

**EMC Solutions Enabler
Symmetrix SRM CLI**
Version 6.0

PRODUCT GUIDE
P/N 300-000-883
REV A06

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.emc.com

Copyright © 2002, 2003, 2004, 2005 EMC Corporation. All Rights Reserved.

Printed January, 2005

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, replication, or distribution of any EMC software described in this publication requires an applicable software license.

Trademark Information

EMC², EMC, Symmetrix, Celerra, CLARiiON, CLARAlert, Documentum, HighRoad, Legato, Navisphere, PowerPath, ResourcePak, SnapView/IP, SRDF, TimeFinder, VisualSAN, and where information lives are registered trademarks and EMC Automated Networked Storage, EMC ControlCenter, EMC Developers Program, EMC OnCourse, EMC Proven, EMC Snap, Access Logix, AutoAdvice, Automated Resource Manager, AutoSwap, AVALONidm, C-Clip, Celerra Replicator, Centera, CentraStar, CLARevent, Connectrix, CopyCross, CopyPoint, DatabaseXtender, Direct Matrix, Direct Matrix Architecture, EDM, E-Lab, Enginuity, FarPoint, FLARE, GeoSpan, InfoMover, MirrorView, NetWin, OnAlert, OpenScale, Powerlink, PowerVolume, RepliCare, SafeLine, SAN Architect, SAN Copy, SAN Manager, SDMS, SnapSure, SnapView, StorageScope, SupportMate, SymmAPI, SymmEnabler, Symmetrix DMX, Universal Data Tone, and VisualSRM are trademarks of EMC Corporation. All other trademarks used herein are the property of their respective owners.

Preface	xi
----------------------	----

Chapter 1 Introduction

Storage Resource Management (SRM) Overview	1-2
Data-Object SRM.....	1-3
Database SRM	1-4
File-System SRM.....	1-5
Logical Volume SRM.....	1-6
SRM Statistics	1-8

Chapter 2 Data Object SRM

Overview	2-2
Data Object Terminology and Architecture	2-2
Examining Device Partitions.....	2-4
Examining Device Data Objects.....	2-5
Examining Logical Volume Data Objects	2-7
Examining File System Data Objects	2-9

Chapter 3 Database SRM

Overview	3-2
Database Architecture and Terminology.....	3-2
Establishing the Database Connection.....	3-16
Setting UNIX Environment Variables.....	3-16
Connecting to the Database.....	3-17
Database Startup Options.....	3-17
Database Shutdown Options	3-20
SRM Database Daemons.....	3-23

What Are Daemons?	3-23
Why Use Daemons?	3-24
Using SRM Database Daemons	3-24
Listing Database Instances	3-27
Examining Database Files	3-28
Examining Database File Attributes	3-28
Listing Tablespace Files	3-30
Translating Database Devices to Symmetrix Groups	3-31
Examining Tablespaces	3-32
Examining Tablespace Attributes	3-32
Listing Tablespace Files	3-34
Listing Tablespace Tables	3-34
Listing Segments in a Tablespace	3-35
Translating Tablespace Devices to Symmetrix Groups	3-36
Examining Schemas	3-37
Examining Schema Attributes	3-37
Listing Schema Files	3-39
Listing Schema Tables	3-39
Listing Segments in a Schema	3-40
Examining Tables	3-41
Examining Table Attributes	3-41
Examining Segments	3-43
Examining Segment Attributes	3-44
Database I/O Control	3-45
Freezing the Database	3-45
Thawing the Database	3-45
Hot Backup Control	3-46
Checkpoint	3-46
Archive Log	3-46
SQL Server Snapshot	3-46
Using the EMC Oracle ASM Library	3-48

Chapter 4 **File System SRM**

Overview	4-2
Terminology	4-2
Finding File Systems	4-3
Examining a File System	4-4
Finding Directories and Files	4-6
Examining Directories	4-7
Listing Files	4-8
Examining Files	4-9

Chapter 5 **Logical Volume SRM**

Overview	5-2
LVM Terminology	5-2
Mirror Configurations	5-5
Logical Volume Configurations	5-9
Mirror Conditions	5-10
Viewing Volume Groups	5-11
Listing the Volume Groups	5-11
Volume Group Details	5-11
Volume Group Control Operations	5-15
Creating a Volume Group	5-15
Adding and Removing Devices	5-15
Destroying a Volume Group	5-16
Recovering a Volume Group	5-16
Importing and Exporting Operations	5-16
Translating Volume Groups to Symmetrix Groups	5-18
Viewing Logical Volumes	5-19
Listing Logical Volumes	5-19
Logical Volume Details	5-19
Logical Volume Conditions	5-21
Logical Volume States	5-21
Logical Volume Attributes	5-21
Logical Volume Control Operations	5-22
Creating a Logical Volume	5-22
Viewing Extents	5-25
Extent Conditions	5-26
Expanded List	5-26
No Extents in List	5-27

Chapter 6 **SRM Statistics**

Retrieving Statistics	6-2
Supported Metrics	6-2

Index	i-1
-------------	-----

2-1	Objects, Extents, and Data Blocks	2-3
3-1	Oracle Database Architecture	3-4
3-2	Segments, Extents, and Data Blocks	3-5
3-3	SQL Database Architecture	3-7
3-4	Sybase Database Architecture	3-9
3-5	Informix Database Architecture	3-10
3-6	DB2/UDB Database Architecture	3-12
3-7	Exchange Database Architecture	3-14
3-8	System View of the Daemon Process	3-23
3-9	Database Daemon Libraries for Communication	3-25
5-1	Volume Group	5-4
5-2	Simple Mirror Configuration	5-5
5-3	Concatenated Mirror Configuration	5-6
5-4	Striped Mirror Configuration	5-7
5-5	RAID 5 VERITAS and Windows 2000 LDM Configuration	5-8

1-1	Data Object SRM Commands	1-3
1-2	Database Object SRM Commands	1-4
1-3	File System SRM Command	1-5
1-4	Logical Volume SRM Commands	1-6
1-5	SRM Statistics Command	1-8
3-1	UNIX Database Environment Variables	3-16
3-2	SRM Database Daemon Names	3-26
5-1	Logical Volume Configuration Types	5-9
5-2	Logical Volume Mirror Configurations	5-10
5-3	Mirror Condition Descriptions	5-10
5-4	Volume Group Types	5-13
5-5	Volume Group Control Options	5-16
5-6	Logical Volume Condition Descriptions	5-21
5-7	Options for Logical Volume Control Operations	5-23
5-8	Logical Volume Extent Conditions	5-26
6-1	SRM Statistics Commands	6-2
6-2	Metric Options for Oracle Database Type	6-7
6-3	Metric Options for SQL Server Databases	6-8
6-4	Metric Options for Sybase Databases	6-9
6-5	Metric Options for IBMUDB Databases	6-10

As part of its effort to continuously improve and enhance the performance and capabilities of the EMC product line, EMC periodically releases new versions of both the EMC Enginuity Operating Environment and Solutions Enabler. Therefore, some functions described in this guide may not be supported by all versions of Enginuity or Solutions Enabler currently in use. For the most up-to-date information on product features, see your product release notes.

If a Solutions Enabler feature does not function properly or does not function as described in this guide, please contact the EMC Customer Support Center for assistance.

Audience This manual provides both guide and reference information for command-line users and script programmers that focus on examining the Storage Resource Management (SRM) information related to various data objects and data handling facilities within your host system.

Organization The following defines the structure of this manual:

Chapter 1, *Introduction*, describes the SYMCLI Storage Resource Management (SRM) components that focus on the mapping of various open system objects.

Chapter 2, *Data Object SRM*, describes the host system mapping to both Symmetrix-stored data objects and objects stored on a host disk. The chapter describes how the extents of data files and objects can be examined using SYMCLI commands.

Chapter 3, *Database SRM*, describes the host system mapping to various installed relational databases, and how the databases and their structure can be examined using SYMCLI commands.

Chapter 4, *File System SRM*, describes the host system mapping of its file systems and how the file systems and their structure can be examined using SYMCLI commands.

Chapter 5, *Logical Volume SRM*, describes the host system mapping to Symmetrix logical volumes and how logical volumes and their groups can be examined using SYMCLI commands.

Chapter 6, *SRM Statistics*, describes the SRM performance statistics for a host's CPU, disk, and memory.

Note: Detailed manpage descriptions of all SYMCLI commands, environment variables, option file parameters, and error codes can now be found in the companion *EMC Solutions Enabler Symmetrix CLI Command Reference*.

Related Documentation

Other Symmetrix publications of related interest are:

- ◆ *EMC Solutions Enabler Support Matrix*
- ◆ *EMC Solutions Enabler Symmetrix CLI Command Reference*
- ◆ *EMC Solutions Enabler Installation Guide*
- ◆ *EMC Solutions Enabler Symmetrix Base Management CLI Product Guide*
- ◆ *EMC Solutions Enabler Symmetrix Device Masking CLI Product Guide*

Conventions Used in This Manual

The following conventions are used in this manual:

In this manual, every use of the word SYMCLI means EMC Solutions Enabler SYMCLI.

Every occurrence of the word MVS in text or in symbolic syntax means OS/390 and z/OS.

Every occurrence of the word OSF1 in text or in symbolic syntax means Tru64 UNIX.

Note: A note calls attention to any item of information that may be of special importance to the reader.



CAUTION

A caution contains information essential to avoid damage or degraded integrity to storage of your data. The caution might also apply to protection of your software or hardware.

Typographical Conventions

This manual uses the following type style conventions in this guide:

bold text	Boldface text provides extra emphasis and emphasizes warnings, and specifies window names and menu items in text.
<i>italic text</i>	<i>Italic text and characters emphasizes new terms, identifies variables in a software syntax (non-literal notation), identifies unique word usage, and applies emphasis in examples and in references to book titles and sections.</i>
fixed space courier font	A fixed space font identifies files and pathnames, and is used in command line entries, displayed text, or program listings.

Where to Get Help

EMC software products are supported directly by the EMC Customer Support Center.

Obtain technical support by calling the EMC Customer Support Center at one of the following numbers:

United States: (800) 782-4362 (SVC-4EMC)

Canada: (800) 543-4782 (543-4SVC)

Worldwide: (508) 497-7901

Language services are available upon request.

Sales and Customer Service Contacts

For the list of EMC sales locations, please access the EMC home page at:

<http://www.emc.com/contact/>

For additional information on the EMC products and services available to customers and partners, refer to the EMC Powerlink website at:

<http://powerlink.emc.com>

Your Comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Please e-mail us at **techpub_comments@emc.com** to let us know about your opinion or any errors concerning this manual.

This chapter provides an overview of the SYMCLI Storage Resource Management (SRM) component that focuses on the open system tasks of mapping and examining various objects on your host system, their relational databases, and the attached storage arrays.

- ◆ Storage Resource Management (SRM) Overview 1-2
- ◆ Data-Object SRM 1-3
- ◆ Database SRM 1-4
- ◆ File-System SRM 1-5
- ◆ Logical Volume SRM 1-6
- ◆ SRM Statistics 1-8

Storage Resource Management (SRM) Overview

The Storage Resource Management (SRM) component, described in this manual, extends the basic SYMCLI command set to include SRM commands that allow you to systematically find and examine attributes of various objects on your host, within your specific relational database, or in the EMC® storage enterprise.

The SYMCLI commands support SRM in the following areas:

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

For a list of all supported platforms and versions, refer to the *EMC Solutions Enabler Support Matrix*.

Data-Object SRM

SYMCLI data-object SRM allows you to examine the mapping of storage devices and the characteristics of data files and objects. These commands allow you to examine the extents of data files and objects, and how they are mapped on storage devices.

Table 1-1 lists the SYMCLI commands you can use to examine the mapping of data objects.

Table 1-1 Data Object SRM Commands

Command	Argument	Displays
symrslv	pd	Logical-to-physical mapping information about any physical device.
	lv	Logical-to-physical mapping information about a logical volume.
	file	Logical-to-physical mapping information about a file.
	dir	Logical-to-physical mapping information about a directory.
	fs	Logical-to-physical mapping information about a file system.
sympart	show	Partition information of a host device.

For more information about data-object mapping, refer to Chapter 2. For more information about the `symrslv` command, refer to *EMC Solutions Enabler Symmetrix CLI Command Reference*.

Database SRM

SYMCLI database SRM commands allow you to examine the host database mapping and the characteristics of a database. The commands provide listings and attributes that describe various databases, their structures, files, tablespaces, and user schemas.

Table 1-2 lists the SYMCLI database object mapping commands.

Table 1-2 Database Object SRM Commands

Command	Argument	Actions
symrdb	list	Lists various physical and logical database objects: • Current relational database instances available • Tablespaces, tables, files, or schemas of a database • Files, segments, or tables of a database tablespace or schema
	show	Shows information about a database object: • Tablespace, tables, file, or schema of a database • File, segment, or a table of a specified tablespace or schema
	stats	Displays performance statistics for databases.
	rdb2dg rdb2cg	Translates the devices of a specified database into a device group or a composite group.
	tbs2dg tbs2cg	Translates the devices of a specified tablespace into a device group or a composite group.
	startup	Provides startup options for a database manager instance.
	shutdown	Provides shutdown options for a database manager instance.

Table 1-2 Database Object SRM Commands (continued)

Command	Argument	Actions
symioctl	freeze	Freezes I/O to a specified database application.
	thaw	Thaws I/O to a specified database application.
	begin backup	Places objects into hot backup mode.
	end backup	Takes objects out of hot backup mode.
	checkpoint	Issues a checkpoint to the RDBMS.
	archive log	Archives the current log.
	begin snapshot	Begins a snapshot backup on SQLServer 2000 and higher.
	end snapshot	Saves snapshot metadata and resumes writes on SQLServer 2000 and higher.
	restore snapshot	Restores previously saved snapshot metadata on SQLServer 2000 and higher.
	abort snapshot	Terminates the snapshot without saving metadata and resumes writes on SQLServer 2000 and higher.

For information about database mapping, refer to Chapter 3. For more information about the `symrdb` and `symioctl` commands, refer to *EMC Solutions Enabler Symmetrix CLI Command Reference*.

File-System SRM

The SYMCLI file-system SRM command allows you to investigate the file systems that are in use on your operating system. The command provides listings and attributes that describe file systems, directories, and files and their mapping to physical devices and extents.

Table 1-3 lists the SYMCLI command you can use to examine the file-system mapping.

Table 1-3 File System SRM Command

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

For more information about file-system mapping, refer to Chapter 4. For more information about the `symhostfs` command, refer to *EMC Solutions Enabler Symmetrix CLI Command Reference*.

Logical Volume SRM

SYMCLI logical volume SRM commands allow you to control and map logical volumes to display a detailed view of the underlying storage devices. Logical volume architecture defined by a Logical Volume Manager (LVM) is a means for advanced applications to improve performance by the strategic placement of data.

Table 1-4 lists the SYMCLI commands you can use to examine the logical volume mapping.

Table 1-4 Logical Volume SRM Commands

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

Table 1-4 Logical Volume SRM Commands (continued)

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

For more information about logical volume mapping, refer to Chapter 5. For more information about the `symlv` and `symvg` commands, refer to *EMC Solutions Enabler Symmetrix CLI Command Reference*.

SRM Statistics

SYMCLI SRM performance statistics commands allow you to retrieve statistics about a host's CPU, disk, memory, logical volumes, and databases.

Table 1-5 lists the statistics commands.

Table 1-5 SRM Statistics Command

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

For more information about statistics, refer to Chapter 6. For more information about the `symhost`, `symlv`, and `symrdb` commands, refer to *EMC Solutions Enabler Symmetrix CLI Command Reference*.

This chapter describes the host system Storage Resource Management (SRM). It explains how the extents of data files and objects can be examined using SYMCLI commands.

- ◆ Overview2-2
- ◆ Examining Device Data Objects.....2-5
- ◆ Examining Logical Volume Data Objects2-7
- ◆ Examining File System Data Objects.....2-9

Overview

SYMCLI data-object SRM commands allow you to examine the extents of data files and objects and partition information of a physical device.

The SRM commands are supported on the following operating systems:

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

Data Object Terminology and Architecture

This section describes data-object terminology and Storage Resource Management (SRM) architecture.

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

A *data object* is an entity such as a file, a logical volume, a file system, or a physical device used to store or back up data. A data object consists of a set of extents (see Figure 2-1). Within a Symmetrix[®] system, an object can have extents in various devices.

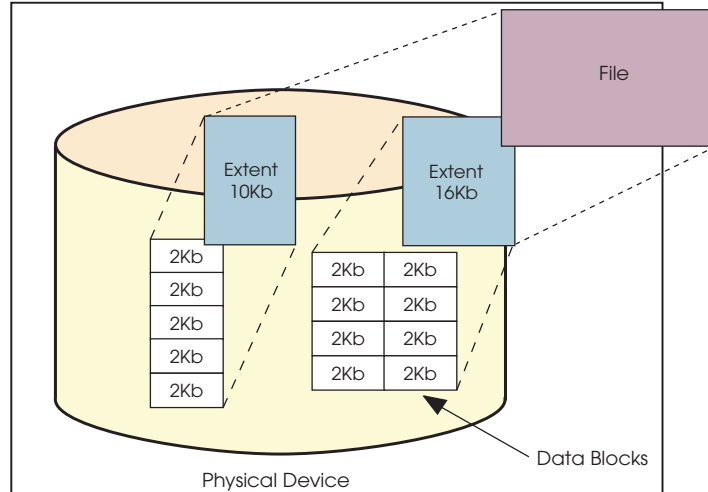


Figure 2-1 Objects, Extents, and Data Blocks

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

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

Examining Device Partitions

Using `sympart`, you can find information about a host device's partitions.

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

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

To examine the partitions for a host physical device, enter:

```
sympart show PdevName
```

The following is a sample command and output:

```
# sympart show c0t0d0s0

Raw Partition Name      : /dev/rdisk/c0t0d0s0
Full Device Name       : /dev/rdisk/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

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	

Examining Device Data Objects

Using `symrslv`, you can find information about data objects down to the physical-extent level.

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

Note: `sympd` can be used to determine which Symmetrix or CLARiiON® devices you need to examine.

To examine a host physical device, enter:

```
symrslv pd PdevName
```

The following is a sample command and output:

```
# symrslv pd /dev/rdsk/c2t0d0

Absolute Path           : /dev/rdsk/c2t0d0
Resolved Object Type    : Physical Device
Resolved Object Size    : 4315m
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 (1):
-----
Size   Array Dev   Offset PPdevName           Offset
-----
4315m  03003 0077    0b    /dev/rdsk/c2t0d0        0b (S)

Mirror Physical Devices (1):
-----
Array Dev   PPdevName           PdevName
-----
03003 0077  /dev/rdsk/c2t0d0    /dev/rdsk/c2t0d0        (S)
}
```

You can apply the `-expand` option to display multiple contiguous extents as separate extents. The default behavior is to logically collapse contiguous extents into one large extent.

You can show information about the object without the details of extent information, using the `-no_extents` option.

You can also apply the `-phys_collapse` option to display physically contiguous extents as one larger extent. The list of extents returned are not logically consistent.

The `-pdev_extents` option will display extents at the physical device (`pdev`) level. It will not expand extents at the underlying meta device level.

Examining Logical Volume Data Objects

Using `symrslv`, you can find the data mapping attributes and examine the extents of a specified logical volume (of the LVM).

Note: `symvg` and `symlv` can be used to determine which logical volume you need to examine.

To examine a logical volume, enter:

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

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

```
# symrslv -g BigVG lv BigLV
```

The following is sample output from this command:

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

```
Number of Mirrors for object (1):
```

```
{
1) Mirror Configuration
```

```
Mirror Physical Extents (3):
```

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

```
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)
}
```

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

You can apply the `-expand` option to display multiple contiguous extents as separate extents. The default behavior is to collapse contiguous extents into one large extent.

You can show information about the object without the details of extent information, using the `-no_extents` option.

You can also apply the `-phys_collapse` option to display physically contiguous extents as one larger extent. The list of extents returned are not logically consistent.

The `-pdev_extents` option will display extents at the physical device (`pdev`) level. It will not expand extents at the underlying meta device level.

For more information about logical volumes, see Chapter 5, *Logical Volume SRM*.

Examining File System Data Objects

Using `symrslv`, you can find the data mapping attributes and examine the extents of a data object specified as a file system or a regular data file.

Note: `symhostfs` can be used to determine which file you need to examine.

To examine a file system, enter:

```
symrslv fs FileSystemName
```

To examine a data file, enter:

```
symrslv file FileName
```

To examine a directory, enter:

```
symrslv dir Directory
```

The command lists the following file attributes:

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

You can apply the `-expand` option to display multiple contiguous extents as separate extents. The default behavior is to collapse contiguous extents into one large extent.

You can also apply the `-phys_collapse` option to display physically contiguous extents as one larger extent. The list of extents returned are not logically consistent.

You can show information about the object without the details of extent information, using the `-no_extents` option.

The `-pdev_extents` option will display extents at the physical device (`pdev`) level. It will not expand extents at the underlying meta device level.

You can also use `symhostfs show` to display object information. For more information about file systems, see Chapter 4, *File System SRM*.

This chapter describes the host system Storage Resource Management (SRM) to various installed relational databases, and how the databases and their structure can be examined using SYMCLI commands.

◆ Overview	3-2
◆ Establishing the Database Connection.....	3-16
◆ SRM Database Daemons.....	3-23
◆ Listing Database Instances.....	3-27
◆ Examining Database Files.....	3-28
◆ Translating Database Devices to Symmetrix Groups	3-31
◆ Examining Tablespaces	3-32
◆ Translating Tablespace Devices to Symmetrix Groups	3-36
◆ Examining Schemas.....	3-37
◆ Examining Tables	3-41
◆ Examining Segments	3-43
◆ Database I/O Control.....	3-45
◆ Using the EMC Oracle ASM Library.....	3-48

Overview

SYMCLI database SRM commands allow you to examine the host database mapping and the characteristics of a database.

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

Database Architecture and Terminology

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

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

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

Oracle Architecture

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

Typical Oracle logical schema objects are:

- ◆ Tables
- ◆ Indexes

- ◆ Views
- ◆ Clusters
- ◆ Database links

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

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

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

- ◆ Contain the impact of any damaged data
- ◆ Back up and recover each partition independently
- ◆ Balance the I/O load by mapping partitions to different disk drives

A *tablespace* is a named storage pool that physically allocates space for the database files. As shown in Figure 3-1, 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 data files that are associated with the corresponding tablespace.

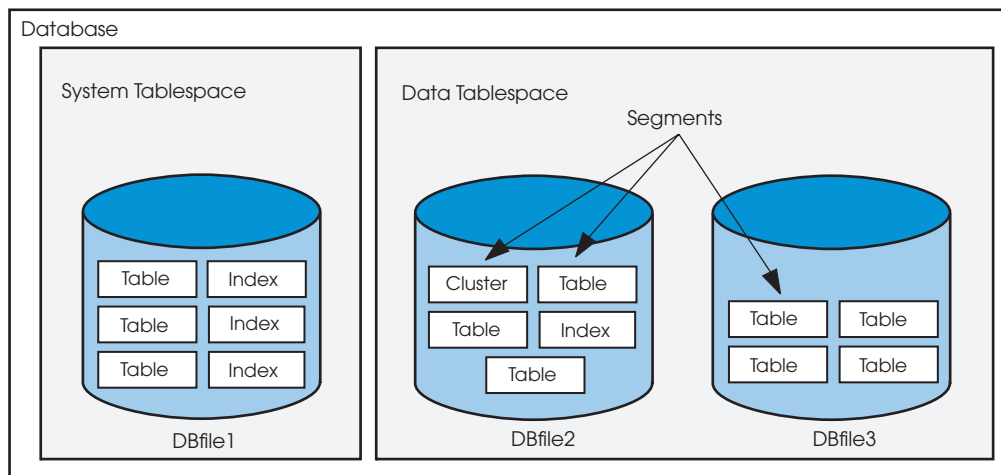


Figure 3-1 Oracle Database Architecture

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

A *segment* is a set of extents that contains all the data blocks for a specific logical storage structure or object within a tablespace (see Figure 3-2). For example, one or more extents are allocated to each table and to each associated index. A segment and all its extents are stored in one tablespace. Within a tablespace, a segment can span data files or have extents with data from more than one file. There are four basic types of segments:

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

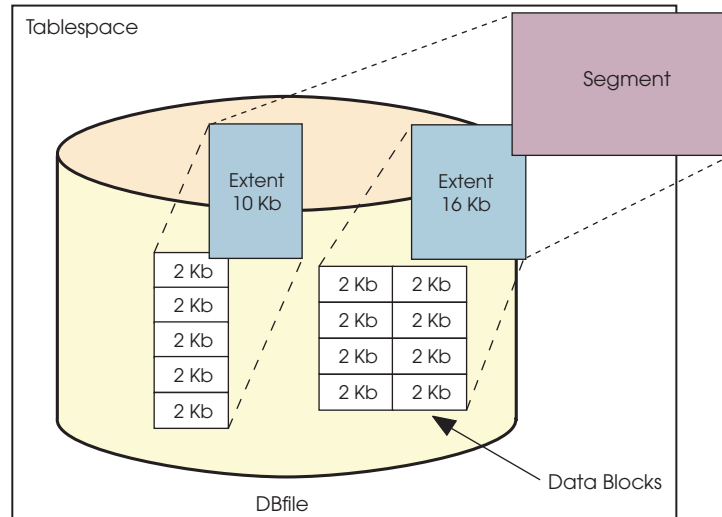


Figure 3-2 Segments, Extents, and Data Blocks

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

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

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

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

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

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

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

SQL Server Architecture

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

Typical SQL Server logical database objects are:

- ◆ Tables
- ◆ Indexes
- ◆ Views

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

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

In SQL Server, an *index* is an optional structure associated with a table that increases the data retrieval performance. Indexes are created on one or more columns of a table. Indexes are useful when an

application often needs to make queries to a table for a range of rows or a specific row. Indexes are logically and physically independent of the table data and can, therefore, be dropped at any time.

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

A *filegroup* is a named storage pool that physically allocates space for the database files. As shown in Figure 3-3, 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 data files that are associated with the corresponding filegroups.

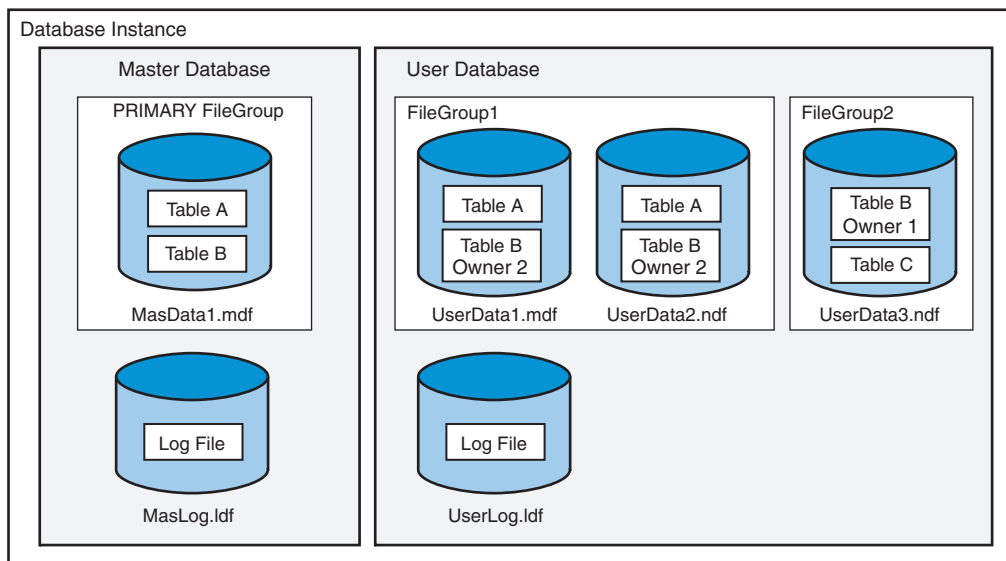


Figure 3-3 SQL Database Architecture

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

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

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

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

Sybase Architecture

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

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

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

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

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

As shown in Figure 3-4, one or more table and index structures make up the device of a database. The data is stored logically in databases and physically in devices that are associated with the corresponding database.

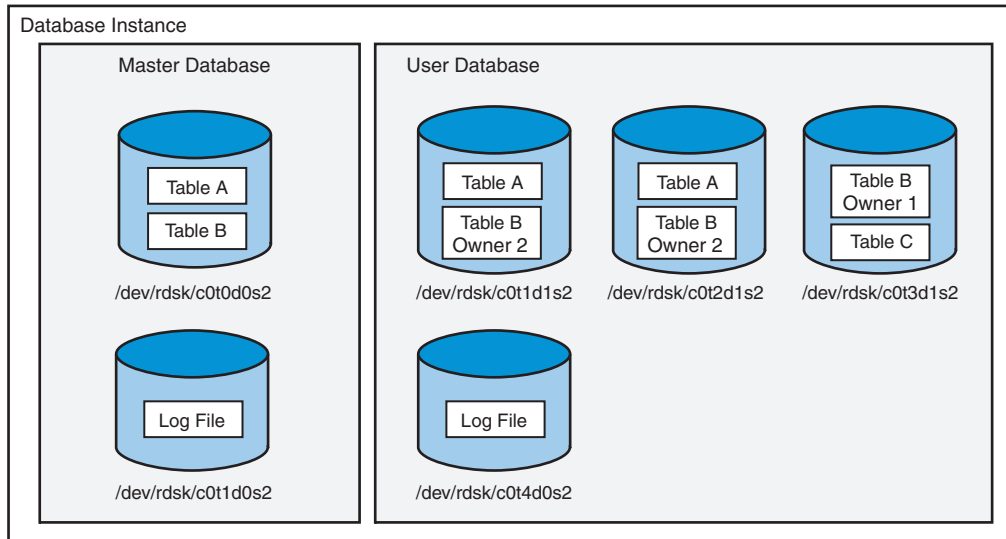


Figure 3-4 Sybase Database Architecture

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

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

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

Informix Architecture

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

In Informix, a *tablespace* is a named storage pool that physically allocates space for the database *chunks* (datafiles). A tablespace can be defined by one or more chunks. As shown in Figure 3-5, one or more table and index structures make up the chunks of tablespace. The data is stored logically in tablespaces and physically in chunks that are associated with the corresponding tablespaces.

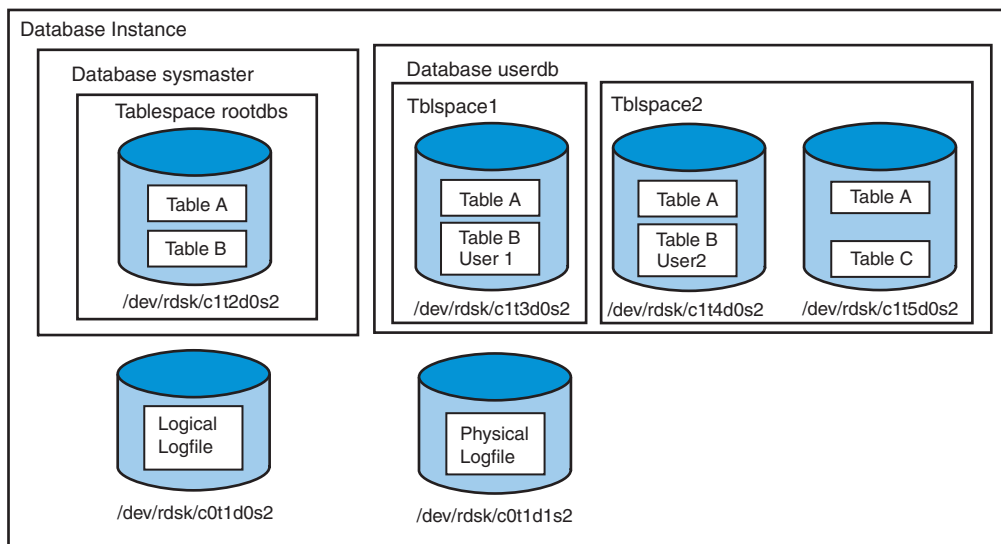


Figure 3-5 Informix Database Architecture

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

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

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

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

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

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

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

IBM DB2/UDB Architecture

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

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

Typical DB2/UDB logical database objects are:

- ◆ Tables
- ◆ Indexes
- ◆ Views
- ◆ Functions
- ◆ LOBs

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

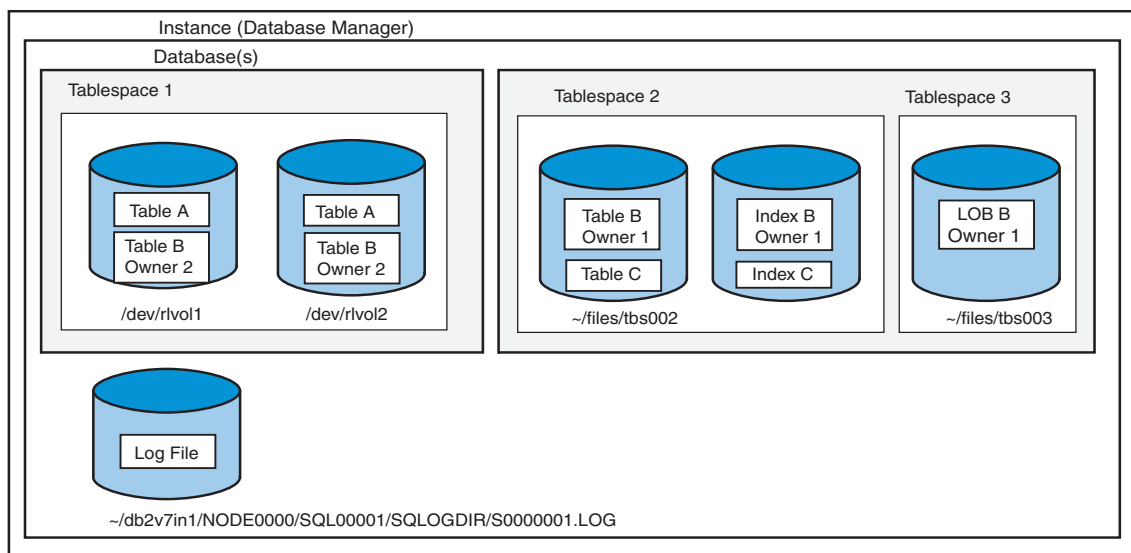


Figure 3-6 DB2/UDB Database Architecture

Each database must contain at least three tablespaces:

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

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

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

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

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

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

Exchange Architecture

Exchange 2000 Server allows two types of databases, a mailbox store and a public store. An Exchange *mailbox store* is a binary tree that is consists entirely of electronic mail data. This data is logically, but not physically grouped into mailboxes. The actual mailbox objects and associated metadata are maintained in the Windows 2000 Active Directory, where they are stored as attributes on a Windows 2000 user object. An Exchange *public store* is almost identical to the mailbox store, except that all of the data is organized logically into a folder hierarchy instead of mailboxes.

From the top down, as shown in Figure 3-7, an Exchange 2000 Server is a single *instance* of Exchange consisting of storage groups and databases. Each Exchange 2000 Server is allowed a maximum of four storage groups. Within SYMCLI, an Exchange 2000 storage group will be referred to as an Exchange 2000 tablespace. Each Exchange 2000 storage group consists of anywhere from one to five databases and a set of transaction logs. All of the databases in a storage group share the same set of transaction logs. All data is written to the transaction logs first, and then placed in memory, and finally committed to the data files on disk. A checkpoint file (whose name is a variation of `E0n.chk`, where *n* is from 0 to 3) is maintained for each storage group to track which transactions have been committed to their respective databases, and which have not been committed.

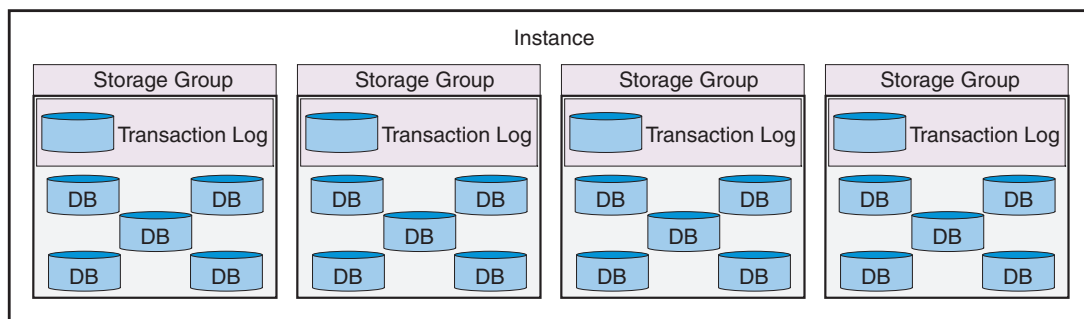


Figure 3-7 Exchange Database Architecture

Different databases in different storage groups can have the same name. To make Exchange database names unique within SYMCLI, they should be in the form *storagegroup\database*. For example, if storage group SG1 and SG2 both have a database DB1, the database names in SYMCLI should be SG1\DB1 and SG2\DB1, respectively.

Because the storage group name is implied in the database name, commands that pass both the database name (using the `-db` option) and the storage group name (using the `-tbs` option) will return as invalid. An Exchange 2000 database consists of two data files, a binary data file, and a streaming file. The binary data file has an extension of `.edb` and is laid out as a binary tree. The streaming file has an extension of `.stm` and is laid out flat, in 64 KB runs. The binary data file is the traditional data file that stores the majority of typical e-mail data. The streaming file is a new file designed to store native Internet content in large, sequential chunks. The sequential layout of the streaming file allows for faster retrieval of Internet content such as movies, sound files, and pictures.

Exchange 2000 databases can be brought online and offline independent of the other databases on the server. This process is referred to as *mounting* and *dismounting* the database. The Solutions Enabler SYMCLI provides control functions for the mounting and dismounting of databases.

EMC Solutions Enabler treats both the transaction log files and the database data files as database files. Be aware of this when querying the Exchange 2000 Server for the files in a given database. Each file returned will be tagged with its appropriate type, either data or log.

There is no concept of the following database objects within Exchange 2000: a schema, a table, or a segment. The concept of both an extent and a data block is maintained through the underlying file system, NTFS, rather than Exchange 2000 itself. Exchange 2000 data blocks are always 4 KB.

SharePoint Portal Server

SharePoint Portal Server is a Web portal server that integrates document management and search capabilities into a single server product. Because SharePoint Portal Server runs on top of Exchange 2000 Server, the architecture for SharePoint Portal Server is identical to that of Exchange 2000 Server. Specifically, SharePoint Portal Server 2001 is restricted to using a single storage group and a single public store.

Establishing the Database Connection

Setting UNIX Environment Variables

Before you begin any SYMCLI database command session, you must define UNIX environment variables on your host system as listed in Table 3-1.

Table 3-1 UNIX Database Environment Variables

For Database	Set Variables
ALL	LD_LIBRARY_PATH ^a LD_LIBRARY_PATH_64 ^b SHLIB_PATH ^c LIBPATH ^d
Oracle	ORACLE_HOME ORACLE_SID PATH
Informix Dynamic Server	INFORMIXDIR ONCONFIG INFORMIXSERVER PATH
Sybase	SYBASE SYBASE_ASE ^e DSQUERY ^f PATH SYBASE_OCS
IBM DB2/UDB	DB2INSTANCE

- a. Set for Solaris and Tru64 UNIX and point to the DB client libraries.
- b. Set for Solaris 64 bit and point to the DB client libraries (only if you need to override LD_LIBRARY_PATH).
- c. Set for HP-UX and point to the DB client libraries.
- d. Set for AIX and point to the DB client libraries.
- e. Set for Sybase startup/shutdown DB; when set during daemon autostart, all other applications that use the daemon must also set the SYBASE_ASE environment variable.
- f. Must point to the correct Sybase server, i.e., Sybase Monitoring server or Sybase Adaptive server. **For statistics, it must point to the Sybase Monitoring server.**

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

Connecting to the Database

You must establish a database connection for access with SYMCLI database commands by setting SYMCLI_RDB_CONNECT. If you are working with one type of database, you can set SYMCLI_RDB_TYPE to save keystrokes with a set of commands.

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

If the connection is remote, you must 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 allow the user to start up and shut down the specified database server manager instance. With the SYMCLI client/server capability, the user is able to start up or shut down the database manager instance on the client side. This eliminates the limitation of most databases, which only allow startup and shutdown of their database manager instance on the database server side.

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

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

Sybase The following options are available for Sybase databases:

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

Where:

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

-m—Starts the database in single user mode.

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

Oracle The following startup options are available for Oracle databases:

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

Where:

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

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

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

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

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

MOUNT — Mounts the database but does not open it.

OPEN — Mounts and opens database.

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

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

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

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

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

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

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

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

Where:

ADDNODE — Issues the ADD NODE command.

RESTART — Issues the RESTART DATABASE command.

STANDALONE — Start the node in STANDALONE mode.

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

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

profile — Specifies the name of the profile.

node — Specifies the node number.

hostname — Specifies the system name.

port — Specifies the port number.

netname — Specifies the net name.

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

computer — Specifies the computer name.

SQLServer The following startup options are available for SQLServer databases:

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

Where:

instance — Instance name to be started.

master_file_path — Master database file.

error_log_path — Error log file.

master_log_path — Master database log file.

virtual_addr_space — Amount of virtual address space in megabytes.

trace_number — Trace number.

Database Shutdown Options

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

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

Sybase The following shutdown options are available for Sybase databases:

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

Where:

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

wait — Brings the server down slowly.

nowait — Shuts the server down immediately.

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

Oracle The following shutdown options are available for Oracle databases:

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

Where:

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

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

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

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

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

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

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

The shutdown options for IBM DB2/UDB:

FORCE — Issues the FORCE APPLICATION (ALL) command.

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

DROP_ACT — Identifies an initial call.

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

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

profile — Specifies the name of the profile.

node — Specifies the node number.

SQLServer The following shutdown options are available for SQLServer databases:

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

Where:

instance — Specifies the instance name to be shut down.

SRM Database Daemons

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

What Are Daemons?

A daemon is a process, or service, that helps your 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 3-8. A single daemon can support connections to multiple instances/databases.

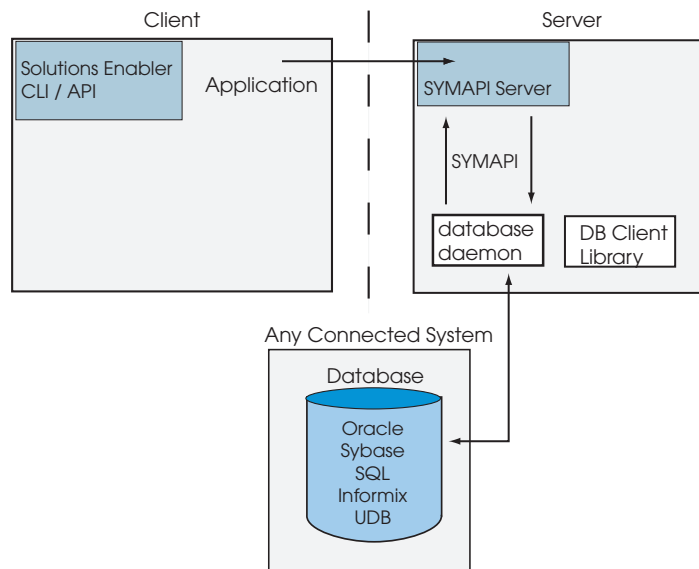


Figure 3-8 System View of the Daemon Process

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.

Using SRM Database Daemons

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

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

The user can start and stop a daemon at any time. If an application is already connected via 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 you shut down the database server while using a daemon, you must restart the daemon to reconnect to the server.

You can start a daemon when the system is booted, and stop it when the system is shut down. You can use the `stordaeomon` command to start, stop, or query the status of SRM daemons. The `stordaeomon start` command requires that you specify the name of the daemon. For example, to start the SRM daemon for an Oracle database, you enter the following:

```
stordaeomon start stororad
```

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

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

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

It is necessary to set the `LD_LIBRARY_PATH` variable, and its counterpart environment variable, for all database daemons except Oracle and SQL Server. See Table 3-1 on page 3-16 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 3-9. These libraries can be 32-bit or 64-bit, however both libraries must use the same bit size.

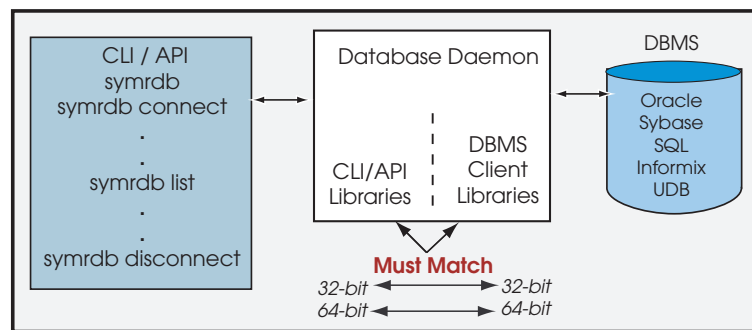


Figure 3-9 Database Daemon Libraries for Communication

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

Table 3-2 lists the set of database daemons.

Table 3-2 SRM Database Daemon Names

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

Refer to the *EMC Solutions Enabler Installation Guide* and the database daemon man pages in the *EMC Solutions Enabler Symmetrix CLI Command Reference* for more information about installing and using daemons.

Listing Database Instances

You can list the available databases (database instances) with the following command:

```
symrdb list -type Oracle
```

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

```
DATABASE NAMES (ORACLE 8.0.6.0.0):  
  
Database Name  
-----  
  
64806
```

Sample output from this database is used throughout this chapter.

Examining Database Files

You can identify all the files of a specified database with the following command:

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

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

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

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

Examining Database File Attributes

You can examine the attributes and the extent information of a specific file in a specified database with the following command:

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

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

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

The following is sample output from this command:

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

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

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

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

Database Allocated Block Size : 10m

Database Free Block Size     : 9m

Database File Unallocated Block Size : N/A
```

```

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

```

Number of Mirrors for object (1):

```
{
```

1) Mirror Configuration

Mirror Physical Extents (69):

```
{
```

```
-----
```

Size	Array	Dev	Offset	PPdevName	Offset
64k	N/A	N/A	N/A	/dev/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

```
...
```

```
}
```

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

You can expand the extent information displayed in the list with the `-expand` option.

You can show information about the object without the details of extent information, using the `-no_extents` option.

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

Listing Tablespace Files

You can list all the files of a specified tablespace and database with the following command:

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

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

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

Translating Database Devices to Symmetrix Groups

You can translate the devices of an existing database and define them to be a Symmetrix device group or composite group.

DB Devices to Device Groups

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

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

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

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

DB Devices to Composite Groups

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

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

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

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

You can also add virtual devices to a device group or composite group by using the `-vdev` option.

Examining Tablespaces

You can identify all the tablespaces of a specified database with the following command:

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

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

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

The following is sample output from this command:

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

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

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

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

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

Examining Tablespace Attributes

You can examine the attributes of a tablespace of a specified database with the following command:

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

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

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

The following is sample output from this command:

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

```
Database Table Space Name : SYSTEM
```

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

```
Number of Host Files (1):
```

```
{
```

```
1) Database File Name: /usr/oracle/oradata/64806/system01.dbf
```

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

```
Number of Mirrors for object (1):
```

```
{
```

```
1) Mirror Configuration
```

```
Mirror Physical Extents (417):
```

```
{
```

```
-----
```

Size	Array	Dev	Offset	PPdevName	Offset
64k	N/A	N/A	N/A	/dev/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

```
...
```

```
}
```

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

You can expand the extent information displayed in the list with the `-expand` option.

You can show information about the object without the details of extent information, using the `-no_extents` option.

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

Listing Tablespace Files

You can list all the files of a specified tablespace and database with the following command:

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

If you have set the connection environment variables (for any database type) described in Table 3-2, you only need to enter:

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

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

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

Listing Tablespace Tables

You can list the tables of a specified tablespace with the following command:

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

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

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

The following is sample output from this command:

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

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

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

Listing Segments in a Tablespace

You can list the segments of a specified tablespace. A segment (storage unit) exists for every data table, partition, and every index.

To list the segments of a tablespace, enter:

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

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

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

The following is sample output from this command:

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

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

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

Translating Tablespace Devices to Symmetrix Groups

You can translate the devices of an existing tablespace and define them to be a Symmetrix device group or composite group.

TBS Devices to Device Groups

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

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

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

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

TBS Devices to Composite Groups

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

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

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

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

You can also add virtual devices to a device group or composite group by using the `-vdev` option.

Examining Schemas

You can identify all the schemas of a specified database with the following command:

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

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

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

The following is sample output from this command:

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

Schema Name
-----

SYS
SYSTEM
DBSNMP
```

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

Examining Schema Attributes

You can examine the attributes of a schema of a specified database with the following command:

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

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

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

The following is sample output from this command:

```
Database Schema Name : SYSTEM
```

```
Number of Host Files (1):
```

```
{
```

```
1) Database File Name: /usr/oracle/oradata/64806/system01.dbf
```

```
Absolute Path           : /usr/oracle/oradata/64806/system01.dbf
Resolved Object Type     : SunOS UFS File
Resolved Object Size     : 64m
Number of Trailing Bytes : 0
Extent byte offset to data : 0
```

```

Number of Physical Devices      : 1
File System Mount Point        : /
File System Device Name        : /dev/dsk/c0t1d0s0

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

Mirror Physical Extents (417):
{
-----
      Size  Array Dev      Offset PPdevName                      Offset
-----
      64k   N/A   N/A      N/A   /dev/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/rdisk/c0t1d0s0          /dev/rdisk/c0t1d0s2
...
}

```

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

You can expand the extent information displayed in the list with the `-expand` option.

You can show information about the object without the details of extent information, using the `-no_extents` option.

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

Listing Schema Files

You can list all the files of a specified schema with the following command:

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

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

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

Listing Schema Tables

You can list the tables of a specified schema with the following command:

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

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

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

Listing Segments in a Schema

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

To list the segments in a schema, enter:

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

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

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

The following is sample output from this command:

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

Segment Name	Owner	Type
-----	-----	-----
AQ\$_QUEUES_CHECK	SYSTEM	Index
AQ\$_QUEUES_PRIMARY	SYSTEM	Index
AQ\$_QUEUE_TABLES_PRIMARY	SYSTEM	Index
AQ\$_SCHEDULES_CHECK	SYSTEM	Index
AQ\$_SCHEDULES_PRIMARY	SYSTEM	Index

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

Examining Tables

You can identify all the tables of a specified tablespace or schema with the following command:

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

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

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

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

Examining Table Attributes

You can examine the attributes of a table of a specified tablespace or schema with the following command:

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

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

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

You can expand the extent information displayed in the list with the `-expand` option.

You can show information about the object without the details of extent information, using the `-no_extents` option.

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

Examining Segments

You can identify all the segments of a specified tablespace or schema with the following command:

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

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

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

The following is sample output from this command:

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

Segment Name -----	Owner -----	Type -----
1.527	SYS	Cache
C_COBJ#	SYS	Cluster
SYS_C00512	SYS	Index
SYS_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

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

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

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

Examining Segment Attributes

You can examine the attributes of a segment of a specified tablespace or schema with the following command:

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

The command shows the attributes about the segment as follows:

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

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

Database I/O Control

Using the `symioctl` command, you can invoke I/O control operations to a specified relational database or database object(s).

Note: First, you must set `SYMCLI_RDB_CONNECT` to your username and password for access to the specified database.

Freezing the Database

You can freeze all I/O access to a specified relational database using the following command:

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

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

If you have set the connection environment variables described in Table 3-2, you only need to enter:

```
symioctl freeze Object Object
```

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

```
symioctl freeze HR Payroll
```

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

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

Thawing the Database

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

```
symioctl thaw
```

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

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

Hot Backup Control

For Oracle only, you can perform hot backup control on a list of tablespace objects, which must be performed before and after a freeze/thaw command. The steps required to split a group of BCV devices 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 your environment.
4. Issue the `symioctl thaw` command.
5. Issue the `symioctl end backup` command.

Checkpoint

You can request the relational database to perform a checkpoint as follows:

```
symioctl checkpoint
```

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

Archive Log

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

SQL Server Snapshot

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

The snapshot arguments for the `symioctl` command extend the freeze/thaw functionality to allow both physical and logical restore of an SQL Server database. The snapshot backup will save metadata

about the database in a file, which will be needed for a subsequent snapshot restore.

The TimeFinder split should be preceded by the following command:

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

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

The TimeFinder split should be followed by this command:

```
symioctl end snapshot Object
```

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

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

```
symioctl abort snapshot Object
```

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

The TimeFinder restore should be followed by this command:

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

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

Using the EMC Oracle ASM Library

The Oracle10g introduces a storage management and provisioning feature called Automatic Storage Management (ASM). ASM provides filesystem and volume manager capabilities built into the Oracle database kernel. These capabilities work with the EMC Solutions Enabler libraries.

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

```
SetDesc:= EMC:Expression
Expression:= Term | Expression 'OR' Term
Term      := Factor | Term 'AND' Factor
Factor:= '(' Expression ')' |
        'NOT' Factor |
        ScalarAttribute Relop Integer-Constant |
        ScalarAttribute Relop
        Integer-Constant Multiplier |
        StringAttribute '==' Character-String
ScalarAttribute:= 'SIZE' | 'size' | 'BLKSZ' | 'blksz' |
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 Oracle user.

Some examples follow:

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

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

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

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

Usage of CLI `asmdscvr`

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

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

The `asmdscvr` utility provides the following:

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

Where:

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

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

`-h` — Prints this message

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

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

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

NOTES:

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

EXAMPLES:

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

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

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

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

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

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

This chapter describes the host file system Storage Resource Management (SRM) and how the file systems and their structure can be examined using SYMCLI commands.

◆ Overview	4-2
◆ Finding File Systems.....	4-3
◆ Examining a File System	4-4
◆ Finding Directories and Files	4-6
◆ Examining Files	4-9

Overview

SYMCLI file-system SRM commands allow you to investigate the file systems that are in use by your operating system. The commands provide listings and attributes that describe file systems, directories, and files, and their mapping to physical devices and extents.

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

- ◆ HP-UX [hfs, VxFS (jfs)]
- ◆ IBM AIX (jfs), AIX 5.1 (jfs, jfs2), VxFS
- ◆ IBM OS/390
- ◆ Linux ext2, ext3, VxFS
- ◆ Sun Solaris (ufs, VxFS)
- ◆ Tru64 UNIX (ufs, AdvFS)
- ◆ Windows NT and Windows 2000, 2003 (NTFS, FAT32)

Terminology

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

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

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

Finding File Systems

You can view the file systems that are on your host system with the following command:

```
symhostfs list
```

The following is sample output:

Mounted File Systems

Device Name -----	Type -----	Mount Point -----
/dev/vg00/lvol3	HP VxFS	/
/dev/vg00/lvol1	HFS	/stand
/dev/vg00/lvol8	HP VxFS	/var
/dev/vg00/lvol7	HP VxFS	/usr
/dev/vg00/lvol4	HP VxFS	/tmp
api193:(pid787)	Unknown Type	/net
api40:/usr3/ftpusers/outgoing	NFS	/apikits

The command lists the following attributes for each file system:

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

Examining a File System

You can examine the attributes of a specified file system with the following command:

```
symhostfs show ObjectName
```

To specify the file system you are examining, use the file-system mount point, as shown here:

```
# symhostfs show /stand
```

The following is a sample of the output from this command:

```
File System Mount Point      : /stand

File System Type             : HFS
Mount Options                : READWRITE ALLOW_SUID QUOTA_OFF

File System Device Name      : /dev/vg00/lvol1

Size                         : 82m
Number Free Blocks           : 117778

Fragment Size                : 1024 bytes
Block Size                   : 8192 bytes

Number Inodes                 : 13440
Number Free Inodes           : 13403

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

Absolute Path                 : /stand
Resolved Object Type          : File System
Resolved Object Size          : 82m
Number of Trailing Bytes      : 0
Extent byte offset to data    : 0
Number of Physical Devices    : 1
File System Mount Point       : /stand
File System Device Name       : /dev/vg00/lvol1

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

Mirror Physical Extents (1):
{
```

Number of Mirrors for object (1):

1) Mirror Configuration

Mirror Physical Extents (1):

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

Mirror Physical Devices (1):

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

}

This output shows that the file system with mount point `/stand` is mounted on logical volume `lv011`, which in turn is created from the underlying device `c0t3d0`.

Using options, you can expand or collapse the returned extent information.

Finding Directories and Files

You can list the subdirectories of any root or parent directory with the following command:

```
symhostfs list -dir dirname
```

The following shows a sample command and output:

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

```
Directory Name : /stand
```

Directory Name	Type	Perms	Owner	Group
-----	-----	-----	-----	-----
lost+found	HFS	drwxr-xr-x	0	0
build	HFS	drwxr-xr-x	0	3
dlkm	HFS	drwxr-xr-x	0	3
dlkm.vmunix.prev	HFS	drwxr-xr-x	0	3

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

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

```
...
```

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

```
Directory Name : /stand/build
```

Directory Name	Type	Perms	Owner	Group
-----	-----	-----	-----	-----
mod_wk.d	HFS	drwxr-xr-x	0	3

```
Directory Name : /stand/build/mod_wk.d
```

```
Directory Name : /stand/dlkm
```

Directory Name	Type	Perms	Owner	Group
-----	-----	-----	-----	-----
system.d	HFS	drwxr-xr-x	0	3
node.d	HFS	drwxr-xr-x	0	3
mod.d	HFS	drwxr-xr-x	0	3

```
...
```

Examining Directories

You can examine the attributes of a directory with the following command:

```
symhostfs show dirname
```

The following is a sample command and output:

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

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

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

Number of Symbolic Links : 2

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

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

Listing Files

You can list all the files of a directory with the following command:

```
symhostfs list -file dirname
```

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

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

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

File Name	Size	Perms	Owner	Group
-----	----	-----	-----	-----
ioconfig	5k	-rw-r--r--	0	3
bootconf	1b	-rw-r--r--	0	3
system	2k	-r--r--r--	0	3
vmunix	11m	-rwxr-xr-x	0	3
kernrel	1b	-r--r--r--	0	3
rootconf	1b	-rw-----	0	0
system.prev	2k	-r--r--r--	0	3
vmunix.prev	11m	-rwxr-xr-x	0	3

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

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

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

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

Examining Files

You can examine the attributes of a file with the following command:

```
symhostfs show filename
```

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

```
# symhostfs show /stand/system

...

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

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

Number of Symbolic Links : 1

Last Access Time    : Mon 17-Mar-2003 16:20
Last Modification Time : Mon 02-Dec-2002 15:12
Last Status Change Time : Mon 02-Dec-2002 15:12

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

Number of Extents in File : (1)
{
  -----
  Extent   Offset in   Extent   Offset
  Type     Device     Size     in File
  -----
  Data      121k      2k        0b
}

Absolute Path       : /stand/system
Resolved Object Type : HP-UX HFS File
Resolved Object Size : 2k
Number of Trailing Bytes : 49
Extent byte offset to data : 0
Number of Physical Devices : 1
File System Mount Point : /stand
File System Device Name : /dev/vg00/lvol1
```

```

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

Mirror Physical Extents (1):
{
-----
      Size  Array Dev      Offset PPdevName                      Offset
-----
      2k    N/A  N/A      N/A  /dev/rdisk/c0t3d0                3m
}

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

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

```

Using options, you can restrict the returned information to just data extents or just metadata extents and/or expand the extent information.

You can restrict the listing to just metadata objects or just data objects with the `-meta` | `-data` options.

You can expand the listing of extents for the file by specifying `-expand` (`-collapse` is the default).

You can show information about the object without the details of extent information using the `-no_extents` option.

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

Logical Volume SRM

This chapter describes the host system logical volume Storage Resource Management (SRM) to Symmetrix devices and how logical volumes and their groups can be examined using SYMCLI commands.

- ◆ Overview5-2
- ◆ Viewing Volume Groups.....5-11
- ◆ Volume Group Control Operations5-15
- ◆ Translating Volume Groups to Symmetrix Groups.....5-18
- ◆ Viewing Logical Volumes5-19
- ◆ Logical Volume Control Operations.....5-22
- ◆ Viewing Extents.....5-25

Overview

SYMCLI logical volume SRM commands allow you to map 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.

The following logical volume managers are currently supported by SYMCLI:

- ◆ HP LVM on the HP-UX platform
- ◆ AIX LVM on the IBM AIX platform
- ◆ VERITAS VxVM on the SunOS (Solaris), HP-UX, AIX, Windows 2000, and Linux platforms
- ◆ EMC PowerPath® Volume Manager on the SunOS (Solaris), HP-UX, and AIX platforms
- ◆ Solstice Disk Suite on the SunOS (Solaris) platform
- ◆ Sequent Volume Manager (ptx/SVM) on the DYNIX/ptx platform
- ◆ Logical Storage Manager (LSM) on the Tru64 UNIX platform
- ◆ Disk Administrator (DiskADM) on the Windows NT platform
- ◆ Logical Disk Manager (LDM) on the Windows 2000, 2003 platforms
- ◆ AS400 on the AS400 platform

LVM Terminology

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

Prior to Windows 2000, logical volumes were addressed using drive letters (A-Z), not the physical device name (\\.\PHYSICALDRIVE*). In Windows 2000, Microsoft introduced a significant new storage feature which removes the 26-volume limitation of the drive letter naming convention. *Volume mount points* are NTFS directories that can be associated with specific logical volumes in a persistent

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

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

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

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

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

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

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

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

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

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

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

Figure 5-1 illustrates a volume group that contains a logical volume (VOL1) whose logical extents are mapped to the physical devices PDEV1 and PDEV2.

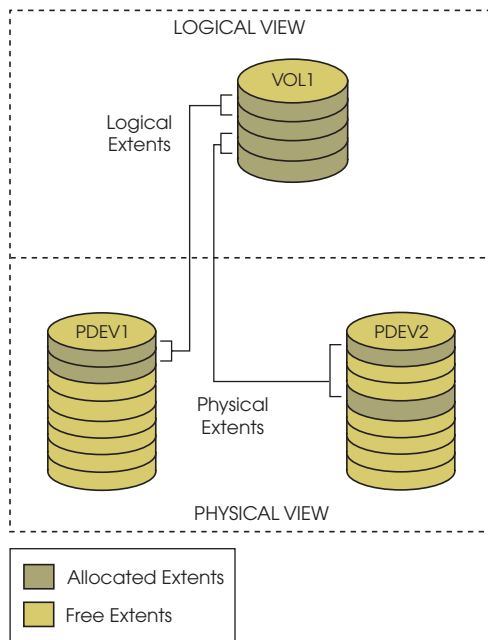


Figure 5-1 Volume Group

Mirror Configurations

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

The following are the basic mirror configurations:

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

Simple Configuration

A simple mirror configuration implies that all physical extents are on the same physical device. In Figure 5-2, all physical extents are on one physical device, PDEV1.

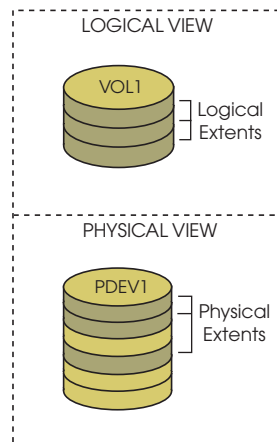


Figure 5-2 Simple Mirror Configuration

Concatenated Configuration

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

In Figure 5-3, the logical extents on VOL1 point to the physical extents on PDEV1 and PDEV2.

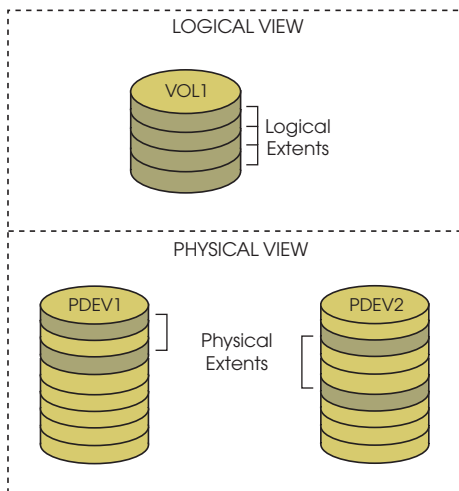


Figure 5-3 Concatenated Mirror Configuration

Striped Configuration

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

In Figure 5-4, VOL1 is a striped logical volume whose logical extent data is interleaved across multiple physical extents on PDEV1, PDEV2, PDEV3, and PDEV4.

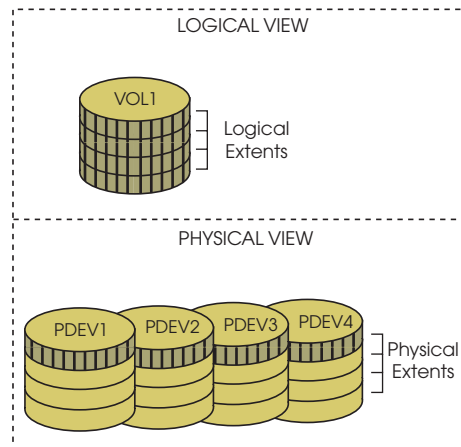


Figure 5-4 Striped Mirror Configuration

RAID 5 Configuration

In a RAID 5 mirror configuration (for VxVM, Windows NT DiskADM, and LDM), the data is written across multiple devices, with multiple stripes of data and one stripe of parity data being written in sets across all devices. The stripe containing the parity data is shifted across each device in the group. By spreading out the parity evenly across all devices, I/O performance is increased. In the case of a disk failure, a stripe's data can be reconstructed using the parity algorithm against the data in the available stripes.

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

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

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

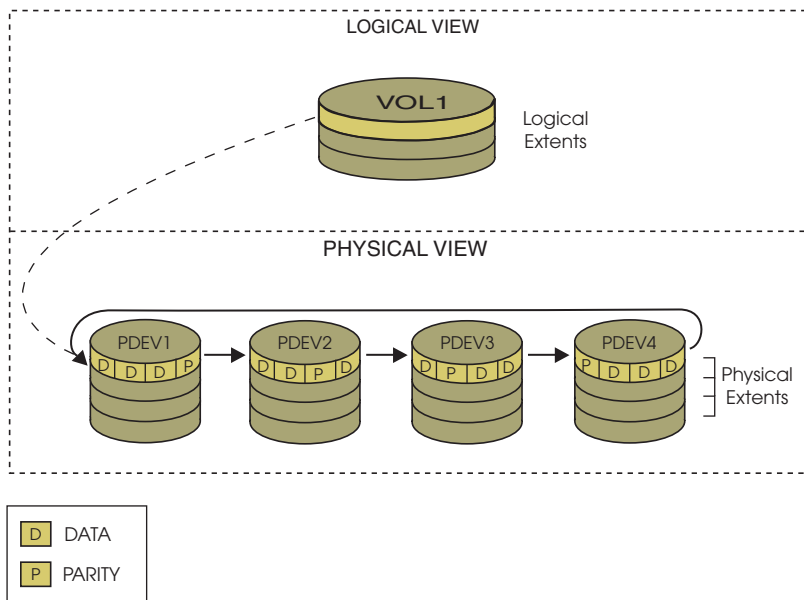


Figure 5-5 RAID 5 VERITAS and Windows 2000 LDM Configuration

Logical Volume Configurations

The logical volume configuration is determined by analyzing the configurations of its mirrors. The logical volume configurations can be viewed using the `symlv show` command. The possible logical volume configuration types are listed in Table 5-1.

Table 5-1 Logical Volume Configuration Types

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

Given the mirror configuration type and the number of mirrors, the logical volume configuration type is determined as shown in Table 5-2.

Table 5-2 Logical Volume Mirror Configurations

Logical Volume Configuration Type	Mirror Configuration	Number of Mirrors
Mirror Simple	Simple	2 or more
Mirror Concatenated	Concatenated	2 or more
Mirror Striped	Striped	2 or more
Mirror Mixed	Varied	2 or more
Simple	Simple	1
Concatenated	Concatenated	1
Striped	Striped	1
RAID5	RAID5	1 with parity data

Mirror Conditions

Table 5-3 describes the types of mirror conditions.

Table 5-3 Mirror Condition Descriptions

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

Viewing Volume Groups

With `symvg`, you can find all the volume groups, and view the details about each volume group currently defined by the logical volume manager on your host system.

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

Listing the Volume Groups

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

```
symvg list
```

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

VOLUME GROUPS (HP-UX LVM):

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 NT Disk Administrator, the task of establishing a volume group does not exist. For this volume type, a volume group name of `rootnt` is returned. This corresponds to other LVM manager default groups of `rootvg` or `rootdg`.

Volume Group Details

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

```
symvg show VgName
```

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

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

The following is sample output from this command:

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

```
Volume Group State           : Enabled
```

```
Volume Group Attributes      : N/A
```

```
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

```
}
```

This command provides the following information:

- ◆ Volume group name
- ◆ Logical volume manager type
- ◆ Volume group state
- ◆ Volume group attributes
- ◆ Physical extent size
- ◆ Maximum number of logical and physical devices allowed
- ◆ Actual number of logical and physical devices in the group
- ◆ Number and names of the physical devices

Note: Users should run `symcfg sync` before running `symvg show` to make sure the device status is reported correctly.

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

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

Volume Group Types Currently, the following operating systems and their associated logical volume managers (volume group type) are supported:

Table 5-4 Volume Group Types

Operating System	Volume Group Type	VgType
HP-UX	HP-UX LVM HP-UX VxVM EMC PowerPath Volume Manager	HP_LVM HP_VXVM EMC_PVM
IBM AIX	AIX VxVM AIX LVM EMC PowerPath Volume Manager	AIX_VXVM AIX_LVM EMC_PVM
SunOS (Solaris)	SunOS VXVM Solstice Disk Suite EMC PowerPath Volume Manager	SUN_VXVM SUN_SOLSTICE EMC_PVM
Tru64 UNIX	OSF1 LSM	OSF1_LSM
Windows 2000, 2003	LDM	NT_LDM
Linux	VxVM	LINUX_VXVM
AS/400	LVM	AS400_LVM

Volume Group State

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

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

Volume Group Control Operations

EMC Solutions Enabler contains the following volume group control operations:

`create` — Creates a new volume group.

`destroy` — Deletes a volume group.

`recover` — Recovers a failed volume group.

`adddev` — Adds devices to a volume group.

`rmdev` — Removes devices from a volume group.

In addition, you can deport a logical volume group's metadata from a system to storage, and import this same volume group information later to another host, such as a backup server.

Refer to the *EMC Solutions Enabler Symmetrix CLI Command Reference* for descriptions of the command syntax and options, and the *EMC Solutions Enabler Support Matrix* for supported platforms.

Creating a Volume Group

You can create a new volume group by using the following form:

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

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

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

Adding and Removing Devices

To extend a volume group by adding specific devices, use the following form:

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

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

To reduce a volume group by removing specific devices, use the following form:

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

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

Destroying a Volume Group

To delete a volume group, use the following form:

```
symvg destroy VgName
```

Recovering a Volume Group

To recover a failed volume group, use the following form:

```
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 you deport a volume group, that group is not available on the system from which it was deported.

As an example, if you have a volume group that spans several disks, and you want a copy of that volume group on another system, you can perform the following actions:

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

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

Control Options

Table 5-5 describes the options that can be used with the `import` and `deport` operations for volume groups.

Table 5-5 Volume Group Control Options

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

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

Translating Volume Groups to Symmetrix Groups

You can translate the devices of an existing logical volume group and define them to be a Symmetrix device group or composite group.

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

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

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

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

Volume Groups to Device Groups

To create an RDF1 device group named `Prod1dg` with only the standard devices from the volume group named `Prod1vg`, enter:

```
symvg vg2dg Prod1vg Prod1dg -nobcv -dgtype R1
```

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

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

Volume Groups to Composite Groups

To create a composite group named `newcg` with the R1 and R1-BCV devices from the volume group named `thisvg`, enter:

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

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

```
symvg vg2cg Accvg myrdfcg -nobcv -cgtype RDF1
```

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, which are sets of allocated disk blocks called extents.

Listing Logical Volumes

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

```
symmlv -g VgName list
```

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

```
# symmlv -g BigVG list
```

The following is sample output from this command:

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

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

Logical Volume Details

You can view more detail (including extent data) about a logical volume of a specified volume group with the following command:

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

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

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

A sample of the output from this command follows:

```
# symmlv -g 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
-----
(M) 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
...(etc. many more)
    }

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)
...
}
```

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

Logical Volume Conditions

Table 5-6 describes the possible conditions of a logical volume.

Table 5-6 Logical Volume Condition Descriptions

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

Logical Volume States

The possible logical volume states are as follows:

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

Logical Volume Attributes

The possible logical volume attributes are as follows:

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

Logical Volume Control Operations

EMC Solutions Enabler contains the following logical volume control operations:

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

`delete` — Deletes a logical volume.

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

`remove` — Removes mirrors of a logical volume.

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

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

In addition, you can display performance statistics for logical volumes. Refer to Chapter 6, *SRM Statistics*, for more information.

Creating a Logical Volume

You can create a logical volume by using the following form:

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

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

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

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

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

Table 5-7 lists other options for the logical volume control commands.

Table 5-7 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>	Volume group type. The default <i>VgType</i> is assumed if no <i>VgType</i> is specified.

Deleting a Logical Volume

You can delete a logical volume of the specified type by using the following form:

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

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

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

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

Adding and Removing Mirrors

To add a mirror image to a logical volume of the specified type, use the following form:

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

You can use the options `-striped | RAID5`, `-ncols`, `-strsize`, and `-pd` with this command (see Table 5-7).

To remove a mirror from a logical volume of the specified type, use the following form:

```
symmlv -g VgName remove LVolName
```

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

Extending and Reducing Logical Volumes

To extend (grow) a logical volume of the specified type, use the following form:

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

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

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

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

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

Viewing Extents

You can view the physical extents associated with each mirror using the following form:

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

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

```
# symmlv -g 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
-----
(M) 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
(...)
}
```

If the Symmetrix device containing the physical extent is determined to be a meta device, and the extent spans multiple meta device members, the extent will be broken into multiple extent definitions.

Each new definition will describe the portion of the extent that resides on a unique Symmetrix meta device member.

If the physical extent is part of a logical volume that does not reside on a Symmetrix or CLARiiON device, the *SymDevNname* and *SymmlD* fields will be blank.

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

- ◆ (C) CLARiiON device
- ◆ (S) Symmetrix device
- ◆ (M) Symmetrix device meta head
- ◆ (m) Symmetrix device meta member

Extent Conditions

Table 5-8 describes the possible conditions of a logical volume extent.

Table 5-8 Logical Volume Extent Conditions

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

Expanded List

With the `-expand` command, you can get an expanded or collapsed display of extent information.

When the default option (`-collapse`) is used, any physical extents that are stored contiguously on disk (which represents contiguous logical extents in the volume) will be collapsed together to form one extent. The size in the extent structure will be adjusted to reflect the total size of the included extents. The offset will be the 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 isn't logically contiguous.

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

No Extents in List

You can display attributes of a logical volume mirror without the details of extent information.

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

```
symlv -g Prodvlg show voll -no_extents
```

The following is sample output from this command:

```
# symlv -g BigVG show BigLV -no_extents

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

Logical Volume Size       : 10000m

Number of Logical Extents : 2500

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

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

   Mirror Physical Extents (0):

   Mirror Physical Devices (3):
   {
```

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

This chapter describes the Storage Resource Management (SRM) statistics and explains how to generate them using SYMCLI commands.

- ◆ Retrieving Statistics6-2
- ◆ Supported Metrics.....6-2

Retrieving Statistics

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

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

Supported Metrics

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

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

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

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

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

These commands provide the option to specify the number of times to display the statistics (`-c`) and the interval time (`-i`) in seconds.

Table 6-1 lists the SRM statistics commands.

Table 6-1 SRM Statistics Commands

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

symhost Examples

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

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

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

The following is an example of the output:

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

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

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

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

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

The following is an example of the output:

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

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

- ◆ CPU — CPU number/ID
- ◆ Pi/s — Page in requests per second
- ◆ Ppi/s — Number of pages paged in per second
- ◆ Po/s — Page out requests per second
- ◆ Ppo/s — Number of pages paged out per second
- ◆ Si/s — Swap in requests per second
- ◆ Psi/s — Number of pages swapped in per second
- ◆ So/s — Swap out requests per second
- ◆ psos — Number of pages swapped out per second

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

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

The following is an example of the output:

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

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

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

symlv Example

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

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

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

The following is an explanation of the output headings.

Volume Group Type : SunOS VxVM

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

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

- ◆ VgName — Volume group name
- ◆ LvName — Logical volume name
- ◆ RW/s — Read and write requests per second
- ◆ R/s — Read requests per second
- ◆ W/s — Write requests per second
- ◆ KbRW/s — KB read and written per second
- ◆ KbR/s — KB read per second
- ◆ KbW/s — KB written per second
- ◆ %Busy — 100 * (logical volume active time / elapsed time)
- ◆ %Wait — 100 * (non-empty wait queue time / elapsed time)

Note: Statistics are not available for AIX_LVM, LINUX_LVM, and DYNIX_SVM.

symrdb Example

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

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

```
H:M:S sessionID memory
```

Where the first column provides the time of day, the second column provides the session ID, and the last column provides the memory usage per second.

Note: All the statistics values are per second.

Database Metrics The following tables list the information that returns from each database metric. There is a separate table for each database type.

Table 6-2 describes the available metrics for the Oracle database.

Table 6-2 Metric Options for Oracle Database Type

Oracle Metric Option	Returns Statistics About:
INSTANCE	• Instance name • Background checkpoint started • Background checkpoint completed • Database connection count • Total memory usage • Database block buffers • Consistent gets • Consistent changes • Database session • Database file • All of the above
SESSION	• Session ID • Total memory usage
FILE	• File name • Data file physical read • Data file physical write • Data file block read • Data file block write • Physical read time • Physical write time
ALL	All of the Oracle metric options.

Table 6-3 describes the available metrics for the SQL Server database.

Table 6-3 Metric Options for SQL Server Databases

SQL Server Metric Option	Returns Statistics About:
INSTANCE	• Number of physical database page reads issued • Number of physical database page writes issued • Number of pages flushed by checkpoint or other operations that Require all dirty pages to be flushed • Number of latch requests that could not be granted immediately and had to wait before being granted • Average latch wait time for latch requests that had to wait, in milliseconds • Total amount of dynamic memory the server is using for the dynamic SQL cache in KB • Read/write throughput for a backup device, in KB
DATABASE	• Name of the database • Reads performed through the log manager cache • Read/write throughput for backup/restore of a database • Total number of log bytes flushed • Number of commits waiting on log flush • Total wait time for log flush, in milliseconds
OBJECT	• Object name • Number of lock requests that could not be satisfied immediately and required the caller to wait before being granted the lock • Total wait time for locks in the last second in milliseconds • The average amount of wait time for each lock request that resulted in a wait in milliseconds • Number of lock requests that resulted in a deadlock
FILE	• Filename • Number of reads issued on the file • Number of writes issued on the file • KB of read issued on the file • KB of write issued on the file • Total amount of time that users waited for the I/Os to complete on the file in milliseconds
ALL	All of the SQL Server metric options.

Table 6-4 describes the available metrics for the Sybase database.

Table 6-4 Metric Options for Sybase Databases

Sybase Metric Option	Returns Statistics About:
SERVER	• Name of the Adaptive Server that is being monitored • Memory allocated for the page cache (in bytes) • Memory allocated for Adaptive Server • CPU busy time • Number of locks
OBJECT	• Name of the database • Name of the database object • Number of data page reads, whether satisfied from cache or from a database device • Number of data page reads that could not be satisfied from the data cache • Number of data pages written to a database device • Number of combined logical page reads and page writes • Number of locks that were granted after waiting for another lock to be released • Number of locks that were granted immediately
DEVICE	• Device name • Number of reads made from a database device • Number of writes made to a database device • Number of times access to a device was granted • Number of times that access to a device had to wait
ALL	All of the Sybase metric options.

Table 6-5 describes the available metrics for the IBM DB2/UDB database.

Table 6-5 Metric Options for IBMUDB Databases

IBMUDB Metric Option	Returns Statistics About:
DATABASE	• Database name • Buffer pool logical data page reads • Buffer pool physical data page reads • Buffer pool logical index page reads • Buffer pool physical index page reads • Buffer pool data page writes • Buffer pool async data page reads • Buffer pool number async read requests • Buffer pool asynch data page writes • Buffer pool async index page reads • Buffer pool async index page writes • Buffer pool async read time • Buffer pool async write time • Log page read • Log page write

Table 6-5 Metric Options for IBMUDB Databases (continued)

IBMUDB Metric Option	Returns Statistics About:
TABLE	• Database name • Table name • Table type • Number of changes to the table • Number of reads from the table • Number of accesses to overflow rec
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 async data page reads • Number of async read requests • Buffer pool async data page writes • Buffer pool async index page reads • Buffer pool async index page writes • Buffer pool async read time • Buffer pool async write time
ALL	All of the IBMUDB metric options.

C

- concatenated mirror configurations 5-6
- control operations
 - for logical volumes 5-21
 - for volume groups 5-15
- Conventions 1-xii

D

- daemons
 - supported databases 3-25
 - using database 3-23
- data blocks 2-3
- data files, examining 2-9
- data objects 1-3, 2-2, 2-9
- data objects, mapping commands 1-3
- database
 - connections 3-17
 - Exchange 3-14
 - files 3-28
 - I/O controls 3-45
 - IBM/UDB 3-11
 - Informix 3-10
 - instance 3-2
 - mapping 3-2
 - Oracle 3-2
 - servers 3-2
 - setting environment variables 3-16
 - SharePoint Portal Server 3-15
 - shutdown options 3-20
 - startup options 3-17
 - Sybase 3-8
- deport, volume group 5-16

- device groups

 - SYMCLI commands for 1-3, 1-4, 1-5, 1-6

- directories

 - examining 2-9

 - examining attributes 4-7

 - listing files 4-8

 - listing subdirectories 4-6

E

- environment variables 3-16

- Exchange

 - database architecture 3-14

 - database terminology 3-15

- extents 2-3, 4-2, 5-3, 5-24

F

- file systems 4-2

 - examining 2-9, 4-4

 - finding 4-3

 - mapping commands 1-5

- files 4-8

 - data objects 2-9

 - examining 2-9

- freezing databases 3-45

H

- host

 - CPU, sample statistics for 6-3

 - disk, sample statistics for 6-5

 - memory, sample statistics for 6-4

- hot backup 3-46

I

- IBM/UDB
 - database architecture 3-12
 - database startup options 3-19
 - database terminology 3-11
- import
 - volume group 5-16
- Informix
 - database architecture 3-10
 - database terminology 3-11
- I-nodes 4-2
- instance 3-2

L

- logical extents 5-3
- logical volume manager (LVM) 5-18
- logical volumes
 - adding mirrors to 5-22
 - attributes 5-20
 - attributes of 2-7
 - conditions 5-20, 5-25
 - control operations for 5-21
 - creating 5-21
 - deleting 5-22
 - examining data 2-7
 - extending 5-23
 - listing 5-18
 - mapping commands for 1-6, 5-2
 - reducing 5-23
 - removing mirrors from 5-22
 - states 5-20
 - statistics for 6-2
 - statistics, example of 6-6
 - viewing details 5-18

M

- metrics, for statistics 6-2
- mirror configurations 5-5
 - concatenated 5-6
 - determining type 5-10
 - RAID 5 5-7
 - striped 5-7

O

- Oracle

- database shutdown options 3-21
- database startup options 3-18
- database terminology 3-2

P

- physical extents 5-3

R

- RAID 5 mirror configurations 5-7

S

- schemas 3-2
 - examining 3-37
 - listing files 3-39
 - listing segments 3-40
 - symrdb command 3-37
- segments 3-40
 - examining 3-43
 - symrdb command 3-43
- SharePoint Portal Server
 - database architecture 3-15
- simple mirror configurations 5-5
- SQL Server
 - database shutdown options 3-22
 - database startup options 3-20
- statistics
 - for logical volumes 6-2
 - host CPU, example of 6-3
 - host disk 6-5
 - host memory, example of 6-4
 - metrics supported 6-2
 - SYMCLI commands for 1-8
- striped mirror configurations 5-7
- Sybase
 - database shutdown options 3-20
 - database startup options 3-17
 - database terminology 3-8
- SYMCLI commands
 - symhost 1-8
 - symhostfs 1-5
 - symioctl 3-45
 - symlv 1-6, 5-24
 - symrdb 3-27, 3-28
 - symrslv 1-3, 2-7
 - symvlg 1-6, 5-11

T

tables

- examining 3-41

- symrdb command 3-41

tablespaces 3-32

thawing databases 3-45

U

UNIX environment variables, setting 3-16

V

volume groups 5-2, 5-11

- adding devices 5-15

- creating 5-15

- deporting 5-16

- destroying 5-15

- importing 5-16

- listing 5-11

- recovering 5-16

- supported 5-13

- viewing 5-11

