

symrslv fs example

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

symrslv lv example 1

The **symrslv lv** command examines a logical volume named `striped` within a volume group named `fsvg`. The **-pdev_extents** option shows information about pdev-level extents only - without displaying any underlying metadvice 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    : Solaris 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          (S)
-----
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): CLARiON Device.
(S): Symmetrix Device.
(M): Symmetrix Device Meta Head.
(m): Symmetrix Device Meta member.
(V): Virtual Disk on VMFS.
```

symrslv lv example 2

The **symrslv lv** command examines the same logical volume (named `striped`) as the previous command, but this time displays (by default) the underlying metadvice configuration for each extent. An (M) at the end of a row of extent data indicates that this extent is located on a metadvice.

symrslv lv striped -g fsvg -blocks

```

Absolute Path           : /dev/vx/rdsk/fsvg/Striped
Resolved Object Type    : Solaris 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                      (M)
-----
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.

```

symrslv pd example

The **symrslv pd** command examines extents of one of the physical devices from the previous display. As shown in that display, device `/dev/rdisk/c4t2d0s2` is the metahead (M) of a VMAX metadvice that includes other metamembers (m). This display identifies those metamembers as VMAX devices 00A1, 00A2, and 00A3.

```
symrslv pd /dev/rdisk/c4t2d0s2 -blocks
```

```

    Absolute Path                : /dev/rdisk/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/rdisk/c4t2d0s2                0b (M)
      8837760b 00541 00A1        0b /dev/rdisk/c4t2d0s2          8837760b (m)
      8837760b 00541 00A2        0b /dev/rdisk/c4t2d0s2      17675520b (m)
      8835840b 00541 00A3        0b /dev/rdisk/c4t2d0s2      26513280b (m)
    }

    Mirror Physical Devices (4):
    {
-----
    Array Dev  PPdevName                      PdevName                      (M)
-----
    00541 00A0 /dev/rdisk/c4t2d0s2          /dev/rdisk/c4t2d0s2          (M)
    00541 00A1 /dev/rdisk/c4t2d0s2          /dev/rdisk/c4t2d0s2          (m)
    00541 00A2 /dev/rdisk/c4t2d0s2          /dev/rdisk/c4t2d0s2          (m)
    00541 00A3 /dev/rdisk/c4t2d0s2          /dev/rdisk/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.

```

Example 3: Displaying volume groups and logical volumes

This example uses an Oracle Solaris host connected to a VMAX array. The Solaris VxVM logical volume manager manages the volume groups and logical volumes.

symvg list example

The **symvg list** command displays logical volume groups defined for this host. The columns containing relate to information that is not applicable to a Veritas VxVM logical volume manager. However, HP-UX and AIX LVMs do use these columns.

symvg list

VOLUME GROUPS (Solaris VxVM):

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

symvg show example

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, VMAX device name, status (RW), and capacity in megabytes. The physical devices in this display are on VMAX array 00541, and three of the devices are metadevices as indicated by M in the Att. column.

symvg show nasty

Volume Group Name : nasty
Volume Group Type : Solaris 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):

```
{
-----
PdevName                               SymID  Sym  Dev  Att.  Sts      Cap
-----
/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.

symlv list example

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: Solaris VxVM
```

Name	Configuration	State	Cond	Num Mirrors	Num Log 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

symlv show example 1

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/rdisk/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 VMAX 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 39 on page 115](#).

```
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:...
```

symmlv show example 2

This **symmlv show** command displays a logical volume named `MirrorStripe`, which the LVM has configured as striped and mirrored (Striped Mir) with two logical volume mirrors.

symmlv -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 (VMAX devices 009B and 009D). The extent marked by an (m) is located on a device that is a member of a VMAX metadvice.

```
{
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 VMAX 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). This example uses an Oracle Solaris host connected to a VMAX array. The Veritas VxVM logical volume manager manages the volume groups and logical volumes.

symvg show example

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, VMAX device name, status (RW), and capacity in megabytes. The physical devices in this display are on VMAX array 03122.

symvg show testvg

```

Volume Group Name : testvg
Volume Group Type : Solaris 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):
{
-----
PdevName           Array  Dev  Att. Sts      Cap
(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.

```

deport and import example

The **symvg deport** command departs the metadata of volume group `testvg` from the system and stores that information in persistent storage. Some platforms may require the **-mapfile** option and a file name for a deport operation. That option is also needed by the 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):
{
-----
PdevName           Array  Dev  Att. Sts      Cap
(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

This example uses a Windows host connected to a CLARiiON array. The files displayed are within a Windows NTFS file system.

symrslv file example 1

The **symrslv** file command expands the extents of the file named `filetest` 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 VMAX device layer. The **symrslv** command drills down directly to the VMAX devices (Dev) on which the file is stored and locates the extents there.

```
symrslv file p:\clartest -expand
```

```

Absolute Path           : P:\FILETEST
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 (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0017	128m	\\.\PHYSICALDRIVE23	128m (S)
64k	00316	0018	128m	\\.\PHYSICALDRIVE24	128m (S)
64k	00316	0018	227m	\\.\PHYSICALDRIVE24	227m (S)
64k	00316	0017	227m	\\.\PHYSICALDRIVE23	227m (S)

```

64k  00316 0018      227m \\.\PHYSICALDRIVE24      227m (S)
64k  00316 0017      227m \\.\PHYSICALDRIVE23      227m (S)
64k  00316 0018      227m \\.\PHYSICALDRIVE24      227m (S)
64k  00316 0017      227m \\.\PHYSICALDRIVE23      227m (S)
64k  00316 0018      227m \\.\PHYSICALDRIVE24      227m (S)
}

```

Mirror Physical Devices (2):

```

{
-----
Array Dev    PPdevName                PdevName
-----
00316 0018   \\.\PHYSICALDRIVE24      \\.\PHYSICALDRIVE24      (S)
00316 0017   \\.\PHYSICALDRIVE23      \\.\PHYSICALDRIVE23      (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.

```

symrslv file example 2

The **symrslv file** command with the **-collapse** option logically collapses the same file's extent data. This 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 (S)
100m  00316 0017    128m \\.\PHYSICALDRIVE23      128m (S)
100m  00316 0018    128m \\.\PHYSICALDRIVE24      128m (V)
}

```

```
Mirror Physical Devices (2):
{
```

```
-----
Array Dev    PPdevName                PdevName
-----
00316 0018   \\.\PHYSICALDRIVE24          \\.\PHYSICALDRIVE24      (S)
00316 0017   \\.\PHYSICALDRIVE23          \\.\PHYSICALDRIVE23      (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 6: Displaying a logical volume

This example uses a Windows host connected to a VMAX storage array.

The **symlv show** command examines a logical volume named **P:** in the volume group **Bigdg**, 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 VMAX devices **0017** and **0018**. The extent condition is Sync (see [Table 39 on page 115](#) for other possible extent conditions).

symlv -g Bigdg 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                      (S)
-----
00316 0017  \\.\PHYSICALDRIVE23             \\.\PHYSICALDRIVE23             (S)
00316 0018  \\.\PHYSICALDRIVE24             \\.\PHYSICALDRIVE24             (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.

