

SANtricity[®] Storage Manager Command Line Interface and Script Commands

Quick Reference for Version 9.10

ES1437-0-E1, First Edition



Proprietary Rights Notice

This document contains proprietary information of Engenio Information Technologies, Inc. The information contained herein is not to be used by or disclosed to third parties without the express written permission of an officer of Engenio. Any product(s) described herein is/are a licensed product of Engenio.

Document Description

Document ES1437-0-1, First Edition. July 2004

This document provides a quick reference for the command line interface and script command software for SANtricity Storage Manager version 9.10, and will remain the official reference source for all revisions and releases of this product until rescinded by an update.

Disclaimer

It is the policy of Engenio Information Technologies to improve products as new technology, components, software, and firmware become available. Engenio reserves the right to make changes to any products herein at any time without notice. All features, functions, and operations described herein may not be marketed by Engenio in all parts of the world. In some instances, photographs and figures are of equipment prototypes. Therefore, before using this document, consult your Engenio representative for information that is applicable and current. **ENGENIO DOES NOT ASSUME ANY RESPONSIBILITY OR LIABILITY FOR THE USE OF ANY PRODUCT(S) DESCRIBED HEREIN EXCEPT AS EXPRESSLY AGREED TO IN WRITING BY ENGENIO.**

License Restriction

The purchase or use of an Engenio product does not convey a license under any patent, copyright, trademark, or other intellectual property right of Engenio or third parties.

Copyright Notice

© 2004. Engenio Information Technologies, Inc. All rights reserved.

Trademark Acknowledgments

Engenio, the Engenio design, and HotScale are trademarks, and SANtricity Storage Manager and SANshare are registered trademarks of Engenio Information Technologies, Inc. All other brand and product names may be trademarks of their respective companies.

Conventions

The following conventions have been used throughout this book.

Definitions of Safety Notices

DANGER indicates an imminently hazardous situation that will result in death or severe personal injury.

WARNING indicates a potentially hazardous situation that could result in death or severe personal injury.

CAUTION indicates a potentially hazardous situation that could result in moderate or minor personal injury.

Definitions of Informational Notices

CAUTION indicates a potentially hazardous situation that could result in data loss (or other interruption) or equipment damage.

IMPORTANT indicates information or criteria that is necessary to perform a procedure correctly.

NOTE indicates a concept that will be clarified or a maintenance tip that will be presented.

Revision Record

Edition or Revision	Date	Affected Pages or Remarks
First Edition	July 2004	New Book.

Part Number: ES1437-0-E1

Contents

COMMAND LINE INTERFACE AND SCRIPT COMMANDS QUICK REFERENCE

About the Command Line Interface	2
How to Use the Command Line Interface	2
Usage Notes	3
CLI Commands	3
Command Line Parameters	5
Exit Status	8
CLI Command Formatting	9
Usage Examples	10
About the Script Commands	12
Script Command Structure	12
Recurring Syntax Elements	13
Usage Guidelines	16
Adding Comments to a Script File	17
Script Command Formatting	18
Script Commands	20
Commands Listed by Function	40
Controller Commands	40
Host Topology Commands	40
Physical Disk (Drive) Commands	41
Remote Volume Mirror Commands	41
Session Command	41
Snapshot Commands	42
Storage Array Commands	42
Tray Commands	43
Uncategorized Command	43
Volume Commands	43
Volume Copy Commands	44
Volume Group Commands	44

List of Tables

COMMAND LINE INTERFACE AND SCRIPT COMMAND QUICK REFERENCE

Table 1. Command Name Conventions	3
Table 2. Command Line Parameters	5
Table 3. Object Types and Identifiers	13
Table 4. Recurring Syntax Elements	13
Table 5. Script Commands	20

About This Book

This book is a quick reference that lists the command line interface commands and the script engine commands for SANtricity Storage Manager 9.10. It provides a brief overview of the command line interface and script commands, lists the commands for the command line interface, lists the commands for the script engine, provides command syntax and briefly describes the purpose of each script command. This book assumes that the user has a knowledge of basic storage area network (SAN) hardware and installation skills. Read the SANtricity Storage Manager Product Release Notes for any updated information regarding hardware, software, or firmware products that may not be covered in this document.

Intended Readers

This book is intended for system operators, system administrators, and service personnel who are responsible for operating, maintaining, troubleshooting, and servicing a storage array. Users must be familiar with computer system operation, maintenance, and repair. In addition, they should understand disk array, RAID, network and Fibre Channel technologies. The reader must have a basic knowledge of SAN hardware functionality (controllers, drives, hosts) and SAN cabling techniques; and must understand disk array, Redundant Array of Independent Disks (RAID), network, and Fibre Channel technologies

Related Publications

Command Line Interface and Script Commands Programming Guide

Web Address

For web sites related to the products in this publication, please see the Product Release Notes.

Command Line Interface and Script Commands Quick Reference

The command line interface and script commands provide a tool set that enables users to configure and maintain a storage array directly from an operating system command line. This quick reference provides

- A brief description of the command line interface.
- A list of the commands and parameters for the command line interface (CLI commands).
- An overview of the script commands.
- An explanation of the command formatting.
- A table that alphabetically lists the script commands and provides a description of the purpose of each command.
- A functional list of the commands, in which the commands are organized by operational utility in the storage array.

This quick reference is intended to help you rapidly find the correct syntax and parameters for commands you are using. It is not intended to be a complete programming guide. This quick reference is for users who have read and understood the information in the *Command Line Interface and Script Commands Programming Guide*. If you have questions related to the commands or parameters listed in this quick reference, refer to the programming guide for additional information.

About the Command Line Interface

The command line interface is a software tool that enables storage array installers, developers, and engineers to configure and monitor storage arrays. Using the command line interface, you can issue commands from an operating system prompt, such as the DOS C:\ prompt, a Linux path, or a Solaris path. Each command performs a specific action for managing a storage array or returning information about the status of a storage array. You can enter individual commands or you can run script files when you need to perform operations more than once (such as installing the same configuration on several storage arrays). The command line interface enables you to load a script file from a disk and run the script file. The command line interface provides a way to run storage management commands on more than one network storage array. You can employ the command line interface in both installation sites and development environments.

The command line interface gives you direct access to a script engine that is a utility in the SANtricity storage management software. The script engine runs commands that enable you to configure and manage storage arrays. The script engine reads the commands, or runs a script file, from the command line and performs the operations instructed by the commands.

How to Use the Command Line Interface

The commands you run on the command line interface (CLI commands) provide access to the script engine, specify the storage array to receive the script commands, and set operation environment parameters.

A CLI command consists of the following elements:

- The term **SMcli**
- Storage array identifier
- Parameters
- Script commands

The general form of a CLI command is:

```
SMcli host parameters script-commands;
```

SMcli invokes the command line interface, **storage array** is the name or IP address of the storage array, **parameters** are CLI parameters that define the environment and purpose for the command, and **script command** is one or more script commands or the name of a script file containing script commands. (The script commands are the storage array configuration commands.)

Usage Notes

If you enter SMcli and a storage array name, but do not specify CLI parameters, script commands, or a script file, the command line interface runs in interactive mode. Interactive mode enables you to run individual commands without prefixing the commands with SMcli. In interactive mode you can enter a single command, see the results, and enter the next command without typing the complete SMcli string. Interactive mode is useful for determining configuration errors and quickly testing configuration changes.

If you enter SMcli without any parameters or with an incorrect parameter, the script engine returns usage information.

CLI Commands

This section lists the CLI commands you can use to identify storage arrays, specify passwords, add storage arrays to configuration files, specify communications parameters, enter individual script configuration commands, or specify a file containing script configuration commands. Table 1 lists the conventions used in the general form of the CLI command.

Table 1 Command Name Conventions

Convention	Definition
a b	alternative (“a” or “b”)
<i>italicized-words</i>	terminals
[...] (square brackets)	zero or one occurrence
{...} (curly braces)	zero or more occurrences
(a b c)	choose only one of the alternatives
bold	terminals

The general forms of the CLI commands, showing the parameters and terminals used in each command, are listed below. Table 2 on page 5 lists definitions for the parameters shown in the general form of the CLI commands.

```
SMcli host-name-or-IP-address [host-name-or-IP-address] [-c "command; {command2;}"] [-n storage-array-name | -w WWN] [-o outputfile] [-p password] [-e] [-S]
```

```
SMcli host-name-or-IP-address [host-name-or-IP-address] [-f scriptfile] [-n storage-array-name | -w WWN] [-o outputfile] [-p password] [-e] [-S]
```

```
SMcli -n storage-array-name | -w WWN [-c "command;
```

```

{command2;}" [-o outputfile] [-p password] [-e] [-S]

SMcli -n storage-array-name | -w WWN [-f scriptfile] [-o
outputfile] [-p password] [-e] [-S]

SMcli -n storage-array-name | -w WWN [-o outputfile] [-p
password] [-e] [-S]

SMcli (-a | -x) email:email-address [-n storage-array-name |
-w WWN | -h host-name | -r (inband_sa | outofband_sa)] [-S]

SMcli (-a | -x) email:email-address host-name-or-IP-address
[host-name-or-IP-address] [-n storage-array-name | -w WWN |
-h host-name | -r (inband_sa | outofband_sa)] [-S]

SMcli (-a | -x) trap:community,host-name-or-IP-address [-n
storage-array-name | -w WWN | -h host-name | -r (inband_sa |
outofband_sa)] [-S]

SMcli (-a | -x) trap:community,host-name-or-IP-address host-
name-or-IP-address [host-name-or-IP-address] [-n storage-
array-name | -w WWN | -h host-name | -r (inband_sa |
outofband_sa)] [-S]

SMcli -d [-w] [-i] [-s] [-v] [-S]

SMcli -m host-name-IP-address -F email-address [-S]

SMcli -A [host-name-or-IP-address [host-name-or-IP-address]]
[-S]

SMcli -X (-n storage-array-name | -w WWN | -h host-name)

SMcli -?

```

Command Line Parameters

Table 2 Command Line Parameters

Parameter	Definition
<i>hostname or IP address</i>	<p>You can specify either the host name or the IP address (xxx.xx.xx.xx) of an in-band managed storage array or an out-of-band managed storage array.</p> <ol style="list-style-type: none"> 1. If you are managing a storage array using a host through in-band storage management, you must use the <code>-n</code> or <code>-w</code> parameter if more than one storage array is connected to the host. 2. If you are managing a storage array using out-of-band storage management through the ethernet connection on each controller, you must specify the <i>host-name-or-IP-address</i> of the controllers. 3. If you have previously configured a storage array in the EMW, you can specify the storage array by its user-supplied name using the <code>-n</code> parameter. 4. If you have previously configured a storage array in the EMW, you can specify the storage array by its world wide name using the <code>-w</code> parameter.
<code>-A</code>	<p>Use this parameter to add a storage array to the configuration files. If you do not follow the <code>-A</code> parameter with a <i>host-name-or-IP-address</i>, auto-discovery scans the local subnet for storage arrays.</p>
<code>-a</code>	<p>Use this parameter to add an SNMP trap destination or an email address alert destination.</p> <ul style="list-style-type: none"> • When adding an SNMP trap destination, the SNMP community is automatically defined as the community name for the trap and the <i>host</i> is the IP address or DNS host name of the system to which the trap should be sent. • When adding an email address for an alert destination the <i>email-address</i> is the email address where you want the alert message to be sent.
<code>-c</code>	<p>Use this parameter to indicate you are entering one or more script commands to run on the specified storage array. Each command must be terminated by a semicolon (;).</p> <p>You cannot place more than one <code>-c</code> parameter on the same command line. You can include more than one script command after the <code>-c</code> parameter.</p>

Parameter	Definition
-d	<p>Use this parameter to display the contents of the script configuration file. The format of the file contents is:</p> <pre data-bbox="597 415 1318 449">storage-Array-name host-name1 host-name-2</pre>
-e	<p>Use this parameter to run the commands without performing a syntax check first.</p>
-f	<p>Use this parameter to specify a file name containing script commands you want to run on the specified storage array. (This parameter is similar to the -c parameter in that both are intended for running script commands. The -c parameter runs individual script commands. The -f parameter runs a file of script commands.)</p> <p>By default, any errors encountered when running the script commands in a file are ignored, and the file continues to run. To override this behavior use the <code>set session errorAction=stop</code> command in the script file.</p>
-F	<p>Use this parameter to specify the email address from which all alerts will be sent.</p>
-h	<p>Use this parameter to specify the host name that is running the SNMP agent to which the storage array is connected. Use this parameter with the -a and -x parameters.</p>
-i	<p>Use this parameter to display the IP address of the known storage arrays. Use this parameter with the -d parameter. The format of the file content is:</p> <pre data-bbox="597 1312 1211 1375">storage-Array-name IP-address-1 IP-address-2</pre>
-m	<p>Use this parameter to specify the host name or IP address of the email server from which email alert notifications will be sent.</p>

Parameter	Definition
-n	<p>Use this parameter to specify the name of the storage array on which you want to run the script commands. This name is optional when you use a “host-name” or “IP address;” however, if you are using the in-band method for managing the storage array, you must use the -n parameter if more than one storage array is connected to the host at the specified address.</p> <p>The storage array name is required when the <i>host-name</i> or <i>IP address</i> is not used; however, the name of the storage array configured for use in the EMW (that is, listed in the configuration file) must not be a duplicate name of any other configured storage array.</p>
-o	<p>Use this parameter with the -c or -f parameters to specify a file name for all output text that is a result of running the script commands. If you do not specify an output file, the output text will go to stdout. All output from commands that are not script commands is sent to stdout, regardless of whether this parameter is or is not set.</p>
-p	<p>Use this parameter to specify the password for the storage array on which you want to run commands. A password is not necessary under the following conditions:</p> <ol style="list-style-type: none"> 1. A password has not been set on the storage array. 2. The password is specified in a script file you are running. 3. You specify the password using the -c parameter and the <code>set session password=string-literal</code> command.
-r	<p>Use this parameter with the -a or -x parameters to specify the name of an organizer node. The name of an organizer node can be either <i>outofband_sa</i> (out-of-band storage array organizer node) or <i>inband_sa</i> (in-band storage arrays organizer node). The -r parameter enables you to set or change the alert notifications for all storage arrays under each organizer node.</p>

Parameter	Definition
-S	Use this parameter to suppress informational messages describing command progress that appear when running script commands. (Suppressing informational messages is also called “silent mode.”) This parameter suppresses the following messages: <ul style="list-style-type: none"> • Performance syntax check. • Syntax check complete. • Executing script. • Script execution complete. • SMcli completed successfully.
-s	Use this parameter with the -d parameter to display the alert settings in the configuration file.
-v	Use this parameter with the -d parameter to display the current global status of the known devices in a configuration file.
-w	Use this parameter to specify the world wide name (WWN) of the storage array. This parameter is an alternate to the -n parameter. Use the -w parameter with the -d parameter of display the WWNs of the known storage arrays. The format of the file content is: <pre>storage-Array-name world-wide-name IP- address-1 IP-address-2</pre>
-X	Use this parameter to delete a storage array from a configuration.
-x	Use this parameter to remove an SNMP trap destination or and email address alert destination. The <i>community</i> is the SNMP community name for the trap and the <i>host</i> is the IP address or DNS host name of the system to which you want the trap sent.
-?	Use this parameter to display usage information about the CLI commands.

Exit Status

After you run a CLI command or CLI and script command, status indicating the success of the operation defined by the command is displayed. The status are:

0 - terminated without error.

1 - terminated with error. Information about the error is also displayed.

CLI Command Formatting

A double quote symbol (“”) used as part of a name or label requires special consideration when running CLI and script commands under a Windows operating system. The following bullet items explain how to use double quotes in names while running under a Windows operating system.

- When double quotes are part of a name or argument, you must insert a backslash (\) before each double quote character. For example:

```
-c "set storageArray userLabel=\"Engineering\";"
```

where Engineering is the storage array name. A second example is:

```
-n "\"My\" Array"
```

where “My” Array is the name of the storage array.

- You cannot use the double quote character (“”) as part of a character string (also called *string literal*) within a script command. For example, you cannot enter the following string to set the storage array name to “Finance” Array:

```
-c "set storageArray userLabel=\"\"Finance\"Array\";"
```

In a UNIX operating system the delimiter around names or labels are single quotes. The UNIX versions of the previous examples are:

```
-c `set storageArray userLabel="Engineering";`
```

```
-n `"My" Array`
```

Because the backslash character (\) is used with double quotes in a Windows operating system, the backslash requires special consideration when used as part of a name or label.

- Use the backslash character (\) as typed, unless you want to use it immediately before a double quote character (“”). Use three backslashes before the double quote character to have the backslash printed when you use the `-n`, `-o`, `-f`, or `-p` parameters. For example, to specify a storage array named Array\, enter:

```
-n "Array\\\""
```

- To use a backslash character (\) as part of a string literal within a script command, the same rules as in the previous bullet item apply except that you need to use five backslashes. For example, to change the name of a storage array to Array\, enter the following string:

```
-c "set storageArray userLabel="Array\\\\\\\";"
```

A Windows operating system requires that you insert a caret (^) before each special script character. Special characters are ^, &, |, <, >

- Insert a caret before each special script character when used with the `-n`, `-o`, `-f`, and `-p` parameters. For example, to specify storage array `CLI&CLIENT` enter the following string:

```
-n "CLI^&CLIENT"
```

- Insert three carets (`^^^`) before each special script character when used within a script command string literal. For example, to change the name of a storage array to `FINANCE&PAYROLL`, enter the following string:

```
-c "set storageArray userLabel=\"FINANCE^^^&PAYROLL\";"
```

Usage Examples

The following examples show how to enter CLI commands on a command line. The examples show the syntax, form, and, in some examples, script commands. Examples are shown for both Windows and UNIX systems. Note that the usage for the `-c` parameter varies depending on your operating system. On Windows operating systems, the script command following the `-c` must be enclosed in double quotation marks (“”). On UNIX systems, the script command following the `-c` must be enclosed in single quotation marks (‘’).

1. This example shows how to delete an existing volume and create a new volume on a storage array. The existing volume name is **Stocks & Bonds**. The new volume name will be **Finance**. The controller host names are **finance1** and **finance2**. The storage array is protected, requiring the password **TestArray**.

Windows operating system:

```
SMcli finance1 finance2 -c "set session
password=\"TestArray\"; delete volume[\"Stock^^^&
Bonds\"]; create volume driveCount[3] RAIDLEVEL=3
capacity=10GB userLabel=\"Finance\"; show storageArray
healthStatus;"
```

UNIX operating system:

```
SMcli finance1 finance2 -c `set session
password="TestArray"; delete volume["Stock & Bonds"];
create volume driveCount[3] RAIDLEVEL=3 capacity=10GB
userLabel="Finance"; show storageArray healthStatus;`
```

2. This example shows how to run commands in a script file named **scriptfile.scr** on a storage array named **Example**. In this example the storage array is protected by the password **My Array**. Output, as a result of commands in the script file, will go to file *output.txt*.

Windows operating system:

```
SMcli -n Example -f scriptfile.scr -p "My Array" -o
```

output.txt

UNIX operating system:

```
SMcli -n Example -f scriptfile.scr -p 'My Array' -o  
output.txt
```

About the Script Commands

You can use the script commands to configure and manage a storage array. The script commands are distinct from the CLI commands; however, you enter the script commands using the command line interface. You can enter individual script commands or you can run a file of script commands. When you enter an individual script command, you include it as part of a CLI command. When you run a file of script commands, you include the file name as part of a CLI command. The script commands are processed by a script engine that:

- Verifies command syntax
- Interprets the commands
- Converts the commands to the appropriate protocol-compliant commands
- Passes the commands to the storage array

At the storage array, the script commands are run by the storage array controllers.

Script Command Structure

All script commands have the following structure:

```
command operand-data { statement-data }
```

where *command* identifies the action to be performed, *operand-data* represents the objects associated with a storage array you want to configure or manage, and *statement-data* provides the information needed to perform the command.

The syntax for *operand-data* is:

```
(object-type | allobject-types | [qualifier] (object-type  
[identifier] {object-type identifier}) | object-types  
[identifier-list])
```

An object can be identified four ways: object type, *all* parameter, square brackets, or a list of identifiers. Use an object type when the command is not referencing a specific object. The *all* parameter means all objects of the specified type in the storage array (for example, allVolumes). To perform a command on a specific object, use square brackets to identify the object (for example, volume[engineering]). Specify a subset of objects with a list of identifiers in square brackets (for example, volumes[sales engineering marketing]). A qualifier is necessary if you want to include additional information to describe the objects. Table 3 lists the object type and identifiers associated with the object types.

Table 3 Object Types and Identifiers

Object Type	Identifier
controller	a or b
drive	tray id and slot id
driveChannel	drive channel identifier
host	user label
hostChannel	host channel identifier
hostGroup	user label
hostPort	user label
remote mirror	primary volume user label
snapshot	volume user label
storageArray	N/A
tray	tray id
volume	volume user label or volume WWN (set command only)
volumeCopy	target volume and optionally the source volume user labels
volumeGroup	volume group number

Statement data is in the form of **attribute=value** (raidLevel=5), an **attribute name** (batteryInstallDate), or an **operation name** (redundancyCheck).

Recurring Syntax Elements

Recurring syntax elements are a general category of variables and parameters you can use in one or more script commands. Table 4 lists the recurring syntax and the values that you can use with the syntax. The conventions listed in Table 1 on page 3 define the meaning of the syntax values.

Table 4 Recurring Syntax Elements

Recurring Syntax	Syntax Value
<i>raid-level</i>	(0 1 3 5)
<i>repository-raid-level</i>	(1 3 5)
<i>capacity-spec</i>	<i>integer-literal</i> [KB MB GB TB Bytes]

Recurring Syntax	Syntax Value
<i>segment-size-spec</i>	<i>integer-literal</i>
<i>boolean</i>	(TRUE FALSE)
<i>user-label</i>	<i>string-literal</i>
<i>user-label-list</i>	<i>user-label { user-label }</i>
<i>create-raid-vol-attr-value-list</i>	<i>create-raid-volume-attribute-value-pair { create-raid-volume-attribute-value-pair }</i>
<i>create-raid-volume-attribute-value-pair</i>	capacity = <i>capacity-spec</i> owner =(a b) readAheadMultiplier = <i>integer-literal</i> segmentSize = <i>integer-literal</i>
<i>trayId</i>	(0 - 99)
<i>slot-id</i>	(1 - 32)
<i>port-id</i>	(0 - 127)
<i>drive-spec</i>	<i>trayId</i> , <i>slotId</i>
<i>drive-spec-list</i>	<i>drive-spec { drive-spec }</i>
<i>tray-id-list</i>	<i>trayId { trayId }</i>
<i>hex-literal</i>	<i>0hexadecimal-literal</i>
<i>volume-group-number</i>	<i>integer-literal</i>
<i>filename</i>	<i>string-literal</i>
<i>error-action</i>	(stop continue)
<i>drive-channel-identifier</i>	(1 2 3 4)
<i>drive-channel-identifier-list</i>	<i>drive-channel-identifier { drive-channel-identifier }</i>
<i>host-channel-identifier</i>	(a1 a2 b1 b2)
<i>drive-type</i>	(fibre SATA PATA)
<i>feature-identifier</i>	(storagePartition2 storagePartition4 storagePartition8 storagePartition16 storagePartition64 storagePartitionMax snapshot remoteMirror volumeCopy)
<i>repository-spec</i>	<i>instance-based-repository-spec</i> <i>count-based-repository-spec</i>

Recurring Syntax	Syntax Value
<i>instance-based-repository-spec</i>	repositoryRAIDLevel = <i>repository-raid-level</i> repositoryDrives =(<i>drive-spec-list</i>) [trayLossProtect = <i>boolean</i> ¹] repositoryVolumeGroup = <i>volume-group-number</i> repositoryFreeCapacityArea = <i>integer-literal</i> ² Specify either repositoryRAIDLevel with repositoryDrives , or repositoryVolumeGroup . Do not specify RAID level or drives with volume group.
<i>count-based-repository-spec</i>	repositoryRAIDLevel = <i>repository-raid-level</i> repositoryDriveCount = <i>integer-literal</i> [driveType = <i>drive-type</i> ³] [trayLossProtect = <i>boolean</i> ⁴]
<i>WWN</i>	<i>string-literal</i>
<i>nvsram-offset</i>	<i>hex-literal</i>
<i>host-type</i>	<i>string-literal</i> <i>integer literal</i>
<i>nvsram-byte-setting</i>	<i>nvsram-value-> 0xhexadecimal</i> <i>integer-literal</i>
<i>nvsram-bit-setting</i>	<i>nvsram-mask, nvsram-value-> 0xhexadecimal, 0xhexadecimal</i> <i>integer-literal</i>
<i>ip-address</i>	[0 - 255].[0 - 255].[0 -255].[0 - 255]
<i>autoconfigure-vols-attr-value-list</i>	<i>autoconfigure-vols-attr-value-pair</i> { <i>autoconfigure-vols-attr-value-pair</i> }
<i>autoconfigure-vols-attr-value-pair</i>	driveType = <i>drive-type</i> ⁵ raidLevel = <i>raid-level</i> volumeGroupWidth = <i>integer-literal</i> volumeGroupCount = <i>integer-literal</i> volumesPerGroupCount = <i>integer-literal</i> ⁶ hotSpareCount = <i>integer-literal</i> segmentSize = <i>segment-size-spec</i> readAheadMultiplier = <i>integer-literal</i>
<i>create-volume-copy-attr-value-list</i>	<i>create-volume-copy-attr-value-pair</i> { <i>create-volume-copy-attr-value-pair</i> }
<i>create-volume-copy-attr-value-pair</i>	copyPriority =(highest high medium low lowest) targetReadOnlyEnabled = <i>boolean</i>

Recurring Syntax	Syntax Value
<i>recover-raid-volume-attr-value-list</i>	<i>recover-raid-volume-attr-value-pair {recover-raid-volume-attr-value-pair}</i>
<i>recover-raid-volume-attr-value-pair</i>	owner=(a b) readAheadMultiplier=<i>integer-literal</i>
<i>cache-flush-modifier-setting</i>	immediate, 0, .25, .5, .75, 1, 1.5, 2, 5, 10, 20, 60, 120, 300, 1200, 3600, infinite

¹For tray loss protection to work, each physical disk in a volume group must be on a separate module. If you set `trayLossProtect=TRUE` and have selected more than one physical disk from any one module, the storage array will return an error. If you set `trayLossProtect=FALSE`, the storage array will perform operations, but the volume group you create may not have tray loss protection.

²To determine if a free capacity area exists, issue the `show volumeGroup` command.

³The default physical disk (drive type) is “fibre” (Fibre Channel).

⁴If you set `trayLossProtect` to true, the storage array will return an error if the controller firmware cannot find drives that will enable the new volume group to have tray loss protection. If you set `trayLossProtect` to false, the storage array will perform the operation even if it means the volume group may not have tray loss protection.

⁵The `driveType` parameter is not required if only one type of physical disk is in the storage array. If you use the `driveType` parameter, you must also use the `hotSpareCount` and `volumeGroupWidth` parameters. If you do not use the `driveType` parameter, the configuration defaults to Fibre Channel physical disks.

⁶`volumesPerGroupCount` is a parameter that defines the number of equal capacity volumes per volume group.

Usage Guidelines

- You must end all commands with a semicolon (;).
- You can enter more than one command on a line, but you must separate each command with a semicolon (;).
- You must separate each base command and its associated primary and secondary parameters with a space.
- The script engine is not case sensitive. You can enter commands using upper case, lower case, or mixed case.
- Add comments to your scripts to make it easier for you and future users to understand the purpose of the script commands. (For information on how to add

comments, refer to “Adding Comments to a Script File” on page 17.)

Adding Comments to a Script File

You can add comments to a script file in three ways.

1. The script engine interprets any text typed after two forward slashes (`//`) until an end of line character is reached. If the script engine does not find an end of line character in the script after processing a comment, an error message is displayed and the script operation is terminated. This error commonly occurs when a comment is placed at the end of a script and you have forgotten to press the *Enter* key.

```
// Deletes the existing configuration.  
set storageArray resetConfiguration=true;
```

2. The script engine interprets any text type between `/*` and `*/` as a comment. If the script engine does not find both a beginning and ending comment notation, an error message is displayed and the script operation is terminated.

```
/* Deletes the existing configuration */  
set storageArray resetConfiguration=true;
```

3. Use the **show** statement to embed comments in a script file that you want to display while the script file is running. The text you want to display must be enclosed by quotation marks (“”).

```
show "Deletes the existing configuration";  
set storageArray resetConfiguration=true;
```

Script Command Formatting

This section describes the general rules for formatting a script command and how the command syntax is presented in the following command descriptions. Syntax unique to a specific command is explained in the “Notes” section at the end of each command description.

- The script commands are not case sensitive. You can type the commands in lower case, upper case, or mixed case. (In the following command descriptions, mixed case is used as an aid to reading the command names and understanding the purpose of the command.)
- You must enter spaces in the commands as they are shown in the command descriptions.
- Square brackets used in two ways:
 - As part of the command syntax
 - To indicate the parameters are optional

The description of each parameter will tell you if you need to enclose a parameter value in square brackets.

- Parentheses shown in the command syntax enclose specific choices for a parameter; that is, if you want to use the parameter you must enter one of the values enclosed in parentheses. Generally, you will not include parentheses in a command; however, in some instances, when you enter lists you must enclose the list in parentheses. Such a list might be a list of tray ID and slotID values. The description of each parameter will tell you if you need to enclose a parameter values in parentheses.
- Vertical bars in a command indicate “or” and separate the valid entries for the parameter. For example, the syntax for the `raidLevel` parameter in the command description appears as:

```
raidLevel=(0 | 1 | 3 | 5)
```

To use the `raidLevel` parameter to set a RAID level of “5” enter:

```
raidLevel=5
```

- When you specify physical disk locations using “trayID” and “slotID” values, separate the ID values by a comma. If you are entering more than one set of ID values, separate each set of values by a space. Enclose the set of values in parentheses. For example:

```
(1,1 1,2 1,3 1,4 2,1 2,2 2,3 2,4)
```

- Italic terms in the command indicate a value or information that you need to provide. For example, when you encounter the italic term:

numberOfDrives

replace the italic term with a value for the number of physical disks you want to include with the command.

- You can use any combination of alphanumeric characters and the underscore (_) and dash (-) characters for the names of the following components:

- Storage arrays
- Host groups
- Hosts
- Volume groups
- Volumes
- Host port

Names can have a maximum of 30 characters. If the label contains more than one word, white spaces, underscores, or dashes, enclose the name in double quotes. In some usage, you must also surround the name with square brackets. The description of each parameter will indicate whether you need to enclose a parameter in double quotes or square brackets. The character string cannot contain a new line. Make sure you use unique names; if you do not use unique names the controller firmware will return an error.

When you enter a world-wide name (WWN) of a host port, some usage requires you to surround the WWN with double quotes. In other uses you must surround the WWN with angle brackets (< >). The description of the WWN parameter will indicate whether you need to enclose the WWN in double quotes or angle brackets.

- Script commands must end with a semicolon (;). You can enter more than one script command on the command line each time you enter a CLI command.

Script Commands

Table 5 Script Commands

activate
<pre>activate storageArray feature=remoteMirror repositoryRAIDLevel=(1 3 5) repositoryDrives=(trayID1,slotID1 ... trayIDn,slotIDn) [trayLossProtect=(TRUE FALSE)]</pre> <p>This command creates the mirror repository volume and activates the Remote Volume Mirror feature. This command enables users to define which physical disks will be used for the repository volume.</p>
<pre>activate storageArray feature=remoteMirror repositoryVolumeGroup=volumeGroupNumber [repositoryFreeCapacityArea=freeCapacityValue trayLossProtect=(TRUE FALSE)]</pre> <p>This command creates the mirror repository volume and activates the Remote Volume Mirror feature. This command enables users to define the volume group in which the repository volume will be located.</p>
<pre>activate storageArray feature=remoteMirror repositoryRAIDLevel=(1 3 5) repositoryDriveCount=numberOfDrives [driveType=(fibre SATA PATA) trayLossProtect=(TRUE FALSE)]</pre> <p>This command creates the mirror repository volume and activates the Remote Volume Mirror feature. This command enables users to define the number of physical disks to be used for the repository volume.</p>
<pre>activate storageArray firmware</pre> <p>This command activates firmware previously downloaded to the pending configuration area on the controllers in the storage array.</p>

autoConfigure

```
autoConfigure storageArray [driveType=(fibre | SATA | PATA)
raidLevel=(0 | 1 | 3 | 5) volumeGroupWidth=numberOfDrives
volumeGroupCount=numberOfVolumeGroups
volumesPerGroupCount=numberOfVolumesPerGroup
hotSpareCount=numberOfHotspares segmentSize=segmentSizeValue
readAheadMultiplier=multiplierValue]
```

This command automatically configures a storage array. Before entering the this command, enter the `show storageArray autoConfiguration` command to return a list of valid physical disk types, RAID levels, volume information, and hot spare information. If you want to modify the configuration, you can change the parameters to meet your configuration requirements. You can change a single parameter or all parameters. After you enter the `autoConfigure storageArray` command, the controllers set up the storage array using either the default parameters or those you selected.

```
autoConfigure storageArray hotSpares
```

This command automatically defines and configures the hot spares in a storage array. You can run this command at any time. This command provides the best hot spare coverage for a storage array.

check volume

```
check volume [volumeName] parity [parityErrorFile=filename]
[mediaErrorFile=filename] [priority=(highest | high | medium |
low | lowest)] [verbose=(TRUE|FALSE)]
```

This command checks a volume for parity and media errors, and writes the results of the check to a file.

clear

```
clear allDriveChannels stats
```

This command resets the statistics for all physical disk channels.

```
clear storageArray configuration
```

This command clears the entire configuration from the controllers in a storage array. Information defining all volume groups, volumes, and hot spares is deleted. Use this command when you need to create a new configuration on a storage array that already has a configuration defined.

Caution:

Potential storage array configuration damage. As soon as you run this command, the existing storage array configuration is deleted.

```
clear storageArray eventLog
```

This command clears the storage array event log by deleting the data in the event log buffer.

Caution:

Potential storage array configuration damage. As soon as you run this command, the existing storage array configuration is deleted.

```
clear storageArray firmwarePendingArea
```

This command deletes from the pending area buffer a firmware image or NVSRAM values you have previously downloaded.

Caution:

Potential storage array configuration damage. As soon as you run this command, the existing storage array configuration is deleted.

```
clear (allVolumes | volume [volumeName] |  
volumes [volumeName1 ... volumeNameN]) reservations
```

This command clears persistent volume reservations.

```
clear (allVolumes | volume [volumeName] |  
volumes [volumeName1 ... volumeNameN]) unreadableSectors
```

This command clears unreadable sector information from one or more volumes.

create

```
create host userLabel="hostName" [hostGroup=( "hostGroupName" |  
defaultGroup) ]
```

This command creates a new host. If you do not specify a host group in which to create the new host, it is created in the default group.

```
create hostGroup userLabel="hostGroupName"
```

This command creates a new host group.

```
create hostPort identifier="wwn" userLabel="portLabel"  
host="hostName"  
[hostType=(hostTypeIndexLabel | hostTypeIndexNumber) ]
```

This command creates a new host port.

```
create volume driveCount=numberOfDrives
raidLevel=(0 | 1 | 3 | 5) userLabel="volumeName"
[driveType=(fibre | SATA | PATA) capacity=volumeCapacity
owner=(a | b) readAheadMultiplier=multiplierValue
segmentSize=segmentSizeValue trayLossProtect=(TRUE | FALSE)]
```

This command creates a volume group across the storage array physical disks and a new volume in the volume group. The storage array controllers choose the physical disks to be included in the volume.

```
create volume drives=(trayID1,slotID1...trayIDn,slotIDn)
raidLevel=(0 | 1 | 3 | 5) userLabel="volumeName"
[capacity=volumeCapacity owner=(a | b)
readAheadMultiplier=multiplierValue
segmentSize=segmentSizeValue trayLossProtect=(TRUE | FALSE)]
```

This command creates a new volume group and volume and enables you to specify the physical disks for the volume.

Important:

You cannot use mixed physical disk types in the same volume group and volume. This command will fail if you specify different types of physical disks for the RAID volume.

```
create volume volumeGroup=volumeGroupNumber
userLabel="volumeName" [freeCapacityArea=freeCapacityValue
capacity=volumeCapacity owner=(a | b)
readAheadMultiplier=multiplierValue
segmentSize=segmentSizeValue]
```

This command creates a volume in the free space of a volume group.

```
create remoteMirror primary="primaryVolumeName"
secondary="secondaryVolumeName"
(remotestorageArrayName="storageArrayName" |
remoteStorageArrayWwn="wwn") [remotePassword="password"
syncPriority=(highest | high | medium | low | lowest)
writeOrder=(preserved | notPreserved)
writeMode=(synchronous | asynchronous)]
```

This command creates both the primary and secondary volumes for a Remote Volume Mirror. This command also sets the write mode (synchronous or asynchronous) and synchronization priority.

```
create snapshotVolume baseVolume="baseVolumeName"  
[repositoryRAIDLevel=(1 | 3 | 5)  
repositoryDrives=(trayID1,slotID1 ... trayIDn,slotIDn)  
userLabel="snapshotVolumeName"  
warningThresholdPercent=percentValue  
repositoryPercentOfBase=percentValue  
repositoryUserLabel="repositoryName"  
repositoryFullPolicy=(failBaseWrites | failSnapShot)  
trayLossProtect=(TRUE | FALSE)]
```

This command creates a snapshot volume. This command enables users to define which physical disks will be used for the snapshot repository volume.

```
create snapshotVolume baseVolume="baseVolumeName"  
[repositoryVolumeGroup=volumeGroupNumber  
repositoryFreeCapacityArea=freeCapacitySize]  
[userLabel="snapshotVolumeName"  
warningThresholdPercent=percentValue  
repositoryPercentOfBase=percentValue  
repositoryUserLabel="repositoryName"  
repositoryFullPolicy=(failBaseWrites | failSnapShot)  
trayLossProtect=(TRUE | FALSE)]
```

This command creates a snapshot volume. This command enables users to define the volume group in which the snapshot repository volume will be located.

```
create snapshotVolume baseVolume="baseVolumeName"  
[repositoryRAIDLevel=(1 | 3 | 5)  
repositoryDriveCount=numberOfDrives  
driveType=(fibre | SATA | PATA) userLabel="snapshotVolumeName"  
warningThresholdPercent=percentValue  
repositoryPercentOfBase=percentValue  
repositoryUserLabel="repositoryName"  
repositoryFullPolicy=(failBaseWrites | failSnapShot)  
trayLossProtect=(TRUE | FALSE)]
```

This command creates a snapshot volume. This command enables users to define the number of physical disks for the snapshot volume. The controller firmware chooses which physical disks to use for the snapshot volume.

```
create volumeCopy source="sourceName" target="targetName"
[copyPriority=(highest | high | medium | low | lowest)
targetReadOnlyEnabled=(TRUE | FALSE)]
```

This command creates a volume copy and starts the volume copy operation.

Important:

You can have a maximum of eight volume copies in progress at one time. If you try to create more than eight volume copies at one time, the controllers will return a status of “Pending” until one of the volume copies that is in progress finishes and returns a status of complete.

deactivate

```
deactivate storageArray feature=remoteMirror
```

This command deactivates the Remote Volume Mirror feature and tears down the mirror repository volume. The host port dedicated to the Remote Volume Mirror is made available for host I/O activity.

delete

```
delete host [hostName]
```

This command deletes a host.

```
delete hostGroup [hostGroupName]
```

This command deletes a host group.

```
delete hostPort [hostPortName]
```

This command deletes a host port.

```
delete (allVolumes | volume [volumeName] |
volumes [volumeName1 ... volumeNameN])
```

This command deletes one or more standard volumes or snapshots and snapshot repository volumes.

Caution:

Potential storage array configuration damage. All data in the volume is lost as soon as you run this command.

```
delete volumeGroup [volumeGroupNumber]
```

This command deletes an entire volume group and its associated volumes.

Caution:

Potential storage array configuration damage. All data in the volume group is lost as soon as you run this command.

diagnose

```
diagnose controller [(a | b)] loopbackDriveChannel=(allchannels
| (1 | 2 | 3 | 4))
testID=(1 | 2 | 3) [patternFile="filename"]
```

This command runs diagnostic tests on the controller. The diagnostic tests consist of loop back tests in which data is written to physical disks and read from the physical disks.

```
diagnose remoteMirror (primary [primaryVolumeName] | primaries
[primaryVolumeName1 ... primaryVolumeNameN]) testID=connectivity
```

This command tests the connection between the specified primary volumes and mirror volumes on a storage array with the Remote Volume Mirror feature installed.

disable

```
disable storageArray feature=(storagePartition2 |
storagePartition4 | storagePartition8 | storagePartition16 |
storagePartition64 | storagePartitionMax | snapshot |
remoteMirror | volumeCopy)
```

This command disables a storage array feature. Issue the show storageArray command to display a list of the feature identifiers for all enabled features in the storage array.

download

```
download drive [trayID,slotID]firmware file="filename"
```

This command downloads a firmware image to a physical disk.

```
download (allTrays | tray [trayID]) firmware file="filename"
```

This command downloads tray firmware.

```
download storageArray driveFirmware file="filename"
[file="filename2"...file="filenameN"]
```

This command downloads firmware images to all physical disks in the storage array.

```
download storageArray firmware [, NVSRAM ] file="filename" [,
"NVSRAM-filename"] [downgrade=(TRUE|FALSE)] [activateNow=(TRUE|
FALSE)]
```

This command downloads firmware and, optionally, NVSRAM values for the storage array controller. If you want to download only NVSRAM values, use the download storageArray NVSRAM command.

```
download storageArray NVSRAM file="filename"
```

This command downloads NVSRAM values for the storage array controller.

enable

```
enable controller [(a | b)] dataTransfer
```

This command revives a controller that has become quiesced while running diagnostics.

```
enable storageArray feature file="filename"
```

This command enables a feature using a feature key file.

recopy

```
recopy volumeCopy target [targetName] [source [sourceName]]
[copyPriority=(highest | high | medium | low | lowest)
targetReadOnlyEnabled=(TRUE | FALSE)]
```

This command reinitiates a volume copy operation using an existing volume copy pair.

recover

```
recover volume (drive=(trayID,slotID) |
drives=(trayID1,slotID1 ... trayIDn,slotIDn) |
volumeGroup=volumeGroupNumber userLabel="volumeName"
capacity=volumeCapacity offset=offsetValue
raidLevel=(0 | 1 | 3 | 5) segmentSize=segmentSizeValue
[owner=(a | b) readAheadMultiplier=multiplierValue]
```

This command creates a RAID volume with the given properties without initializing any of the user data areas on the disks. Parameter values are derived from the Recovery Profile data file for the storage array.

recreate

```
recreate storageArray mirrorRepository
repositoryRAIDLevel=(1 | 3 | 5)
repositoryDrives=(trayID1,slotID1 ... trayIDn,slotIDn)
[trayLossProtect=(TRUE | FALSE)]
```

This command creates a new Remote Volume Mirror repository volume using the parameters defined for a previous Remote Volume Mirror repository volume. The underlying requirement is that you have previously created a Remote Volume Mirror repository volume. This command enables users to define which physical disks will be used for the Remote Volume Mirror repository volume.

```
recreate storageArray mirrorRepository
repositoryRAIDLevel=(1 | 3 | 5)
repositoryVolumeGroup=volumeGroupName
[repositoryFreeCapacityArea=freeCapacityValue
trayLossProtect=(TRUE | FALSE)]
```

This command creates a new Remote Volume Mirror repository volume using the parameters defined for a previous Remote Volume Mirror repository volume. The underlying requirement is that you have previously created a Remote Volume Mirror repository volume. This command enables users to define the volume group in which the Remote Volume Mirror repository volume will be located.

```
recreate storageArray mirrorRepository
repositoryRAIDLevel=(1 | 3 | 5)
repositoryDriveCount= numberOfDrives
[driveType=(fibre | SATA | PATA) trayLossProtect=(TRUE | FALSE)]
```

This command creates a new Remote Volume Mirror repository volume using the parameters defined for a previous Remote Volume Mirror repository volume. The underlying requirement is that you have previously created a Remote Volume Mirror repository volume. This command enables users to define the number of physical disks to be used for the Remote Volume Mirror repository volume.

```
recreate snapshot (volume [volumeName] |
volumes [volumeName1 ... volumeNameN])
[userLabel= "snapshotVolumeName"
warningThresholdPercent=percentValue
repositoryFullPolicy=(failBaseWrites | failSnapShot)]
```

This command starts a fresh copy-on-write operation using an existing snapshot volume.

remove

```
remove remoteMirror (localVolume [volumeName] |
localVolumes [volumeName1 ... volumeNameN])
```

This command removes the mirror relationship between the primary volume and secondary volume.

```
remove volumeCopy target [targetName] [source [sourceName]]
```

This command removes a volume copy pair.

```
remove (allVolumes | volume [volumeName] |
volumes [volumeName1 ... volumeNameN] | accessVolume) lunMapping
(host= "hostName" | hostGroup= ("hostGroupName" | defaultGroup))
```

This command removes the logical unit number mapping.

repair

```
repair volume [volumeName] parity parityErrorFile=filename
[verbose=(TRUE | FALSE)]
```

This command repairs the parity errors on a volume.

reset

```
reset controller [(a | b)]
```

This command resets a controller.

Important:

When you reset a controller, the controller is not available for I/O operations until the reset is complete. If a host is using volumes owned by the controller being reset, the I/O directed to the controller will be rejected. Before resetting the controller, either verify that the volumes owned by the controller are not in use or ensure a multi-path driver is installed on all hosts using these volumes.

```
reset storageArray batteryInstallDate [controller=(a | b)]
```

This command resets the age of the batteries in a storage array to zero days. You can reset the batteries for an entire storage array or the battery in a specific controller.

```
reset storageArray RLSBaseline
```

This command resets the Read Link Status (RLS) baseline for all devices.

```
reset storageArray volumeDistribution
```

This command reassigns (moves) all volumes to their “preferred” controller.

resume

```
resume remoteMirror (primary [volumeName] |
primaries [volumeName1 ... volumeNameN])
writeConsistency=(TRUE | FALSE)
```

This command resumes a suspended Remote Volume Mirror operation.

revive

```
revive drive [trayID,slotID]
```

This command forces the specified physical disk to the optimal state.

```
revive volumeGroup [volumeGroupNumber]
```

This command forces the specified volume group and associated failed physical disks to the optimal state.

save

`save controller [(a | b)] NVSRAM file="filename"`

This command saves a copy of the controller NVSRAM values to a file. This command saves all regions.

`save allDrives logFile="filename"`

This command saves the log sense data to a file. Log sense data is maintained by the storage array for each physical disk.

`save storageArray configuration file="filename"
[(allconfig | globalSettings=(TRUE | FALSE)
volumeConfigAndSettings=(TRUE | FALSE)
hostTopology=(TRUE | FALSE) lunMappings=(TRUE | FALSE))]`

This command creates a script file that you can use to create the current storage array volume configuration.

`save storageArray (allEvents | criticalEvents) file="filename"
[count=numberOfEvents]`

This command saves events from the major event log (MEL) to a file. You can save either all the events or only the critical events.

`save storageArray performanceStats file="filename"`

This command saves the performance statistics to a file. Before you use this command, issue the `set session performanceMonitorInterval` and `set session performanceMonitorIterations` commands to specify how often statistics are collected.

`save storageArray RLSCounts file="filename"`

This command saves the RLS counters to a file. Before using this command, issue the `reset storageArray RLSBaseline` command to get current data.

`save storageArray stateCapture file="filename"`

This command saves the state capture to a file.

`save storageArray supportData file="filename"`

This command saves the support related information to a file.

set

```
set controller [(a | b)]
availability=(online | offline | serviceMode)
bootp gatewayIPAddress=ipAddress
globalNVSramByte [nvramOffset]=(nvramByteSetting |
nvramBitSetting)
hostNVSramByte [hostType, nvramOffset]=(nvramByteSetting |
nvramBitSetting)
ipAddress=ipAddress rloginEnabled=(TRUE | FALSE)
subnetMask=ipAddress
```

This command defines the properties for the controllers

```
set driveChannel [(1 | 2 | 3 | 4)] status=(optimal | degraded)
```

This command defines how the physical disk channel performs.

```
set (drive [trayID,slotID] |
drives [trayID1,slotID1 ... trayIDn,slotIDn])
hotSpare=(TRUE | FALSE)
```

This command assigns or unassigns one or more physical disks as a hot spare.

```
set drive [trayID,slotID] operationalState=failed
```

This command sets a physical disk to the failed state. (To return a physical disk to the optimal state, use the `revive drive` command.)

```
set host [hostName] hostGroup=( "hostGroupName" | none |
defaultGroup) userLabel= "newHostName"
```

This command assigns a host to a host group or moves a host to a different host group. You can also create a new host group and assign the host to the new host group with this command. The actions performed by this command depend on whether the host has individual volume-to-LUN mappings or does not have individual volume-to-LUN mappings.

```
set hostChannel [(a1 | a2 | b1 | b2)] preferredID=portID
```

This command defines the loop ID for the host channel.

```
set hostGroup [hostGroupName] userLabel= "newHostGroupName"
```

This command renames a host group.

```
set hostPort [portLabel] host= "hostName"
hostType=(hostTypeIndexLabel | hostTypeIndexNumber)
userLabel= "newPortLabel"
```

This command changes the host type for a host port. You can also change a host port label with this command.

```
set remoteMirror (localVolume [volumeName] |  
localVolumes [volumeName1 ... volumeNameN])  
role=(primary | secondary) [force=(TRUE | FALSE)]  
syncPriority=(highest | high | medium | low | lowest)  
writeOrder=(preserved | notPreserved)  
writeMode=(synchronous | asynchronous)
```

This command defines the properties for a Remote Volume Mirror pair.

```
set session errorAction=(stop | continue)  
password="storageArrayPassword"  
performanceMonitorInterval=intervalValue  
performanceMonitorIterations=iterationValue
```

This command defines how you want the current script engine session to run.

```
set (volume [volumeName] |  
volumes [volumeName1 ... volumeNameN])  
userLabel="snapshotVolumeName"  
warningThresholdPercent=percentValue  
repositoryFullPolicy=(failBaseWrites | failSnapShot)
```

This command defines the properties for a snapshot volume and enables you to rename a snapshot volume.

```
set storageArray cacheBlockSize=cacheBlockSizeValue  
cacheFlushStart=cacheFlushStartSize  
cacheFlushStop=cacheFlushStopSize  
defaultHostType=("hostTypeName" | hostTypeIdentifier)  
failoverAlertDelay=delayValue mediaScanRate=(disabled | 1-30)  
password="password" userLabel="storageArrayName"
```

This command defines the properties of the storage array.

```
set storageArray time
```

This command sets the clocks on both controllers in a storage array by synchronizing the controller clocks with the clock of the host from which you issue this command.

```
set storageArray trayPositions=(controller | 0 - 99)
```

This command defines the position of all modules in the storage array. All modules in the storage array must be included in the definition list.

```

set (allVolumes | volume [volumeName] |
volumes [volumeName1 ... volumeNameN] | volume <wwn>)
cacheFlushModifier=cacheFlushModifierValue
cacheWithoutBatteryEnabled=(TRUE | FALSE)
mediaScanEnabled=(TRUE | FALSE)
mirrorCacheEnabled=(TRUE | FALSE)
modificationPriority=(highest | high | medium | low | lowest)
owner=(a | b) readCacheEnabled=(TRUE | FALSE)
writeCacheEnabled=(TRUE | FALSE)
readAheadMultiplier=integer-literal

```

This command defines the properties for a volume. This command is applicable to one or more volumes.

```

set (volume [volumeName] | volume <wwn>)
addCapacity=volumeCapacity
[addDrives=(trayID1,slotID1 ... trayIDn,slotIDn)]
redundancyCheckEnabled=(TRUE | FALSE)
segmentSize=segmentSizeValue userLabel=volumeName

```

This command defines the properties for a volume. This command is applicable to only one volume.

```

set (volume [volumeName] | volume <wwn> | accessVolume)
logicalUnitNumber=LUN (host="hostName" |
hostGroup=("hostGroupName" | defaultGroup))

```

This command defines the properties for a volume. This command is applicable to volume mapping.

```

set volumeCopy target [targetName] [source [sourceName]]
copyPriority=(highest | high | medium | low | lowest)
targetReadOnlyEnabled=(TRUE | FALSE)

```

This command defines the properties for a volume copy pair.

```

set volumeGroup [volumeGroupNumber]
addDrives=(trayID1,slotID1 ... trayIDn,slotIDn)
raidLevel=(0 | 1 | 3 | 5) owner=(a | b)
availability=(online | offline)

```

This command defines the properties for a volume group.

show

```
show storageArray autoConfiguration
[driveType=(fibre | SATA | PATA) raidLevel=(0 | 1 | 3 | 5)
volumeGroupWidth=numberOfDrives
volumeGroupCount=numberOfVolumeGroups
volumesPerGroupCount=numberOfVolumesPerGroup
hotSpareCount=numberOfHotspares segmentSize=segmentSizeValue
readAheadMultiplier=multiplierValue]
```

This command displays the default auto configuration that the storage array will create if you issue the `autoConfigure storageArray` command. If you want to determine whether the storage array can support specific properties, enter the parameter for the properties when you issue this command. You do not, however, need to enter any parameters for this command to return configuration information. If you do not specify any properties, this command returns the RAID 5 candidates for each physical disk type. If RAID 5 candidates are not available, this command returns candidates for RAID 3, RAID 1, or RAID 0. When you specify auto configuration properties, the controllers will validate that the firmware will support the properties.

```
show (allControllers | controller [(a | b)]) [summary]
```

For each controller in a storage array, this command returns status, physical characteristics, and configuration information.

```
show (allControllers | controller [(a | b)]) NVSRAM
[hostType=(hostTypeIndexLabel | host="hostName")]
```

This command returns the NVSRAM bytes for the specified host type. If users do not enter the optional parameters, this command returns the entire NVSRAM.

```
show (allDrives [driveType=(fibre | SATA | PATA)] |
drive [trayID,slotID] |
drives [trayID1,slotID1 ... trayIDn,slotIDn]) [summary]
```

For each physical disk in the storage array, this command returns total number of physical disks, type of physical disk (Fibre, SATA, PATA), basic physical disk information, physical disk channel information, hot spare coverage, details for each physical disk.

```
show (driveChannel [(1 | 2 | 3 | 4)] |
driveChannels [(1 | 2 | 3 | 4) ... (1n | 2n | 3n | 4n)] |
allDriveChannels) stats
```

This command displays cumulative physical disk channel I/O and error information. If the controller has automatically degraded a channel, this command also displays interval statistics. When using this command, you can display information about one specific physical disk channel, several physical disk channels, or all physical disk channels.

```
show allDrives downloadProgress
```

This command returns the status of firmware downloads for the physical disks targeted by the `download drive firmware` or `download storageArray driveFirmware` commands.

```
show allHostPorts
```

For all host ports connected to a storage array, this command returns host port identifier, host port name, and host type.

```
show remoteMirror candidates primary="volumeName"
remoteStorageArrayName= "storageArrayName"
```

This command returns information about the candidate volumes on the remote storage array that you can use as secondary volumes for a primary volume.

```
show remoteMirror (localVolume ["volumeName"] |
localVolumes ["volumeName1" ... "volumeNameN"])
synchronizationProgress
```

This command returns the progress of data synchronization between the primary volume and secondary volume in a Remote Volume Mirror. This command displays the progress as a percentage of data synchronization that has been completed.

```
show storageArray profile batteryAge defaultHostType
healthStatus hostTypeTable hotSpareCoverage features time
volumeDistribution [summary]
```

This command returns configuration information about the storage array. The parameters return lists of values for the components and features in the storage array. You can enter the command with a single parameters or more than one parameter. If you enter the command without any parameters, the entire storage array profile is displayed (which is the same information as if you entered the `profile` parameter).

```
show storageArray hostTopology
```

This command returns storage partition topology, host type labels, and host type index for the host storage array.

```
show storageArray lunMappings [host ["hostName"] |  
hostgroup ["hostGroupName"]]
```

This command returns information from the array profile about the storage array LUN mappings. Default group LUN mappings are always displayed. If you run this command without any parameters, this command returns all LUN mappings.

```
show storageArray unreadableSectors
```

This command returns a table of the addresses of all sectors in the storage array that cannot be read.

```
show "string"
```

This command shows a string of text from a script file. This command is similar to the echo command in MS DOS and UNIX.

```
show (allVolumes | volume [volumeName] |  
volumes [volumeName1 ... volumeNameN]) [summary]
```

This command returns detailed information about all volumes in a storage array.

```
show volume ["volumeName"] actionProgress
```

For a long-running operation that is currently running on a volume, this command returns the volume action and percentage completed of the operation.

```
show volumeCopy (allVolumes | source ["sourceName"] | target  
["targetName"])
```

This command returns information about volume copy operations. You can retrieve information about a specific volume copy pair or all volume copy pairs in the storage array.

```
show volumeCopy sourceCandidates
```

This command returns information about the candidate volumes that you can use as the source for a volume copy operation.

```
show volumeCopy source ["sourceName"] targetCandidates
```

This command returns information about the candidate volumes that you can use as the target for a volume copy operation.

```
show volumeGroup [volumeGroupName]
```

For a volume group, this command returns status (online or offline), drive type (Fibre, SATA, or PATA), tray loss protection (yes or no), current owner (controller slot A or slot B), associated volumes and free capacity, and associated physical disks (drives).

Note:

You can use the free capacity area value when creating a volume based on the free capacity of a volume group.

```
show (allVolumes | volume [volumeName] |
volumes [volumeName1 ... volumeNameN] performanceStats
```

This command returns information about the performance of the volumes in a storage array.

```
show (allVolumes | volume [volumeName] |
volumes [volumeName1 ... volumeNameN]) reservations
```

This command returns information about the volumes that have reservations.

start

```
start driveChannel [(1 | 2 | 3 | 4)]locate
```

This command identifies the physical disk modules connected to a specific physical disk channel by turning on the indicator lights for the physical disk module connected. (Use the `stop driveChannel locate` command to turn off the physical disk tray indicator lights.)

```
start drive [trayID,slotID] initialize
```

This command starts physical disk initialization.

Caution:

Potential storage array configuration damage. This command will destroy user data.

```
start drive [trayID,slotID] locate
```

This command locates a physical disk by turning on the physical disk indicator lights. (Use the `stop drive locate` command to turn off the physical disk indicator light.)

```
start drive [trayID,slotID] reconstruct
```

This command starts reconstructing a physical disk.

```
start remoteMirror primary ["volumeName"] synchronize
```

This command starts Remote Volume Mirror synchronization.

```
start storageArray locate
```

This command locates a storage array by turning on the indicator lights for the storage array. (Use the `stop storageArray locate` command to turn off the indicator lights for the storage array.)

```
start tray [trayID] locate
```

This command locates a module by turning on the indicator lights. (Use the `stop tray locate` command to turn off the indicator lights for the module.)

```
start volumeGroup [volumeGroupName] defragment
```

This command starts a defragment operation on the specified volume group.

Note:

Defragmenting a volume group starts a long running operation that you cannot stop.

```
start volume [volumeName] initialize
```

This command starts the formatting of a volume in a storage array.

Note:

Initializing a volume is a long-running operation that you cannot stop.

stop

```
stop driveChannel locate
```

This command turns off the physical disk tray indicator lights that were turned on by the `start driveChannel locate` command.

```
stop snapshot (volume [volumeName] |  
volumes [volumeName1 ... volumeNameN])
```

This command stops a copy-on-write operation.

```
stop storageArray driveFirmwareDownload
```

This command stops a firmware download to the physical disks in a storage array that was started with the `download storageArray driveFirmware` command. This command does not stop a firmware download that is already in progress to a physical disk; however, this command stops all firmware downloads to physical disks that are waiting for the download.

```
stop storageArray locate
```

This command turns off the storage array indicator lights that were turned on by the `start storage array locate` command.

```
stop tray locate
```

This command turns off the tray indicator lights that were turned on by the start tray locate command.

```
stop volumeCopy target [targetName] [source [sourceName]]
```

This command stops a volume copy operation.

```
suspend remoteMirror (volume [volumeName] |  
volumes [volumeName1 ... volumeNameN])  
writeConsistency=(TRUE | FALSE)
```

This command suspends a Remote Volume Mirror operation.

Commands Listed by Function

Controller Commands

Clear Drive Channel Statistics

Diagnose Controller

Enable Controller

Reset Controller

Save Controller NVSRAM

Set Controller

Set Drive Channel Status

Set Host Channel

Show Controller

Show Controller NVSRAM

Show Drive Channel Stats

Start Drive Channel Locate

Stop Drive Channel Locate

Host Topology Commands

Create Host

Create Host Group

Create Host Port

Delete Host

Delete Host Group

Delete Host Port

Set Host

Set Host Group

Set Host Port

Show Host Ports

Physical Disk (Drive) Commands

- Download Drive Firmware
- Revive Drive
- Save Drive Log
- Set Drive Hot Spare
- Set Drive State
- Show Drive
- Show Drive Download Progress
- Start Drive Initialize
- Start Drive Locate
- Start Drive Reconstruction
- Stop Drive Locate

Remote Volume Mirror Commands

- Activate Remote Volume Mirroring Feature
- Create Remote Volume Mirror
- Deactivate Remote Volume Mirror
- Diagnose Remote Mirror
- Recreate Remote Volume Mirroring Repository
- Remove Remote Volume Mirror
- Resume Remote Volume Mirror
- Set Remote Volume Mirror
- Show Remote Volume Mirror Volume Candidates
- Show Remote Volume Mirror Volume Synchronization Progress
- Start Remote Volume Mirror Synchronization
- Suspend Remote Volume Mirror

Session Command

- Set Session

Snapshot Commands

Create Snapshot Volume

Recreate Snapshot

Set Snapshot Volume

Stop Snapshot

Storage Array Commands

Activate Storage Array Firmware

Autoconfigure Storage Array

Autoconfigure Storage Array Hot Spares

Clear Storage Array Configuration

Clear Storage Array Event Log

Clear Storage Array Firmware Pending Area

Disable Storage Array Feature

Download Storage Array Drive Firmware

Download Storage Array Firmware/NVSRAM

Download Storage Array NVSRAM

Enable Storage Array Feature Key

Reset Storage Array Battery Install Date

Reset Storage Array RLS Baseline

Reset Storage Array Volume Distribution

Save Storage Array Configuration

Save Storage Array Events

Save Storage Array Performance Statistics

Save Storage Array RLS Counts

Save Storage Array State Capture

Save Storage Array Support Data

Set Storage Array

Set Storage Array Time

Set Storage Array Tray Positions
Show Storage Array Auto Configure
Show Storage Array
Show Storage Array Host Topology
Show Storage Array LUN Mappings
Show Storage Array Unreadable Sectors
Start Storage Array Locate
Stop Storage Array Drive Firmware Download
Stop Storage Array Locate

Module Commands

Download Environmental Card Firmware
Start Tray Locate
Stop Tray Locate

Uncategorized Command

Show String

Volume Commands

Check Volume Parity
Clear Volume Reservations
Clear Volume Unreadable Sectors
Create RAID Volume (Automatic Drive Select)
Create RAID Volume (Manual Drive Select)
Create RAID Volume (Free Capacity Base Select)
Delete Volume
Recover RAID Volume
Remove Volume LUN Mapping
Repair Volume Parity

- Set Volume
- Show Volume
- Show Volume Action Progress
- Show Volume Performance Statistics
- Show Volume Reservations
- Start Volume Initialization

Volume Copy Commands

- Create Volume Copy
- Recopy Volume Copy
- Remove Volume Copy
- Set Volume Copy
- Show Volume Copy
- Show Volume Copy Source Candidates
- Show Volume Copy Target Candidates
- Stop Volume Copy

Volume Group Commands

- Delete Volume Group
- Revive Volume Group
- Set Volume Group
- Show Volume Group
- Start Volume Group Defragment