Command Line Interface

Reference Guide

adaptec®

Contents

1 Introduction

2 General Control Commands

3 container Commands

4 controller Commands

5 disk Commands

1

Introduction

In this Chapter

The CLI provides a command line alternative to Adaptec Storage Manager. Through the CLI, you perform most of the storage management tasks that you can perform with the Adaptec Storage Manager GUI and, in addition, some tasks not available in Storage Manager. With CLI, you can also use the CLI commands in DOS command scripts and UNIX shell scripts.

Audience

This *Reference Guide* is for system administrators and experienced users who are familiar with drive configuration and who have a general understanding of the operating systems they are using. This guide also assumes you are familiar with RAID concepts.

Accessing the CLI

This section discusses accessing the CLI in various operating systems.

Accessing the CLI from the MS-DOS Prompt

To access the CLI from the MS-DOS prompt, move to the directory containing the `afacli` executable and then type `afacli`, as in the following example:

```
C:\>cd Program Files\Adaptec_SCSI_RAID\AAC

C:\Program Files\Adaptec_SCSI_RAID\AAC>afacli
```

Accessing the CLI in Windows

This section discusses the various methods for accessing the CLI in Windows.

Note: The following procedure assumes that you accepted the default location for the software during installation.

To access the CLI:

1. Click the **Start** button and move the mouse cursor to **Programs**. Click on **Programs**.
2. Move the mouse cursor to **SMBE**. The Adaptec program group will display.
3. Move the mouse cursor to the **CLI** icon. Click on **CLI**.

Accessing CLI in Linux and UNIX

To access the CLI from the UNIX/Linux prompt, display a window and type `aaccli` in any directory. When the system displays the `CLI>` prompt, you can use CLI commands.

For the commands to work in any directory, the path in the startup file (`.login` or `.cshrc`) must include the directory where the software is installed. See your UNIX/Linux documentation for information on setting up directory paths in the `.login` and `.cshrc` files.

Accessing CLI in NetWare

To access the CLI from the NetWare server console, type `aaccli` at the prompt. When the system displays the `CLI>` prompt, which indicates that you can now use CLI commands.

Terminology

This section discusses terminology used in this *Appendix*.

Adaptec 2410SA/2810SA Controllers Only

In the device ID format `C:ID:L`:

- C represents Channel, which is always zero.
- ID represents Port ID, or Port number, on the SATA RAID controller. This is the only value that is used.
- L represents LUN, which is always zero.

Note: Only the value for Port ID is used.

Adaptec 21610SA Controllers Only

In the device ID format `C:ID:L`:

- C represents Channel.
- ID represents Port ID. (For mapping, see below.)
- L represents LUN, which is always zero.

Your Serial ATA adapter maps Port IDs as follows:

- Controller ports 0 through 7 are mapped to IDs 0 through 7 on Channel 0
- Controller ports 8 through 15 are mapped to IDs 0 through 7 on Channel 1

For example, port 5 is 0:5:0; port 12 is 1:4:0.

All Controllers

The following terms are used in discussing the CLI:

- **Array, container**—A logical disk created from available space and made up of one or more partitions on one or more physical disks.

- **Stripe, chunk**—Contiguous set of data distributed across all the disks in an array. A striped array distributes data evenly across all members in equal-sized sections called *stripes*.
- **Free or available space**—Space on an initialized disk that is not in use.
- **Partition**—Contiguous area of a physical drive that makes up some or all of an array.
- **Hotspare, failover disk**—A hard drive, an array member, that is not used in day-to-day data storage, but is instead reserved for use as a replacement for one of the other drives in the array in the event of a failure.

Conventions

This section discusses conventions important to the interface.

- **Case sensitivity**—CLI syntax elements are not case-sensitive except for strings.
- **Command abbreviations**—You can abbreviate commands, subcommands, and switches. You must provide enough characters so the commands are not ambiguous.
- **Subcommand Mode**—Subcommand mode simplifies command entry when you are performing repetitive commands on the same item. Subcommand mode works only for commands that have one or more subcommands.

To enter subcommand mode, type the first part of any command that has subcommands and press **Enter**. The command you entered replaces the CLI> prompt and becomes the first part of any subsequent command you enter.

For example, to use subcommand mode for the `array` command:

1. Type `array` at the CLI> prompt, and press **Enter**.

   The CLI> prompt changes to `_array >`.

2. Enter valid subcommands, switches, and parameters for any `array` command and press **Enter**. For example, to change the number of array 1 to 2, simply type the following at the `_array >` prompt and press **Enter**:

   ```
   move 1 2
   ```

To exit subcommand mode, press **Enter**. (You may have to press **Enter** more than once to exit subcommand mode.)

- **CLI-unique commands**

  For NetWare systems, the following commands are unique to the CLI in that you use these commands when accessing the CLI from the NetWare server console:

  - `open`
  - `close`
  - `exit`
  - `toggle_more`

- **Comments**—If the first non-blank character on a line in the CLI is an exclamation point (!), the rest of the line will be ignored. You can type comments after the exclamation point. This is particularly useful in scripts where you want to comment out sections of code.

Command Syntax

You can enter only valid CLI commands at the command line prompt.

All commands use the following syntax:

```
[class] action [object] [/switch1[=value1]
[/switch2[=value2]...]] [parameter1 [parameter2...]]
```

The CLI syntax uses the following conventions:

`[text]`	The text element within brackets is optional.
`{type}`	The type specified within braces is the value type (for example, boolean, integer, string, and so on).
`parameter{type}`	The parameter defined by the value type.
`[text...]`	Two or more text elements.

Class

CLI commands are grouped according to *class*. Classes currently supported include array, controller, diagnostic, disk, logfile, task, and enclosure and are required except in subcommand mode.

Action

An *action* specifies an operation to be performed by the CLI, such as open, create, list, exit, or show. An action is a required element, except in subcommand mode. In most cases, actions are preceded by a class and followed by an object, switches, and parameters.

Object

An *object* describes what to use in an action. Examples include mirror, volume, mstripe, and drive_letter.

Switch

A *switch* is an element applied to a command that allows a variation of the command. Switches are always optional.

If you specify a switch, it may require a value. Switches that are Boolean do not require a value when the switch is used because the switch defaults to `TRUE` or `FALSE`. Optionally, you can explicitly set a Boolean switch to `TRUE` or `FALSE`.

Value

A *value* is a type that applies to a switch. For each switch, you can have only *one* type of value. For example, a switch that is defined as a string value can never take an integer value. A space may be used within a value only within a quoted string. Examples of valid value types include Boolean, integer, string, `scsi_device`, `free_space`, and `array`.

You delimit a value that applies to a switch by using an equal sign, as in the following example:

```
AAC0>disk list /all =TRUE
Executing: disk list /all=TRUE
```

Note that a space to the left or right of the equal sign is permitted.

Parameter

Although some commands have optional parameters, *parameters* are usually mandatory. You must place optional parameters after mandatory parameters at the end of the command. For example, the `array remove failover` command has one required `scsi_device` parameter and additional optional `scsi_device` parameters: `array remove failover {array} {scsi_device} [{scsi_device}...]`.

As with switch values, parameters can be of only *one* type. For example, the `array` parameter can take only an integer value.

All parameters are separated by blank space. A space can be used in a parameter only within a quoted string.

Blank Space

Blank space is one or more spaces or tabs.

Parameter and Switch Value Types

A parameter or switch value can be only one of several types. Simple value types include string, Boolean, and integer. Other value types include objects that can be manipulated, such as arrays (or arrays) and SCSI devices. This section explains the CLI parameter and switch value types.

Boolean

A *boolean* has a value of either `TRUE` or `FALSE` and can be specified with any of the following key words:

- `TRUE` can be specified with `TRUE`, `1`, `ON`, or `YES`. For switches, you can also specify `TRUE` by using the switch without any switch value. For example, `/readonly` and `/readonly=TRUE` are identical.
- `FALSE` can be specified with `FALSE`, `0`, `OFF`, or `NO`. When a switch is not specified, it takes the default value (usually, `FALSE`) specified in the switch description section of a command. For example, `open aac0` is the same as open `/readonly=FALSE aac0`.

integer

An *integer* is a positive or negative number that has a value between -2^{63} and $(2^{63} - 1)$. Although the valid range is usually much smaller, all integers have 64 bits of precision and do not contain decimal points.

You can specify an integer as a mathematical equation that uses an asterisk (*) to multiply, a plus sign (+) to add, a minus sign (–) to subtract, a slash (/) to divide, and parentheses [( )] to specify order of operations. If you do not use parentheses, all operations are completed left to right. No spaces are allowed in the expression.

Numbers in an equation that:

- Begin and end with no suffix are decimal
- Begin with `0x` or end with `h` are hexadecimal
- End with `o` indicate octal
- End with `z` indicate binary

You can attach special multipliers to the end of any number to allow for easy translation to reasonable disk sizes. Table 1-1 lists the letters and their multiplicative values:

Table 1-1 Letters and Multiplicative Values

Letter	Action
K (kilobytes)	multiplies by 1024
M (megabytes)	multiplies by 1024*1024
G (gigabytes)	multiplies by 1024*1024*1024
T (terabytes)	multiplies by 1024*1024*1024*1024

You cannot use decimal points. To specify 1.5 GB, for example, you must use (3G/2).

Note: All suffixes are case-insensitive. That is, you can use upper or lower-case characters. For example you can specify 10 M or 10 m.

Table 1-2 lists examples of valid integers and their corresponding values:

Table 1-2 Integers and Values (in Decimal)

Integer	Value (in decimal)
219	219
3*4	12
(5+3*24)	192
(5+(3*24))	77
0x123	291
(12+52h+1010z)	104
100M	104,857,600
2G	2,147,483,648

string

You can specify a text *string* with or without quotation marks. If spaces are needed, however, the string must be specified with quotation marks; otherwise, the CLI interprets spaces as delimiters.

A string can use either double quotes or single quotes but must start and end with the same type of quotes. Also, within a quoted string, a quotation mark of the same type used to surround the string can be specified by repeating that quotation mark twice in the string.

Table 1-3 lists examples of valid strings and their corresponding values.

Table 1-3 Strings and Corresponding Values

String	Value
VOL_set	VOL_set
"VOL label"	VOL label
'Use " mark'	Use " mark
"Use ' mark"	Use ' mark
"Use Both "" and ' mark"	Use Both " and ' mark

scsi_device

Specifies a SCSI device. SCSI device descriptions have three parts: SCSI channel number, SCSI device ID, and SCSI device logical unit number (LUN).

The following syntax defines the `scsi_device` parameter:

```
({integer},{integer},{integer})
```

Each integer corresponds to one component of a SCSI device descriptor or specifier: the first `integer` is the SCSI channel number, the second `integer` is the SCSI device ID, and the third integer is the SCSI device LUN.

The actual values that the CLI supports are as follows:

- SCSI channel number = (for example, 0, 1, 2, 3, and so on). See the installation guide for your controller to determine the actual number of channels it supports.
- SCSI device ID = (0 through 15 inclusive)
- SCSI device LUN = 0 through 7 inclusive

Table 1-4 displays the syntax for SCSI device switch value type abbreviations:

Table 1-4 SCSI Device Switch Abbreviations

Abbreviation	Syntax
SCSI channel number, SCSI device ID	**({integer},{integer})**
SCSI device ID	**({integer})**
SCSI device ID	**{integer}**

Table 1-5 displays valid SCSI device specifiers. The second two examples contain a default LUN number, and the last two examples contain default channel and LUN numbers. You can specify these defaults using the `disk set default` command (see page 5-11).

Table 1-5 SCSI Device Specifiers

Example	SCSI Channel Number, SCSI Device ID, SCSI Device LUN
(1,1,4)	Channel 1, Device 1, LUN 4
(3,2,0)	Channel 3, Device 2, LUN 0
(3,2)	Channel 3, Device 2, LUN default[1]
(0,1)	Channel 0, Device 1, LUN default[1]
(1)	Channel default[1], Device 1, LUN default[1]
1	Channel default[1], Device 1, LUN default[1]

[1] Set by the disk set default command.

free_space

Freespace (also known as *available space*) is specified by a SCSI device and, optionally, a size. If you do not specify a size, parentheses are optional.

Therefore, the following syntaxes are allowed for the `free_space` parameter:

```
({scsi_device},{freespace_size})
({scsi_device})
{scsi_device}
```

If you do not specify the `freespace_size` parameter, it defaults to the size of the first freespace area available on the specified SCSI device. Note that offsets are not specified for freespace. The offset used is the first offset that starts a freespace area large enough to fit `size` bytes.

Table 1-6 lists valid `free_space` specifiers.

Table 1-6 Freespace Specifiers

Freespace Specifier	SCSI Channel Number, SCSI Device ID, SCSI Device LUN; Use *n* Amount of Freespace
`((1,2),1G)`	SCSI Device: Channel 1, Device ID 2, LUNdefault; use 1 GB of freespace
`5`	SCSI Device: Channel default, Device ID 5, LUN default; use all freespace if empty or all of the first freespace available is some is used
`(8,2G)`	SCSI Device: Channel default, Device ID 8, LUN default; use 2 GB of freespace

Note: When specifying freespace during an array create volume operation, you must explicitly specify the SCSI device's channel, device ID, and logical unit number. You cannot use any default values for the SCSI device. In addition, you must also specify the size of the freespace(s).

container

A controller currently supports 24 visible arrays. A visible array is an array that is visible to the operating system and users. Visible arrays are identified with array IDs 0 through 23.

Array IDs 24 through 63 are reserved for hidden arrays. A hidden array is an array that is not visible to the operating system and can only be used by other arrays. (However, the `array list` command displays hidden arrays.)

The Parameters section uses the following syntax to specify an array:

`{array}`

This syntax specifies the ID number (0 to 63 inclusive) of the array.

A controller assigns a unique ID to each of its arrays when you create it.

Status Information

When invoked from the MS-DOS prompt or the Windows Start button, the CLI displays status information in the title bar of a DOS command prompt window as it executes an asynchronous command. On NetWare, the status information appears in a different window. On UNIX systems, the CLI displays this status information at the bottom of the CLI window.

The following example shows the status information for the `array create mirror` command:

```
Stat:OK!Task:101,Func:MCR Ctr:0,State:DNE 100.0%
```

The following sections describe each item that the CLI displays in the title bar.

The Stat and Task Items

The Stat item displays the status of the currently running task. Typically, the item displays the value `OK!` to indicate the task is executing correctly. This is the value that appears in the example.

The Task item displays the ID number associated with a specific task. The controller assigns each task a unique ID number. The task ID that appears in the example is 101.

The Func Item

The Func item displays the type of task running on the controller. Table 1-7 describes the values that the Func item can display.

Table 1-7 Function Values

Value	Meaning
`FSV`	File system verify task.
`FSX`	File system extend task. The `array extend mvolume` and `array extend volume` commands cause the `FSX` value to display.
`FTF`	array format task with a FAT file system specified. The `array format` command with the `/file_system` switch set to FAT causes the `FTF` value to display.
`MCR`	Mirror set create or multilevel mirror set create task. The `array create mirror` and `array create mmirror` commands cause the `MCR` value to display.

Table 1-7 Function Values (Continued)

Value	Meaning
MMR	Merge a broken mirror task. The `array merge` command causes the `MMR` value to display.
MSC	array scrub task. The `array scrub` command causes the `MSC` value to display.
NTF	array format task with an NTFS file system specified. The `array format` command with the `/file_system` switch set to NTFS causes the `NTF` value to display.
R5R	RAID-5 rebuild task.
R5S	RAID 5 array create task with the scrub method specified. The `array create raid5` command with the `/scrub` switch specified causes the `R5S` value to display.
RCF	array reconfigure task. The `array reconfigure` command causes the `RCF` value to display.
SCV	Verify all blocks on a SCSI disk device task. The `disk verify` command causes the `SCV` value to display.
SCZ	Clear an entire SCSI disk task. The `disk zero` command causes the `SCZ` value to display.
SVR	Verify all blocks and repair bad blocks on a SCSI disk device task. The `disk verify` command with the `/repair` switch causes the `SVR` value to display.

The Ctr and State Items

The Ctr item displays the ID number of the array associated with the task. In the example, the array's ID is 0 (zero).

The State item displays the state of the task along with a running percentage value that indicates the progress of the currently running task. The item shows the percentage in tenths of a percent increments. The currently running task is complete when the State item displays 100%.

Table 1-8 describes the state values that the State item can display.

Table 1-8 State Values

State	Meaning
BAD	The task failed and is no longer running.
DNE	The task successfully completed or the task is no longer running (that is, the task is done).
RUN	The task is running.
SUS	The task was suspended. Typically, you suspend a task with the `task suspend` command.
UNK	The controller reported an unknown status for the task.

2

General Control Commands

In this Chapter

The CLI general control commands are discussed in alphabetical order and use the following syntax:

```
command [subcommand] [/switch{=value}]
[parameter]
```

close

To close the currently opened controller when all access is completed, use the `close` command.

Syntax

```
close
```

exit

To close the currently opened controller and exit the CLI, use the `exit` command.

Syntax

```
exit
```

help, ?

To invoke general or topical Help commands, use the `help` command or the `?` (question mark).

Syntax

```
? [{command}]
{command} ?
help [/full] [{command}]
help [/full} {command subset}
```

Parameters

If the command has more subcommands, `help` lists the subcommands and their functions. If a complete command is used, such as `array list`, the CLI `help` displays all possible switches.

Switches

`/full`

Displays all relevant commands along with the command format and all command switches.

history_size

To set the size of the command history buffer, use the `history_size` command.

Command Availability

This command is supported only on UNIX.

Syntax

```
history_size {buffer_size}
```

Parameters

`{buffer_size}`

Specifies the size of the command history buffer. The default size is 200.

open

To open a controller, use the `open` command. The `open` command prepares a particular controller for access by the CLI. If you specify this command when another controller is open during a particular command session, the CLI closes the currently opened controller and then opens the specified controller.

Syntax

```
open [/readonly{=boolean}]
[/domain{=string}] {string}
```

Parameters

`{string}`

Specifies the computer name and the controller you want to open. For the string, use the standard format `\\nodename\AAC`*n*, where *n* is the controller number.

For local controllers, you can omit the `\\nodename` in the string specification.

If the computer name has a dash (-) in the name (for example, proj-athena), enclose the entire string within quotes. For example:

```
"\\proj-athena\aac0"
```

Switches

`/readonly{=boolean}`

Specifies whether to open the controller for read-only access. A value of `TRUE` indicates the CLI opens the controller for read-only access.

If you do not specify this switch, it defaults to `/readonly=FALSE` (which means the CLI opens the controller for read/write access).

Note that if you open the controller with read-only access, you can use only the commands that do not change the controller configuration.

/domain{=string}

Specifies the domain (the local domain or a trusted domain) in which the specified computer that contains the controller resides. If you do not specify this switch, the CLI assumes the local domain.

This switch is supported on Windows only.

Note: You can open controllers for read-write access in any GUI or CLI session only once per controller. Thereafter, any user can open and access the previously opened controller only in read-only mode.

reset_window

To reset the window, use the `reset_window` command.

Command Availability

This command is supported only on UNIX.

Syntax

```
reset_window
```

toggle_more

To turn on or off the <Press any key to continue> functionality, use the `toggle_more` command.

After you type `toggle_more` on the NetWare console, the command displays an appropriate message indicating whether the <Press any key to continue> functionality is on or off.

Command Availability

This command is supported only on NetWare.

Syntax

```
toggle_more
```

3

container Commands

In this Chapter

The array commands are discussed in alphabetical order and use the following syntax:

```
container action [object] [/switch{=value}]
[parameter]
```

Note: The examples in this chapter assume that you have first opened the controller before executing the container command. The initial step of opening the controller is not shown.

container add_level

To create a multilevel volume set from an existing array by adding a volume set on top of the array, use the `container add_level` command. The existing array cannot be a multilevel array.

The `container add_level` command enables dynamic expansion of an array and can be executed even while an array is in use. After executing the command, you can use the `container extend mvolume` command to add more space.

Syntax

```
container add_level {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to convert to a multilevel volume set.

The specified array must not be a multilevel array.

Examples

Before creating a multilevel volume set, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a stripe set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Stripe 45MB       32KB   None    0:02:0 64.0KB:  15.0MB
                                           0:03:0 64.0KB:  15.0MB
                                           0:04:0 64.0KB:  15.0MB
```

The following example creates a multilevel volume set on top of array 0, which is a stripe set previously created with the `container create stripe` command:

```
AAC0>container add_level 0
Executing: container add_level 0
```

Use the `container list` command after using the `container add_level` command to display information about the multilevel volume set, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
0     Volume 45MB              None
   63    Stripe 45MB       32KB          0:02:0 64.0KB:  15.0MB
                                         0:03:0 64.0KB:  15.0MB
                                         0:04:0 64.0KB:  15.0MB
```

The following list describes the change to the display as a result of creating a multilevel volume set with the `container add_level` command:

- The **Num Label** column displays two ID numbers. The first ID (0) is the newly created volume set. The second ID (63) is the original array (the stripe set).

  If you specified a label when creating the multilevel volume set, it appears in this column. Because no label was specified when the multilevel volume set was created, no label appears in the column.

- The **Type** column displays two array types: Volume and Stripe. The Volume array type indicates that the newly created array is a volume set. This is the array created after the `container add_level` command completes.

  The Stripe array type indicates the array whose ID was specified to the `container add_level` command. This is the original array previously created with the `container create stripe` command.

Related Commands

`container` commands:

- `container extend mvolume` (page 3-43)
- `container list` (page 3-50)
- `container promote` (page 3-63)

container create mirror

To create a mirror set from a single-partition volume set and freespace, use the `container create mirror` command. Any data on the original volume set remains intact during mirror creation and the user sees no interruption in service.

If you created a mirror set on a NetWare server, you can run the `list devices` command on the NetWare console to verify its creation and then create the necessary NetWare disk partitions and volumes.

For information on how to create partitions and volumes, see the appropriate NetWare documentation.

Syntax

```
container create mirror [/wait{=boolean}]
[/io_delay{=integer}] {container} {scsi_device}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array on which to create a mirror set. You create a mirror set from a single-partition volume set.

`{scsi_device}`

Specifies the ID for the SCSI device whose freespace you want to use for mirroring the volume set specified in the container parameter. A SCSI ID consists of a SCSI bus number (e.g., 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive). Refer to your controller's *Installation and User's Guide* to determine the number of buses it supports.

For further details, see *scsi_device* on page 1-10.

Switches

`/wait{=boolean}`

Specifies whether to create the mirror set synchronously. If you set this switch to `TRUE`, the command prompt does not return until the mirror-set creation completes. If you set this switch to `FALSE`, the mirror-set creation starts asynchronously and the command prompt returns immediately.

`/io_delay{=integer}`

Specifies the number of milliseconds the controller waits between the I/Os required to create the mirror set. If you do not specify this switch, the I/O delay is always zero (0). The I/O delay value is not preserved between reboots of the operating system.

Examples

Before creating a mirror set, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is one existing array (array 0, a volume set) on this controller prior to the time the mirror set is created:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 10MB              NTFS    0:02:0 64.0KB:  10.0MB
```

The following example shows how to create a mirror set synchronously from volume set 0 using freespace on SCSI device (0,3,0):

```
AAC0>container create mirror /wait /io_delay=10 0 (0,3,0)
Executing: container create mirror /wait=TRUE
/io_delay=10 0
(CHANNEL=0, ID=3,LUN=0)
```

As the command executes, note the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!Task:101,Func:MCR Ctr:0,State:RUN 97.2%
```

For further details on status information, see page 1-13.

Use the `container list` command after the `container create mirror` command completes execution to display information about the mirror set, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ -------  ------ -------------
   0     Mirror 10MB              NTFS     0:02:0 64.0KB:  10.0MB
                                           0:03:0 64.0KB:  10.0MB
```

The following list describes the changes to the display as a result of using the `container create mirror` command:

- The **Type** column displays a new array type, which in the example is Mirror instead of the previously displayed Volume.
- The **State** column displays Normal (instead of a blank) as the mirror state.

(Note that this column appears only if you specify the `/full` switch with the `container list` command.)

- The **Scsi C: ID: L** column displays the SCSI device IDs for the two halves of the mirror set.
- The **Partition Offset:Size** column displays the partition offset and size for the two halves of the mirror set.

  If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

After creating a mirror set, you can manipulate it by using the

- `container split` command to split a mirror set
- `container unmirror` command to unmirror a mirror set

Related Commands

`container` commands:

- `container create mmirror` (page 3-8)
- `container list` (page 3-50)
- `container split` (page 3-109)
- `container unmirror` (page 3-114)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)

container create mmirror

To create a multilevel array of mirror sets from a multilevel array of single-partition volume sets and freespace, use the `container create mmirror` command. Typically, you use this command after promoting a volume set built from multiple partitions with the `container promote` command.

The `container create mmirror` command is fully dynamic. If you use this command, users will not see any change or experience any interruption in service. However, performance may be reduced.

Notes

When creating a multilevel array of mirror sets on a NetWare server, you specify the container IDs (as described in the Parameters section) for the arrays from which you want to create the multilevel mirror set. If NetWare is using any of these arrays, an appropriate message displays. This message indicates that one or more of the arrays you specified is in use by NetWare. The message directs you to remove any NetWare volumes and partitions from these arrays. You can then create a multilevel array of mirror sets on a NetWare server.

After you create a multilevel array of mirror sets you can run the `list devices` command on the NetWare console to verify its creation and then create the necessary NetWare disk partitions and volumes.

For information on how to create partitions and volumes, see the appropriate NetWare documentation.

Syntax

```
container create mmirror [/io_delay{=integer}] [/
wait{=boolean}] {container} {scsi_device}
[{scsi_device}...]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array whose underlying volume sets the command converts to mirror sets. You create a multilevel array of mirror sets from a multilevel array of single-partition volume sets and freespace.

`{scsi_device}`

Specifies the ID for the SCSI device whose freespace you want to use to create the multilevel array of mirror sets. The size of this freespace should be greater than or equal to the size of the first underlying volume set. A SCSI ID consists of a SCSI bus number (e.g., 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive). See the installation guide for your controller to determine the number of buses it actually supports.

For further details, see *scsi_device* on page 1-10.

`{scsi_device}...`

Specifies the IDs for additional SCSI devices whose freespaces you want to use to create the multilevel array of mirror sets. There must be sufficient contiguous space available on each specified device.

The number of SCSI devices you specify must be equal to the number of single-partition volume sets that make up the multilevel array of volume sets. The size of the freespace used is equal to its corresponding underlying volume set.

Switches

`/io_delay{=integer}`

Specifies the number of milliseconds the controller waits between the I/Os required to create the multilevel array of mirror sets. If you do not specify this switch, the I/O delay is always zero (0). The I/O delay value is not preserved between reboots of the operating system.

`/wait{=integer}`

Specifies whether to create the multilevel array of mirror sets synchronously or asynchronously. If you set this switch to `TRUE`, the command creates the multilevel array of mirror sets synchronously and the command prompt does not return until the mirror-set creation task completes.

If you do not specify this switch, the mirror-set creation starts asynchronously and the command prompt returns immediately.

Examples

Before creating a multilevel array of mirror sets, use the `container list` command to obtain information about any existing arrays. As the following example shows, array 0 is a volume set that consists of two underlying volume sets created with the `container promote` command:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ -------- ------ -------------
F: 0     Volume 30MB              NTFS
   63    Volume 15MB                       0:02:0 64.0KB:  15.0MB
   62    Volume 15MB                       0:03:0 64.0KB:  15.0MB
```

The following example shows how to create a multilevel array of mirror sets with the following characteristics:

- The array specified (0) is a multilevel array of volume sets that was previously created with the `container promote` command.
- The freespace from SCSI devices (0,4,0), and (0,5,0) are used to mirror the underlying volume sets.
- The default I/O delay is taken.
- The multilevel array of mirror sets is created asynchronously.

```
AAC0>container create mmirror 0 (0,4,0) (0,5,0)
Executing: container create mmirror 0 (CHANNEL=0,ID=4,LUN=0)
(CHANNEL=0,ID=5,LUN=0)
```

As the command executes, note the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!Task:101,Func:MCR Ctr:63,State:RUN 97.2%
```

For further details on status information, see page 1-13.

Use the `container list` command after using the `container create mmirror` command to display information about the multilevel array of mirror sets:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ -------- ------ -------------
F: 0     Volume 30M               NTFS
   63    Mirror 15MB                       0:02:0 64.0KB:  15.0MB
                                            0:04:0 64.0KB:  15.0MB
   62    Mirror 15MB                       0:03:0 64.0KB:  15.0MB
                                            0:05:0 64.0KB:  15.0MB
```

The following list describes the change to the display as the result of creating a multilevel array of mirror sets with the `container create mmirror` command:

- The **Dr** column is blank to indicate that the newly created mirror sets (63 and 62) do not have a drive letter assigned to them.
- The **Type** column displays Mirror for arrays 63 and 62 to indicate that the underlying arrays are mirror sets.
- The **State** column displays the state of the arrays, which in this example are all in the Normal state.

  Note that this column appears only if you specify the `/full` switch with the `container list` command.
- The **Scsi C:ID:L** column displays the SCSI device ID for the disk(s) on which the underlying mirror set(s) reside. In the example, mirror set 63 resides on disk 0:02:0 and 0:04:0. Mirror set 62 resides on disk 0:03:0 and 0:05:0.
- The **Partition Offset:Size** column displays the partition offset(s) and size(s) for the underlying mirror set(s). In the example mirror set 63 and 62 have partition offsets of 64.0 KB and sizes of 15.0 MB.

  If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container list` (page 3-50)
- `container promote` (page 3-63)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)

container create mstripe

To create a multilevel stripe set from equally sized arrays, use the `container create mstripe` command. The top level of a multilevel stripe array can only be a stripe set. You can create the following types of multilevel stripe set:

- Stripe set of mirror sets
- Stripe set of volume sets
- Stripe set of stripe sets
- Stripe set of RAID 5 arrays (a RAID-50 set)

Notes

When creating a multilevel array of stripe sets on a NetWare server, you specify the container IDs (as described in the Parameters section) for the arrays from which you want to create the multilevel stripe set. If NetWare is using any of these arrays, an appropriate message displays. This message indicates that one or more of the arrays you specified is in use by NetWare. The message directs you to remove any NetWare volumes and partitions from these arrays. You can then create a multilevel array of stripe sets on a NetWare server.

After you create a multilevel array of stripe sets you can run the `list devices` command on the NetWare console to verify its creation and then create the necessary NetWare disk partitions and volumes.

For information on how to create partitions and volumes, see the appropriate NetWare documentation.

Syntax

```
container create mstripe [/stripe_size{=integer}]
[/label{=string}]{container} [{container}...]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array from which you want to create the multilevel stripe set.

`{container}...`

Specifies the ID number or numbers (0 to 63) of one or more arrays from which you want to create the multilevel stripe set. You can use up to 16 arrays to create a multilevel stripe set. All specified arrays must be the same size as the first `container` parameter and separated by blank spaces.

Switches

`/stripe_size{=integer}`

Note: This keyword does not support RAID 50 arrays because RAID 50 arrays support only a stripe size of 64 KB.

Specifies the stripe size for the multilevel stripe set. Valid values are 16 KB, 32 KB, and 64 KB.

If you do not specify the switch, it defaults to 64 KB.

`/label{=string}`

Specifies a label to be assigned to the newly created multilevel stripe set. You can specify a maximum of sixteen characters for the label.

If you do not specify the switch, it defaults to no label. If you do not specify a label, you can do so later by using the `container set label` command.

Examples

Before creating a multilevel stripe set, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there are two existing arrays (mirror sets) on this controller at the time the multilevel stripe set is created. These mirror sets were previously created with the `container create mirror` command.

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Mirror 15MB              None    0:02:0 64.0KB:  15.0MB
                                         0:02:0 15.0MB:  15.0MB
   1     Mirror 15MB              None    0:03:0 64.0KB:  15.0MB
                                         0:03:0 15.0MB:  15.0MB
```

The following example shows how to create a multilevel stripe set from two equally sized arrays, using the default stripe size of 64 KB. In this example, the two equally sized arrays are mirror sets:

```
AAC0>container create mstripe 0 1
Executing: container create mstripe 0 1
container 0 created
```

On UNIX systems, the message displayed after you execute the `container create mstripe` command includes the root special file associated with the newly created multilevel stripe set.

Use the `container list` command after using the `container create mstripe` command to display information about the multilevel stripe set, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Stripe 30MB   0   64KB   None
    63   Mirror 15MB                        0:02:0 64.0KB:  15.0MB
                                            0:02:0 15.0MB:  15.0MB
    62   Mirror 15MB                        0:03:0 64.0KB:  15.0MB
                                            0:03:0 15.0MB:  15.0MB
```

The following list describes the change to the display as the result of creating a multilevel stripe set with the `container create mstripe` command:

- The **Num Label** column displays the ID number (in the example, 0) of the newly created multilevel stripe set and (indented to the right) the ID numbers of the arrays that make up the multilevel stripe set. In the example, arrays 63 and 62 are the arrays (mirror sets) that make up the multilevel stripe set.

  If you specify a label when creating the multilevel stripe set, it appears in this column. Because no label was specified when the multilevel stripe set was created, no label appears in the column.

  On UNIX systems, the root special file associated with the multilevel stripe set also appears in this column.

- The **Type** column displays the type Stripe for the newly created multilevel stripe set. The **Type** column also displays the array type for the underlying arrays. In the example, Mirror appears in the **Type** column to indicate that the arrays that make up the multilevel volume set are mirror sets.

- The **Total Size** column displays the total size for the multilevel stripe set. This size is the total of all the underlying arrays. In the example, the total size for array 0 (the multilevel stripe set) is the sum of the two underlying arrays (30 MB).

  The **Total Size** column also displays the sizes of all the underlying arrays. In the example, the size of array 63 (15.0 MB) and array 62 (15.0 MB) were specified when the mirror sets were created with the `container create mirror` command.
- The **Stripe Size** column displays the stripe size specified for the multilevel stripe set. In this example, the display shows 32 KB, the default stripe size.
- The **Usage** column displays None to indicate that the newly created multilevel stripe set does not have a file system on it. To create an NTFS or FAT file system on an array, use the `container format` command.
- The **State** column displays Normal for arrays 0, 63, and 62 to indicate that the state of the array is normal.

  Note that this column appears only if you specify the `/full` switch with the `container list` command.
- The **Scsi C:ID:L** column displays the SCSI device ID for the disk on which you created the multilevel stripe set. It also displays the SCSI device ID(s) for the disk(s) on which the array(s) that make up the multilevel stripe set reside. In the example, the display shows that array 63's partitions are on SCSI disk 0:02:0. array 62's partitions are on SCSI disk 0:03:0.
- The **Partition Offset: Size** column displays the partition offset and the size of the arrays that make up the newly created multilevel stripe set. In the example, the display shows that array 63 has partition offsets of 64.0 KB and 15.0 MB and sizes of 15.0 MB. array 62 has partition offsets of 64.0 KB and 15.0 MB and sizes of 15.0 MB. You specify the partition offset and size when you create the array.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container list` (page 3-50)
- `container set label` (page 3-101)

container create mvolume

To create a multilevel volume set, use the `container create mvolume` command. You can create the following types of multilevel volume sets:

- Volume set of stripe sets
- Volume set of mirror sets
- Volume set of RAID 5 arrays
- Volume set of volume sets

Notes

When creating a multilevel array of volume sets on a NetWare server, you specify the container IDs (as described in the Parameters section) for the arrays from which you want to create the multilevel volume set. If NetWare is using any of these arrays, an appropriate message displays. This message indicates that one or more of the arrays you specified is in use by NetWare. The message directs you to remove any NetWare volumes and partitions from these arrays. You can then create a multilevel array of volume sets on a NetWare server.

After you create a multilevel volume set you can run the `list devices` command on the NetWare console to verify its creation and then create the necessary NetWare disk partitions and volumes.

For information on how to create partitions and volumes, see the appropriate NetWare documentation.

Syntax

```
container create mvolume [/label{=string}]
{container} [{container}...]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array from which you want to create the multilevel volume set.

`{container}...`

Specifies the ID (0 to 63) of one or more additional arrays from which you want to create the multilevel volume set. You can use up to 16 arrays. Typically, the additional arrays you specify are the same type as the first array. For example, if the first array is a stripe set then any subsequent arrays are also stripe sets.

Switches

`/label{=string}`

Specifies a label to be assigned to the newly created multilevel volume set. You can specify a maximum of sixteen characters for the label.

If you do not specify the switch, it defaults to no label. If you do not specify a label, you can do so later by using the `container set label` command.

Examples

Before creating a multilevel volume set, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there are two existing arrays (stripe sets) on this controller at the time the multilevel volume set is created:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Stripe 45MB       64KB   None    0:02:0 64.0KB:  15.0MB
                                          0:03:0 64.0KB:  15.0MB
                                          0:04:0 64.0KB:  15.0MB
   1     Stripe 15MB       64KB   None    0:05:0 64.0KB:  15.0MB
```

The following example shows how to create a multilevel volume set from two single arrays:

```
AAC0> container create mvolume 0 1
Executing: container create mvolume 0 1
container 0 created
```

On UNIX systems, the message displayed after you execute the `container create mvolume` command includes the root special file associated with the newly created multilevel volume set.

Use the `container list` command after using the `container create mvolume` command to display information about the multilevel array of volume sets after you create it, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Volume 60.0MB             None
    63   Stripe 45MB       64KB           0:02:0 64.0KB:  15.0MB
                                          0:03:0 64.0KB:  15.0MB
                                          0:04:0 64.0KB:  15.0MB
    62   Stripe 15MB       64KB           0:05:0 64.0KB:  15.0MB
```

The following list describes the columns that contain information as a result of creating a multilevel volume set with the `container create mvolume` command:

- The **Num Label** column displays the ID number (in the example, 0) of the newly created multilevel volume set and (indented to the right) the ID numbers of the arrays that make up the multilevel volume set. In the example, arrays 63 and 62 are the stripe sets that make up the multilevel volume set.

  If you specify a label when creating the multilevel volume set, it appears in this column. Because no label was specified when the multilevel volume set was created, no label appears in the column.

  On UNIX systems, the root special file associated with the multilevel volume set also appears in this column.

- The **Type** column displays the type name **Volume** for the newly created multilevel volume set. The **Type** column also displays the array type for the underlying arrays. In the example, **Stripe** appears in the **Type** column to indicate that the arrays that make up the multilevel volume set are stripe sets.

- The **Total Size** column displays the total size for the multilevel volume set. This size is the total of all the underlying arrays. In the example, the total size for array 0 (the multilevel volume set) is the sum of the two underlying arrays (60 MB).

  The **Total Size** column also displays the sizes of all the underlying arrays. In the example, the size of array 63 (45 MB) and array 62 (15 MB) were specified when the stripe sets were created with the `container create stripe` command.

- The **Stripe Size** column displays the stripe size for each of the arrays that make up the multilevel volume set. You specify the stripe size when you create stripe sets and RAID 5 arrays. In the example, the stripe size is 32 KB.
- The **Usage** column displays **None** to indicate that the newly created multilevel volume set does not have a file system on it. To create an NTFS or FAT file system on an array, use the `container format` command.
- The **Scsi C:ID:L** column displays the SCSI device ID for the disk on which you created the multilevel volume set. It also displays the SCSI device ID(s) for the disk(s) on which the array(s) that make up the multilevel volume set reside. In the example, the display shows that array 63's partitions reside on three SCSI disks: 0:02:0, 0:03:0, and 0:04:0. array 62's partition resides on SCSI disk 0:05:0.
- The **Partition Offset: Size** column displays the partition offset and the size of the arrays that make up the newly created multilevel volume set. In the example, the display shows that array 63 has a partition offset of 64 KB and a size of 15 MB on each of the three SCSI disks. array 62 has a partition offset of 64 KB and a size of 15 MB. You specify the partition offset and size when you create the underlying arrays.

Related Commands

`container` commands:

- `container create mmirror` (page 3-8)
- `container create raid5` (page 3-21)
- `container create stripe` (page 3-27)
- `container set label` (page 3-101)

container create raid5

To create a RAID 5 array, use the `container create raid5` command. The RAID 5 array must have a minimum of three disks.

When creating the RAID 5 array, you must initialize the array parity by using the scrub method (the `/scrub` switch).

Notes

If you created a RAID 5 array you can run the `list devices` command on the NetWare console to verify its creation and then create the necessary NetWare disk partitions and volumes.

For information on how to create partitions and volumes, see the appropriate NetWare documentation.

Syntax

```
container create raid5 [/cache{=boolean}]
[/clear{=boolean}] [/stripe_size{=integer}]
[/label{=string}] [/scrub{=boolean}]
[/wait{=boolean}] {free_space} [{scsi_device}...]
```

Parameters

`{free_space}`

Specifies the SCSI device and its associated freespace used to create the RAID 5 array.

For further details, see *free_space* on page 1-11.

`{scsi_device}...`

Specifies one or more SCSI devices. A SCSI ID consists of a SCSI bus number (e.g., 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive). See the installation guide for your controller to determine the number of buses it actually supports.

For further details, see *scsi_device* on page 1-10.

The command uses the freespace(s) from the SCSI device(s) you specify to create the RAID 5 array. The size that the command uses from these device(s) is identical to the one you specify in the `free_space` parameter. The minimum number of partitions in a RAID 5 array is 3, and the maximum number of partitions is 16.

Switches

`/cache{=boolean}`

Specifies whether to enable the array's raw array cache. You can use this switch only if a native operating system's file system (for Windows, the NTFS or FAT file system) resides on the array. If you do not want to enable the RAID 5 array's raw array cache when you create it, you can do so later by using the `container set cache` command. In fact, the `container set cache` command gives you more control in setting the NVRAM write-back cache.

If you set this switch to `TRUE`, the command causes the controller to:

- Enable the read-ahead cache setting for the specified array.

You should always enable the read-ahead cache to optimize performance, unless your application–which is unlikely–is doing completely random reads.

- Enable when protected the NVRAM write-back cache setting for the specified array. This means the controller enables the array's NVRAM write-back cache only if a battery is present and its status is OK.

This switch defaults to `FALSE`, which means the command causes the controller to disable the array's raw array cache. If you accept the default, the command:

- Disables the read-ahead cache
- Disables the NVRAM write-back cache

`/clear{=boolean}`

Specifies whether to set up the parity during RAID-5 creation by clearing the entire RAID 5 array. If you specify `TRUE`, the command sets up the parity during RAID-5 creation by clearing the RAID 5 array.

The default is `FALSE`; the command does not clear the entire RAID 5 array.

If you use neither this switch nor the `/scrub` switch, the command uses the scrub method by default.

Unlike the scrub method where the drive is immediately available, the clear method does not make the drive available for use until the parity-initialize operation completes.

If you specify `TRUE` for both the `/scrub` and `/clear` switches, the command displays an appropriate error message and returns to the prompt.

`/stripe_size{=integer}`

Specifies the stripe size for the RAID 5 array. Valid values are 16 KB, 32 KB, and 64 KB.

If you do not specify a value for this switch, it defaults to 64 KB.

`/label{=string}`

Specifies a label to be assigned to the newly created RAID 5 array. You can specify a maximum of sixteen characters for the label.

If you do not specify the switch, it defaults to no label. If you do not specify a label, you can do so later by using the `container set label` command.

Note that this label is not the label that displays in Windows Explorer. The label displayed by Windows Explorer comes from the label specified with the `container format` command.

`/scrub{=boolean}`

Specifies whether to set up the parity during RAID-5 creation by scrubbing the RAID 5 array. If you set this switch to `TRUE`, the command sets up the parity by scrubbing the RAID 5 array. Although the drive is immediately available, it is not parity-protected until the background scrub action completes.

This switch defaults to `TRUE`.

If you set this switch to `FALSE`, the command will set up parity by clearing the RAID 5 array. In other words, setting `/scrub=FALSE` enables the same behavior as `/clear=TRUE`.

This switch is supported on Windows and NetWare.

For UNIX, the RAID 5 array is always scrubbed.

`/wait{=boolean}`

Specifies whether the command prompt returns only after the parity-protect operation completes. If you set this switch to `TRUE`, the command prompt returns only after the RAID 5 array is parity-protected. The scrub or zero action completes.

The default is `FALSE`; the command prompt returns immediately before the parity-protect operation completes.

Examples

The following example creates a RAID 5 array with the following characteristics:

- Creates the RAID 5 array on four disk drives connected to channel 0
- Specifies a freespace of 10 MB
- Uses the scrub method
- Indicates the command prompt return only after the scrub operation completes
- Specifies a stripe size of 64 KB
- Accepts the default label

```
AAC0>container create raid5 /stripe_size=64K /scrub ((0,02,0),10M)
(0,03,0) (0,04,0) (0,05,0)
Executing: container create raid5 /stripe_size=65,536  /scrub=TRUE
((CHANNEL=0,ID=2,LUN=0),10,485,760 ) (CHANNEL=0,ID=3,LUN=0)
(CHANNEL=0,ID=4,LUN=0) (CHANNEL=0,ID=5,LUN=0)
container 0 created
```

As the command executes, note that the title bar of the MS-DOS window displays the status of the command. For example:

```
Stat:OK!Task:101,Func:R5S Ctr:0,State:RUN 84.6%
```

For further details on status information, see page 1-13.

On UNIX systems, the message displayed after you execute the `container create raid5` command includes the root special file associated with the newly created RAID 5 array.

Use the `container list` command after using the `container create raid5` command to display information about the RAID 5 array.

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     RAID-5 30.0MB      64KB  None    0:02:0 64.0KB:  10.0MB
                                          0:03:0 64.0KB:  10.0MB
                                          0:04:0 64.0KB:  10.0MB
                                          0:04:0 64.0KB:  10.0MB
```

The following list describes the columns that contain information as a result of creating a RAID 5 array with the `container create raid5` command:

- The **Num Label** column displays the ID number of the newly created RAID 5 array, which in the example is 0.

  If you specify a label when creating the RAID 5 array, it appears in this column. Because no label was specified when the RAID 5 array was created, no label appears in the column.

  On UNIX systems, the root special file associated with the RAID 5 array also appears in this column.

- The **Type** column displays RAID-5 to indicate that the newly created array is a RAID 5 array.
- The **Total Size** column displays the total size of the freespace available. In this example, 40.0 MB of freespace was specified, which means that this column displays 30.0 MB as available.
- The **Stripe Size** column displays the stripe size you specified as the `stripe_size` parameter, which in the example is 64 KB.

- The **Usage** column displays **None** to indicate that the newly created RAID 5 array does not have a file system on it.To create an NTFS or FAT file system on an array, use the `container format` command.
- The **State** column displays the state of an array, which in the example shows **Unprot** to indicate that the RAID 5 array is not yet redundant. This means that the command has not completed the creation of the RAID 5 array. If the `container list` command is executed after the RAID 5 array completes execution, the State column is blank.

(Note that this column appears only if you specify the `/full` switch with the `container list` command.)

- The **Scsi C:ID:L** column displays the SCSI device ID(s) for the disk(s) on which you created the RAID 5 array, which in the example are: 0:02:0, 0:03:0, 0:04:0, and 0:05:0.
- The **Partition Offset: Size** column displays the partition offset(s) and size(s) of the newly created RAID 5 array, which in the example is 64.0 KB and 10.0 MB respectively.

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container set cache` (page 3-91)
- `container set label` (page 3-101)

container create stripe

To create a stripe set from freespace, use the `container create stripe` command.

Notes

If you created a stripe set you can run the `list devices` command on the NetWare console to verify its creation and then create the necessary NetWare disk partitions and volumes.

For information on how to create partitions and volumes, see the appropriate NetWare documentation.

Syntax

```
container create stripe [/cache{=boolean}]
[/stripe_size{=integer}] [/label{=string}]
{free_space} [{scsi_device}...]
```

Parameters

`{free_space}`

Specifies the SCSI device and its associated freespace used to create the stripe set.

For further details, see *free_space* on page 1-11.

`{scsi_device}...`

Specifies one or more SCSI devices. A SCSI ID consists of a SCSI bus number (e.g., 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive). See the installation guide for your controller to determine the number of buses it actually supports.

For further details, see *scsi_device* on page 1-10.

The command uses the freespace(s) from the SCSI device(s) to create the stripe set. The size from the device is identical to the one you specify in the `free_space` parameter.

The minimum number of partitions for a stripe set is 2, and the maximum number is 16.

Switches

`/cache{=boolean}`

Specifies whether to enable the array's raw array cache.You can use this switch only if a native operating system's file system (for Windows, the NTFS or FAT file system) resides on the array. If you do not want to enable the stripe set's raw array cache when you create it, you can do so later by using the `container set cache` command. In fact, the `container set cache` command gives you more control in setting the NVRAM write-back cache.

If you set this switch to `TRUE`, the command causes the controller to enable the read-ahead cache setting for the specified array.

You should always enable the read-ahead cache to optimize performance, unless your application–which is unlikely–is doing completely random reads.

- Enable when protected the NVRAM write-back cache setting for the specified array. This means the controller enables the array's NVRAM write-back cache only if a battery is present and its status is OK.

This switch defaults to `FALSE`, which means the command causes the controller to disable the array's raw array cache. If you accept the default, the command:

- Disables the read-ahead cache
- Disables the NVRAM write-back cache

`/stripe_size{=integer}`

Specifies the stripe size for the stripe set. Valid values are 16 KB, 32 KB, and 64 KB.

If you do not specify a value for this switch, it defaults to 64 KB.

`/label{=string}`

Specifies a label to be assigned to the newly created array. You can specify a maximum of sixteen characters for the label.

If you do not specify the switch, it defaults to no label. If you do not specify a label, you can do so later by using the `container set label` command.

Note that this label is not the label that displays in Windows Explorer. The label displayed by Windows Explorer comes from the label specified with the `container format` command.

Examples

Before creating a array, use the `disk show space` command to obtain information about the available SCSI devices, as in the following example:

```
AAC0>disk show space
Executing: disk show space
Scsi C:ID:L Usage     Size
----------- --------- -------------
  0:00:0   Free        64.0KB:  11.0MB
  0:01:0   Free        64.0KB:  11.0MB
  0:02:0   Free        64.0KB:  49.0MB
  0:03:0   Free        64.0KB:  49.0MB
  0:04:0   Free        64.0KB:  49.0MB
  0:05:0   Free        64.0KB:  49.0MB
  0:06:0   Free        64.0KB:  11.0MB
  0:08:0   Free        64.0KB:  11.0MB
```

The example list shows that there are eight SCSI disks available to create the array. There must be enough contiguous freespace on all the specified drives.

The following example shows how to create a array with the following characteristics:

- 15 MB of freespace on the (0,2,0) SCSI disk
- The default stripe size of 64 KB
- Two freespaces of 15 MB on the (0,3,0) and (0,4,0) SCSI disks

```
AAC0>container create stripe ((0,2,0), 15M) (0,3,0) (0,4,0)

Executing: container create stripe ((CHANNEL=0,ID=2,LUN=0),15,728,640 )
(CHANNEL=0,ID=3,LUN=0) (CHANNEL=0,ID=4,LUN=0)

container 0 created
```

On UNIX systems, the message displayed after you execute the `container create stripe` command includes the root special file associated with the newly created array.

Use the `container list` command after using the `container create stripe` command to display information about the array, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Stripe 45.0MB      64KB  None    0:02:0 64.0KB:  15.0MB
                                          0:03:0 64.0KB:  15.0MB
                                          0:04:0 64.0KB:  15.0MB
```

The following list describes the items in the display that contain information as the result of creating a array with the `container create stripe` command:

- The **Num Label** column displays the ID number of the newly created array, which in the example is 0.

  If you specify a label when creating the array, it appears in this column. Because no label was specified when the array was created, no label appears in the column.

  On UNIX systems, the root special file associated with the array also appears in this column.

- The **Type** column displays the type `Stripe`, to indicate that the newly created array is a array.

- The **Total Size** column displays the size of the freespace that makes up the array. In this example, the total size consists of the freespace (15.0 MB) specified with SCSI device (0,2,0) and equal sizes associated with SCSI devices (0,3,0), and (0,4,0) for a total of 45.0 MB.

- The **Stripe Size** column displays the stripe size you specified for the array. In this example, the stripe size is the default (32 KB).

- The **Usage** column displays **None** to indicate that the newly created array does not have a file system on it. To create an NTFS or FAT file system on an array, use the `container format` command.

- The **Scsi C:ID:L** column displays the SCSI device ID(s) for the disk(s) whose associated freespaces you used to create the array. In this example, the display shows 0:02:0, 0:03:0, and 0:04:0.

- The **Partition Offset: Size** column displays the partition offset(s) and the size(s) for the disk(s) whose associated freespaces you used to create the array. In this example, the display shows 64.0KB:15MB, 64.0KB:15.0MB, and 64.0KB:15.0MB.

Related Commands

`container` commands:

- `container create mstripe` (page 3-12)
- `container create mvolume` (page 3-17)
- `container list` (page 3-50)
- `container set cache` (page 3-91)
- `container set label` (page 3-101)

`disk` commands:

- `disk show space` (page 5-26)

container create volume

To create a volume set from freespace, use the `container create volume` command.

Notes

If you created a volume set you can run the `list devices` command on the NetWare console to verify its creation and then create the necessary NetWare disk partitions and volumes.

For information on how to create partitions and volumes, see the appropriate NetWare documentation.

Syntax

```
container create volume [/cache{=boolean}] [/
label{=string}] {scsi_device},{free_space}
[{scsi_device},{free_space}...]
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI device whose freespace you want to use for creating the volume set. A SCSI ID consists of a SCSI bus number (e.g., 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive). See the installation guide for your controller to determine the number of buses it actually supports.

For further details, see *scsi_device* on page 1-10.

Note that you must explicitly specify the entire ID for the SCSI device (the SCSI channel number, device ID, and device logical unit number).

`{free_space}`

Specifies the freespace used to create the volume set on the specified SCSI device.

For further details, see *free_space* on page 1-11.

`{scsi_device},{free_space}...`

Specifies any additional SCSI devices and associated freespaces used to create the volume set. A volume set can have a maximum of 16 partitions (or freespaces, since each freespace becomes a partition).

Note that you must explicitly specify the entire ID for any additional SCSI devices (the SCSI channel number, device ID, and device logical unit number).

Switches

`/cache{=boolean}`

Specifies whether to enable the array's raw array cache. You can use this switch only if a native operating system's file system (for Windows, the NTFS or FAT file system) resides on the array. If you do not want to enable the volume set's raw array cache when you create it, you can do so later by using the `container set cache` command. In fact, the `container set cache` command gives you more control in setting the NVRAM write-back cache.

If you set this switch to `TRUE`, the command causes the controller to enable the read-ahead cache setting for the specified array.

You should always enable the read-ahead cache to optimize performance, unless your application–which is unlikely–is doing completely random reads.

- Enable when protected the NVRAM write-back cache setting for the specified array. This means the controller enables the array's NVRAM write-back cache only if a battery is present and its status is OK.

This switch defaults to `FALSE`, which means the command causes the controller to disable the array's raw array cache. If you accept the default, the command:

- Disables the read-ahead cache
- Disables the NVRAM write-back cache

`/label{=string}`

Specifies a label to be assigned to the newly created volume set. You can specify a maximum of sixteen characters for the label.

If you do not specify the switch, it defaults to no label. If you do not specify a label, you can do so later by using the `container set label` command.

Note that this label is not the label that displays in Windows Explorer. The label displayed by Windows Explorer comes from the label specified with the `container format` command.

Examples

Before creating a volume set, use the `disk show space` command to obtain information about the available SCSI devices, as in the following example:

```
AAC0>disk show space
Executing: disk show space
Scsi C:ID:L Usage     Size
----------- --------- -------------
  0:00:0   Free        64.0KB:  11.0MB
  0:01:0   Free        64.0KB:  11.0MB
  0:02:0   Free        64.0KB:  49.0MB
  0:03:0   Free        64.0KB:  49.0MB
  0:04:0   Free        64.0KB:  49.0MB
  0:05:0   Free        64.0KB:  49.0MB
  0:06:0   Free        64.0KB:  11.0MB
  0:08:0   Free        64.0KB:  11.0MB
```

The example list shows that there are eight SCSI disks available to create the volume set.

The following example creates a volume set on SCSI disk `(0,2,0)` with a freespace of 15 MB:

```
AAC0>container create volume ((0,2,0), 15M)
Executing: container create volume ((CHANNEL=0,ID=2,LUN=0),15,728,640)
container 0 created
```

On UNIX systems, the message displayed after you execute the `container create volume` command includes the root special file associated with the newly created volume set.

Use the `container list` command after executing the `container create volume` command to display information about the volume set, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Volume 15.0MB             None    0:02:0 64.0KB:  15.0MB
```

The following list describes the columns that contain information as the result of creating a volume set with the `container create volume` command:

- The **Num Label** column displays the ID number of the newly created volume set, which in the example is 0.

  If you specify a label when creating the volume set, it appears in this column. Because no label was specified when the volume set was created, no label appears in the column.

  On UNIX systems, the root special file associated with the array also appears in this column.
- The **Type** column displays Volume to indicate that the newly created array is a volume set.
- The **Total Size** column displays the value(s) specified in the `free_space` parameter(s), which in the example is 15.0 MB.
- The **Usage** column displays **None** to indicate that the newly created volume set does not have a file system on it. To create an NTFS or FAT file system on an array, use the `container format` command.
- The **Scsi C:ID:L** column displays the SCSI device ID for the disk on which you created the volume set, which in the example is 0:02:0.
- The **Partition Offset: Size** column displays the partition offset and the size of the newly created volume set's partition, which in the example are 64.0 KB and 15.0 MB. The size is the size you specified in the `free_space` parameter(s).

Related Commands

`container` commands:

- `container extend volume` (page 3-47)
- `container format` (page 3-50)

- `container list` (page 3-50)
- `container promote` (page 3-63)
- `container set cache` (page 3-91)
- `container set label` (page 3-101)

`disk` commands:

- `disk show space` (page 5-26)

container delete

To delete an array from the currently opened controller, use the `container delete` command.

Notes

If you are deleting an array on a NetWare system, you must first go to the NetWare console and remove the corresponding volume and partition. For information on how to perform these operations, see the appropriate NetWare documentation.

After you delete an array on a NetWare server, you must run the list devices command on the NetWare console so that the corresponding virtual disk is removed from the system's internal device table.

Syntax

```
container delete [/always{=boolean}]
[/unconditional{=boolean}]{container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to delete. The array can be any array on the controller.

You can never delete an array if files are open on it.

for the syntax associated with specifying a partner and a container ID.

Switches

`/always{=boolean}`

Specifies whether to delete the array, even if it has a file system. If you specify `TRUE`, the command deletes the array even if it has a file system. If you specify `FALSE`, the command deletes the array only if it has no file system.

This switch defaults to `FALSE`. In both cases, all user files must be closed; the `/always` switch cannot override this restriction.

`/unconditional{=boolean}`

Specifies whether to delete the array, even if the array has open files on it. If you specify `TRUE`, the command deletes the array even if it has open files on it.

The switch defaults to `FALSE` (that is, the `container delete` command does not delete an array that has open files on it).

WARNING: Unconditionally deleting an array that is in use can cause a system crash under some circumstances.

Examples

Before deleting an array, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a multilevel volume set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 40.0MB             NTFS
    63   Volume 10.0MB                    0:02:0 64.0KB:  10.0MB
    62   Volume 15.0MB                    0:02:0 64.0KB:  10.0MB
    61   Volume 15.0MB                    0:04:0 64.0KB:  10.0MB
```

The following example shows how to delete array 0 using the `/always` switch to indicate a delete even though the array has a file system:

```
AAC0>container delete /always=TRUE 0
Executing: container delete /always=TRUE 0
```

Use the `container list` command after using the `container delete` command to confirm that the command actually deleted the specified array, as in the following example:

```
AAC0>container list

Executing: container list

No containers found.
```

Note that in the example, the deletion of array 0 caused the deletion of all the underlying (hidden) arrays (arrays 63, 62, and 61).

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container create mmirror` (page 3-8)
- `container create mstripe` (page 3-12)
- `container create mvolume` (page 3-17)
- `container create raid5` (page 3-21)
- `container create stripe` (page 3-27)
- `container create volume` (page 3-32)
- `container list` (page 3-50)

container extend file_system

To extend a file system so that it uses all of the space in an array, use the `container extend file_system` command. This command allows you to extend the NTFS file system.

Typically, you use this command after extending an array (by adding a level to it with the `container add_level` command and, possibly, by extending it with the `container extend mvolume` command).

If you extend an NTFS file system, you must reboot your system in order for the extension to take effect.

Notes

The following notes relate to using the `container extend file_system` command to extend an NTFS file system:

- If you extend an NTFS file system, the command displays an appropriate message indicating that you need to reboot the system to show the new space. If you check the Windows Event Log, a message similar to the following appears:

  ```
  The file system structure on the disk is corrupt
  and unusable. Please run the chkdsk utility on the
  device \Device\Harddisk0\Partition1 with label "".
  ```

  The previous message always appears in the Windows Event Log even if the file system extend operation is successful. You do not need to run the chkdsk utility after a successful file system extend operation.
- You cannot extend an NTFS file system that resides on a boot array.

Command and Switch Availability

This command is supported on Windows.

Syntax

```
container extend file_system {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array whose file system you want to extend.

Examples

Typically, you would use the `container extend file_system` command after adding a level to an array and, perhaps, after extending a multilevel volume set.

Before extending a file system, use the `container list` command to obtain information about any existing arrays:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
E: 0     Volume 300MB             NTFS
    63   Volume 100MB                     1:00:0 64.0KB:   100MB
    62   Volume 100MB                     1:01:0 64.0KB:   100MB
    61   Volume 100MB                     1:02:0 64.0KB:   100MB
```

The following list describes the sequence that caused the previous example to display:

1. Create a volume set of 100 MB on disk (1,0,0) with the `container create volume` command.
2. Add a level to the volume set created in Step 1 with the `container add_level` command.
3. Assign a drive letter to the volume set created in Step 1 with the `container assign drive_letter` command.
4. Create an NTFS file system on the volume set created in Step 1 with the `container format` command.
5. Create a second volume set of 100 MB on disk (1,1,0) with the `container create volume` command.
6. Create a third volume set of 100 MB on disk (1,2,0) with the `container create volume` command.
7. Use the `container extend mvolume` command to add the arrays created in Steps 5 and 6 to the array created in Step 1.

The following example extends the NTFS file system to make use of the additional arrays:

```
AAC0>container extend file_system 0
Executing: container extend file_system 0
The system has been marked to expand the file system on the next
reboot.
Reboot the system to get the file system to expand to show the new
space.
```

Use the `container list` command after using the `container extend file_system` command to display information about the array after you extend a file system. Note that there is no change in the display as a result of using the `container extend file_system` command.

Related Commands

`container` commands:

- `container format` (page 3-50)

container extend mvolume

To extend a multilevel volume set by adding one or more arrays to it, use the `container extend mvolume` command. Any file system on the multilevel volume set remains intact, and can be extended to include the added space.

Command and Switch Availability

This command is supported on Windows.

Syntax

```
container extend mvolume  {container_to_extend}
{container} [{container}...]
```

Parameters

`{container_to_extend}`

Specifies the ID number (0 to 63) of the array to extend. Typically, this array is a multilevel volume set.

Specifies the ID number (0 to 63) of the array to add to the previously specified multilevel volume set. If a file system exists on this array, the command displays an appropriate error message and does not allow you to create the multilevel volume set. This prevents the loss of any data residing in files on the array.

`{container}...`

Specifies the ID number (0 to 63) of the array or arrays to add to the previously specified multilevel volume set. The previous file system information applies to these additional arrays as well.

Examples

Before extending a multilevel volume set, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a multilevel volume set) on this controller. In addition, the example shows array 1 and array 2 (both volume sets created with the `container create volume` command).

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ -------  ------ -------------
F: 0     Volume 10MB              NTFS
    63   Volume 10MB                       0:02:0 64.0KB:  10.0MB
   1     Volume 15MB              None     0:03:0 64.0KB:  15.0MB
   2     Volume 15MB              None     0:04:0 64.0KB:  15.0MB
```

The following example extends a multilevel volume set (array 0) by adding arrays 1 and 2 (which are both volume sets):

```
AAC0>container extend mvolume 0 1 2
Executing: container extend mvolume 0 1 2
container 0 extended
```

As the command executes, note that the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!Task:102,Func:FSX Ctr:0,State:RUN 84.3%
```

For further details on status information, see page 1-13.

Use the `container list` command after using the `container extend mvolume` command to display information about the multilevel array of volume sets after you extend it, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ -------  ------ -------------
F:  0     Volume 40MB              NTFS
    63   Volume 10MB                       0:02:0 64.0KB:  10.0MB
    62   Volume 15MB                       0:03:0 64.0KB:  15.0MB
    61   Volume 15MB                       0:04:0 64.0KB:  15.0MB
```

The following list describes the change to the display as the result of extending a multilevel array with the `container extend mvolume` command:

- The **Num Label** column continues to display the ID number (0) of the volume set previously created with the `container create volume` command and the ID number (63) of the volume set previously created with the `container add_level` command. In addition, the column displays the ID numbers (in this example, 62 and 61) of the volume sets created as a result of the `container extend mvolume` command.

  If you specified labels when creating the volume sets, they appear in this column. Because no labels were specified when the volume sets were created, no labels appear in the column.
- The **Type** column continues to display Volume to indicate that the arrays previously created with the `container create volume` and the `container add_level` commands are volume sets. In addition, the column displays Volume for the volume sets created as a result of the `container extend mvolume` command.
- The **Total Size** column displays a new total size, taking into account the freespaces associated with the volume sets created as a result of the `container create volume` and the `container add_level` commands.
- The **Scsi C:ID:L** column displays the SCSI IDs for the devices on which the volume sets were created, which in the example are 0:02:0, 0:03:0, and 0:04:0.
- The **Partition Offset:Size** column displays the offsets and sizes for the partition(s) on which the volume sets were created, which in the example are 64.0KB:10MB, 64.0KB:15.0MB, and 64KB:15MB.

  If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

Related Commands

`container` commands:

- `container add_level` (page 3-3)
- `container create mvolume` (page 3-17)
- `container create volume` (page 3-32)
- `container extend file_system` (page 3-40)
- `container list` (page 3-50)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)

container extend volume

To extend a volume set by adding freespace to it, use the `container extend volume` command. The file system on the volume set remains intact and can be extended to include the added space.

Command and Switch Availability

This command is supported on Windows.

Syntax

```
container extend volume  {container}
[{free_space}] [{free_space}...]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array (volume set) to extend.

`{free_space}`

Specifies the SCSI device and its associated freespace used to extend the specified array (volume set).

For further details, see *free_space* on page 1-11.

`{free_space}...`

Specifies a SCSI device or devices and associated freespace or freespaces used to extend the specified array (volume set). You can specify a maximum of 16 freespace elements. If you exceed 16 freespace elements, the command displays an appropriate error message.

Examples

Before extending a volume set, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array (volume set) 1 on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
E: 1     Volume 100MB            NTFS    2:02:0 200MB: 100MB
```

Before extending the volume set, you might want to use the `disk show space` command to display space usage information on the SCSI devices from which you plan to add freespace. The following example extends a volume set by adding freespace from two SCSI devices:

```
AAC0>container extend volume 1 ((2,0,0), 32M) ((2,1,0), 32M)
Executing: container extend volume 1
((CHANNEL=2,ID=0,LUN=0),33,554,432)((CHANNEL=2,ID=1,LUN=0),33,554,432)
container 1 expanded.
```

As the command executes, note that the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!,Task:101,Func:FSX Ctr:0,State:RUN 83.8%
```

For further details on status information, see page 1-13.

Use the `container list` command after using the `container extend volume` command to display information about the volume set after you extend it, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
E: 1     Volume 164MB             NTFS    2:02:0 200MB: 100MB
                                         2:00:0 64.0KB:  32.0MB
                                         2:01:0 64.0KB:  32.0MB
```

The following list describes the change to the display as the result of extending a volume set with the `container extend volume` command:

- The **Total Size** column displays a new size, taking into account the value(s) specified for the `free_space` parameters. In the example the new size is 164 MB.
- The **Usage** column continues to display NTFS, indicating that array 1 has an NTFS file system on it.
- The **Scsi C:ID:L** column displays the SCSI ID for the disk on which the original volume set was created. It also displays the SCSI ID(s) for the disk(s) from which you specified the freespace. In the example, these disks are 2:00:0 and 2:01:0.

- The **Partition Offset:Size** column displays the partition offset(s) and size(s) for the partition(s) associated with the extended volume set. In the example the partition offsets and sizes are 64.0 KB and 32.0 MB.

  If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

Related Commands

`container` commands:

- `container create volume` (page 3-32)
- `container list` (page 3-50)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)
- `disk show space` (page 5-26)

container list

To display information about one or all arrays on the controller, use the `container list` command. The display contains such information as the container's ID number and other useful information. Typically, you use the `container list` command to obtain specific information about arrays prior to using other array-related commands.

Syntax

```
container list [/all{=boolean}] [/full{=boolean}]
[{container}]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array whose information you want to display. To display information about all of the arrays on the system, omit the ID number from the command.

Switches

`/all{=boolean}`

Specifies whether to list all arrays on the system. If you specify `TRUE`, the command displays all arrays on the system. If you specify `FALSE`, the command displays only the array you specify in the command.

This switch defaults to `FALSE` if you specify a `container`. Otherwise, the switch defaults to `TRUE`.

`/full{=boolean}`

Specifies whether to display detailed information. If you specify `TRUE`, the command displays detailed information. If you specify `FALSE`, the command does not display detailed information.

This switch defaults to `FALSE`.

Examples

The following example shows how to list nondetailed information about all arrays on the system:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Volume 15.0MB             None    0:02:0 64.0KB:  15.0MB
   1     Stripe 45.0MB     32KB    None    0:02:0 15.0MB:  15.0MB
                                             0:03:0 64.0KB:  15.0MB
                                             0:04:0 64.0KB:  15.0MB
```

The detailed `container list` display contains the previous columns of information plus the following columns:

- State
- RO
- Lk
- Task
- Done%
- Ent
- Creation Date
- Creation Time
- Boot Device
- System Files

The following sections provide brief descriptions of each `container list` column.

The Dr Column

In Windows, this column displays a blank space if you did not assign a drive letter to the array. Otherwise, the column contains the letter associated with the array. Use the `container assign drive_letter` command to assign a drive letter to an array.

The **Dr** column does not appear in UNIX or NetWare systems.

The Num Label Column

This column displays the container ID (a number from 0 to 63 inclusive). Typically, the CLI `container create`-related commands automatically assign an ID to an array. However, you can renumber an array with the `container move` command.

This column also displays the label assigned to the array when the array was created. If no label was assigned to the array then no label appears in the column. You can assign labels with the following commands:

- `container create mstripe`
- `container create mvolume`
- `container create raid5`
- `container create stripe`
- `container create volume`
- `container set label`

If your controller is running on a UNIX operating system, then the **Num Label** column also displays a root special file. On UNIX systems, arrays are associated with root special files not drive letters.

See your UNIX documentation for information on how to mount the array or to create a file system.

The Type Column

This column displays the type of array. Table 3-1 describes the type values that can display in the Type column.

Table 3-1 Container Types

container Type	Meaning
`Mirror`	The container is a mirror set.
`Stripe`	The container is a stripe set.
`Volume`	The container is a volume set.
`RAID-5`	The container is a RAID 5 array.
`Reconf`	The container was reconfigured.

The CLI automatically assigns the array type as a result of creating arrays with the `container create`-related commands.

The CLI assigns the array type `Reconf` when the array is reconfigured from one array type to another. The `container reconfigure` command provides switches that allow you to reconfigure an array into specific array types.

The Reconf array type does not appear on UNIX systems.

The Total Size Column

This column displays the size of the array. You specify this size when creating the array with one of the `container create`-related commands.

The Stripe Size Column

This column displays the stripe size for the array. You specify the stripe size when using the `container create mstripe`, `container create raid5`, and `container create stripe` commands.

The Usage Column

This column displays information about the data on or the status of the array. Specifically, the column can display the items listed in Table 3-2.

Table 3-2 Container Usage Items

Item	Meaning
`FAT`	The FAT file system resides on this array.
`MultPart`	A file system resides on a multi-partition array. A multi-partition array is an array that has multiple operating system or DOS partitions.
`NetWare`	The array resides in a NetWare environment.
`None`	No file system resides on this array.
`NTFS`	The NTFS file system resides on this array.
`Open`	For UNIX, there is a mounted file system on this array.
`UnCfged`	The array is a phantom array or the array is offline.
`Unknown`	An unknown file system resides on this array. For UNIX, the operating system recognized this array, and there has not been a query (e.g., mount, fdsk, read, write) on the array.
`UnMap'd`	The array is unusable and cannot be mounted.
`Valid`	The UNIX operating system recognized this array, and there has been a query (e.g., mount, fdsk, read, write) on the array.

The Scsi C:ID:L Column

This column displays the SCSI channel number, the SCSI ID, and the SCSI logical unit number for the SCSI disk or disks on which the array was created.

The Partition Offset:Size Column

This column displays the offset and size for the underlying partitions.

If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

The State Column

This column displays information about the state of the array. Specifically, the column can display the items listed in Table 3-3.

Table 3-3 Container State Items

Container State	Meaning
`Copy`	Indicates that the array is the copy array in an array reconfiguration operation.
`Create`	Indicates the creation of a mirror set.
`Dest`	Indicates that the array is the destination array in an array reconfiguration operation.
`Normal`	Indicates that the mirror set is in the normal state.
`Raid5`	Indicates that the array is a RAID 5 array in an array reconfiguration operation.
`Source`	Indicates that the array is the source array in an array reconfiguration operation.
`Temp`	Indicates that the array is a temporary array in an array reconfiguration operation.
`Unprot`	The RAID 5 array is not redundant. The `Unprot` array state typically displays during the creation of the RAID 5 array. Upon completion of RAID 5 array creation, the CLI replaces the `Unprot` state with the `Normal` state.

The RO Column

This column displays the letters `RO` if the array is read-only. Otherwise, if the array is read-write the column displays a blank space. You can explicitly make an array read-only and read-write with the `container readonly` and `container readwrite` commands.

Some CLI commands cause an array to temporarily become read-only without the use of the `container readonly` command.

The Lk Column

This column displays the letter `L` if the array is locked. Otherwise, if the array is not locked, the column displays a blank space. You can explicitly lock and unlock an array with the `container lock` and `container unlock` commands.

Some CLI commands (during their execution) cause the array to become locked. Typically these commands unlock the array upon completing execution.

The Task Column

This column displays the task or tasks running on a specified array or SCSI disk. Specifically, the column can display the items listed in Table 3-4.

Table 3-4 Task Types

Task	Meaning
`Create`	A create mirror set or create multilevel mirror set task is running on the specified array. When the create mirror set or create multilevel mirror set task completes, the specified array is a mirror set or a multilevel array of mirror sets. The create a mirror set or create a multilevel mirror set task runs as a result of using the `container create mirror` or `container create mmirror` command.
`FmtFAT`	An array format FAT file system task is running on the specified array. The format FAT file system task runs as a result of using the `container format` command with the `/file_system` switch set to `FAT`.
`FmtNTFS`	An array format NTFS file system task is running on the specified array. The format NTFS file system task runs as a result of using the `container format` command with the `/file_system` switch set to `NTFS`.

Table 3-4 Task Types (Continued)

Task	Meaning
`Rebuild`	A rebuild task is running on the specified array. Typically, the rebuild task runs when the controller is in a rebuild for a redundant array (RAID 5 array, mirror set, or multilevel array of mirror sets).
`Reconfg`	An array reconfigure task is running on the specified array. When the array reconfigure task completes, the specified array becomes a different array (for example, from a volume set to a stripe set). The array reconfigure task runs as a result of using the `container reconfigure` command.
`Scrub`	A scrub task is running on the specified redundant array. When the scrub task completes, the specified redundant array has reconstructed data on one partition based on data found on the other partition (for mirror sets and multilevel arrays of mirror sets). Or, the specified redundant array recalculates and replaces, if necessary, the parity information (for RAID 5 arrays). The scrub task runs as a result of the `container scrub` command. The scrub task also runs as a result of the `container create raid5` command with the `/scrub` switch specified.
`Verify`	A verify with no repair of bad blocks task is running on the specified SCSI disk. When the verify with no repair of bad blocks task completes, the specified SCSI disk's blocks were verified without repairing any detected defects. The verify with no repair of bad blocks task runs as a result of using the `disk verify` command without specifying the `/repair` switch.
`VfyRepl`	A verify with repair of bad blocks task is running on the specified SCSI disk. When the verify with repair of bad blocks task completes, the specified SCSI disk's blocks are verified with repairs. The verify with repair of bad blocks task runs as a result of using the `disk verify` command with the `/repair` switch.
`Zero`	A clear disk task is running on the specified SCSI disk. When the clear disk task completes, the specified SCSI disk is cleared (that is, all data is erased and cannot be recovered). The clear disk task runs as a result of using the `disk zero` command.

The Done % Column

This column displays a running percentage value that indicates the progress of the currently running task. The display shows the percentage in tenths of a percent increments. The currently running task is complete when the Done % column displays 100%.

The Ent Column

This column displays the number of elements associated with each array. The maximum is 16 elements.

The Creation Date and Creation Time Columns

The Creation Date column displays the date on which you created the array. The following example shows an example of a date that can appear in this column:

```
082999
```

As the example shows, the command displays the date in the form `mmddyy` where

- `mm` is a two-digit number that indicates the month in which you created the array. For example, 01 indicates the month of January, 12 indicates the month of December, and so forth.
- `dd` is a two-digit number that indicates the day of the month in which you created the array. For example, 29 indicates the 29th day.
- `yy` indicates the year you created the array. For example, 97 indicates the array was created in the year 1997.

The **Creation Time** column displays the time in 24-hour format at which you created the array. The following example shows an example of a time that can appear in this column:

```
11:55:49
```

As the example shows, the command displays the time in the form `hhmmss` where

- `hh` is a two-digit number that indicates the hour at which you created the array.
- `mm` is a two-digit number that indicates the minute at which you created the array.
- `ss` is a two-digit number that indicates the second at which you created the array.

The Boot Device Column

This column displays a value that indicates whether the array resides on the boot device. An X appears in the column if the array resides on the boot device. Otherwise, if the array does not reside on the boot device, no value appears in the column.

The **Boot Device** column does not appear on UNIX systems.

The System Files Column

This column displays a value that indicates whether the array resides on the system device. An X appears in the column if the array resides on the system device. Otherwise, if the array does not reside on the system device, no value appears in the column.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container create mstripe` (page 3-12)
- `container create raid5` (page 3-21)
- `container create stripe` (page 3-27)
- `container scrub` (page 3-88)

`disk` commands:

- `disk verify` (page 5-29)
- `disk zero` (page 5-31)

container lock

Caution: Use the `container lock` command only under the direction of technical support.

To lock an array into volatile memory space on the currently opened controller, use the `container lock` command. When an array is locked into volatile memory space, the `container reconfigure` command has no effect. A locked array cannot be moved, deleted, made read-only, or used to create a multilevel array.

Syntax

```
container lock {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to lock into volatile memory space.

Examples

Before locking an array, use the `container list` command (with the `/full` switch) to obtain information about any existing arrays. As the following example shows, there is an array 0 (a volume set) on this controller. Note that the **Lk** column is blank:

```
AAC0>container list /full
Executing: container list /full=TRUE
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    Lk
-- ----- ------ ------ --- ------ ------- --------  --
F: 0     Volume 10.0MB            NTFS    0:02:0
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **Lk** column.

The following example locks array 0 into volatile memory space:

```
AAC0>container lock 0
Executing: container lock 0
```

Use the `container list` command (with the `/full` switch) after using the `container lock` command to display information about the array you just locked, as in the following example:

```
AAC0>container list /full
Executing: container list /full=TRUE
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    Lk
-- ----- ------ ------ --- ------ ------- --------  --
F: 0     Volume 10.0MB            NTFS    0:02:0    L
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **Lk** column.

As the result of locking an array (in this example, array 0) with the `container lock` command, the **Lk** column displays L (instead of a blank) to indicate that the specified array is now locked.

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container unlock` (page 3-112)

container move

To renumber an array, use the `container move` command. If another array already has the new number, the command returns an error.

Notes

A controller currently supports 24 visible arrays. A visible array is an array that is visible to the operating system and users. Visible arrays are identified with container IDs 0 through 23.

Container IDs 24 through 63 are reserved for hidden arrays. A hidden array is an array that is not visible to the operating system and can only be used by other arrays. (The `container list` command, however, displays hidden arrays.)

The `container move` command does not prevent you from assigning a hidden container ID (24 through 63) to a visible array. If you assign a visible array with a hidden container ID (24 through 63), the operating system no longer sees the visible array. In general, when renumbering a visible array, use container IDs 0 through 23. When renumbering a hidden array, use container IDs 24 through 63.

Syntax

```
container move {container} {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to renumber.

`{container}`

Specifies the ID number (0 to 63) to assign to the array specified in the first `container` parameter.

Examples

Before renumbering an array, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a volume set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 20.0MB            NTFS    0:02:0 64.0KB:  10.0MB
                                         0:03:0 64.0KB:  10.0MB
```

The following example renumbers array 0 to array 5:

```
AAC0>container move 0 5
Executing: container move 0 5
```

Use the `container list` command after using the `container move` command to display information about the array you just renumbered, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 5     Volume 20.0MB            NTFS    0:02:0 64.0KB:  10.0MB
                                         0:03:0 64.0KB:  10.0MB
```

As a result of renumbering array 0 to array 5 with the `container move` command, the **Num Label** column displays ID number 5 instead of 0. This column also displays the label assigned to the array when the array was created. If no label was assigned to the array then no label appears in the column.

Related Commands

`container` commands:

- `container list` (page 3-50)

container promote

To create a multilevel volume set from a stripe set or volume set, use the `container promote` command. In this case, the resulting array is an array of single-partition volume sets, each of which has a single partition from the original array.

Typically, you use the `container promote` command as part of a process to provide fault tolerance for existing volume sets. After `container promote` is executed, use `container create mmirror` (which provides the fault tolerance) to make this array an array of mirror sets.

Notes

The `container promote` command differs from the `container add_level` command as follows:

- The `container create volume` command creates a single-level array.
- If you use the `container add_level` command and specify the ID for a single-level volume set, the result is a volume set with two partitions.
- If you use the `container promote` command and specify the ID for a single-level volume set, the result is a multilevel volume set (a volume set with two volumes under it).

Syntax

```
container promote {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to promote to a multilevel array.

Examples

Before promoting an array to a multilevel array, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there is an array 0 (a stripe set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Stripe 45.0MB     32KB   None    0:02:0 64.0KB:  15.0MB
                                          0:03:0 64.0KB:  15.0MB
                                          0:04:0 64.0KB:  15.0MB
```

The following example promotes array 0 to a multilevel array:

```
AAC0>container promote 0
Executing: container promote 0
```

Use the `container list` command after using the `container promote` command to display information about the array you just promoted, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Stripe 45.0MB     32KB   None
    63   Volume 15.0MB                    0:02:0 64.0KB:  15.0MB
    62   Volume 15.0MB                    0:03:0 64.0KB:  15.0MB
    61   Volume 15.0MB                    0:04:0 64.0KB:  15.0MB
```

The following list describes the change to the display as the result of promoting an array (in this example, array 0) with the `container promote` command:

- The **Num Label** column displays four container IDs: 0 (the array that was promoted to a multilevel volume set), 63 (one of the underlying volume sets), 62 (a second underlying volume set), and 61 (a third underlying volume set).

  This column also displays the label assigned to the array when the array was created. If no label was assigned to the array then no label appears in the column.

- The **Type** column displays the Volume array type for the three underlying volume sets.

- The **Total Size** column displays the sizes of the arrays. Note that the underlying arrays are each 15.0 MB and array 0's size is the total of the two underlying arrays.

- The **Scsi C:ID:L** column displays the SCSI channel number, SCSI device ID, and SCSI logical unit number for the underlying arrays. In the example, array 63 resides on disk 0:02:0, array 62 resides on disk 0:03:0, and array 61 resides on disk 0:04:0.
- The **Partition Offset:Size** column displays the offset and size for the underlying arrays' partitions. In the example, arrays 63, 62, and 61 have 64.0 KB offsets and 15.0 MB sizes.

  If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

As stated previously, you can now use the `container create mmirror` command to make a promoted array an array of mirror sets.

Related Commands

`container` commands:

- `container create mmirror` (page 3-8)
- `container create volume` (page 3-32)
- `container list` (page 3-50)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)

container readonly

To change an array's read-write status to read-only status, use the `container readonly` command. Subsequent modifications to the data on the array are prohibited. To use the `container readonly` command, the array cannot be in use by any application.

Syntax

```
container readonly {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to make read-only.

Examples

Before making an array read-only, use the `container list` command (with the `/full` switch) to obtain information about any existing arrays. As the following example shows, there is an array 0 (a volume set) on this controller. Note that the **RO** column is blank:

```
AAC0>container list /full
Executing: container list /full=TRUE 0
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    RO
-- ----- ------ ------ --- ------ ------- --------  ----
F: 0     Volume 10.0MB             NTFS    0:02:0
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **RO** column.

The following example changes array 0 from read-write status to read-only status:

```
AAC0>container readonly 0
Executing: container readonly 0
```

Use the `container list` command (with the `/full` switch) after using the `container readonly` command to display information about the array you just made read-only, as in the following example:

```
AAC0>container list /full
Executing: container list /full=TRUE
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    RO
-- ----- ------ ------ --- ------ ------- --------  ----
F: 0     Volume 10.0MB              NTFS   0:02:0    RO
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **RO** column.

As a result of making an array (in this example, array 0) read-only with the `container readonly` command, the **RO** column displays RO (instead of a blank) to indicate that the specified array is now read-only.

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container readwrite` (page 3-68)

container readwrite

To change an array's read-only status to read-write status, use the `container readwrite` command. Subsequent modifications to the data on the array are allowed. To use the `container readwrite` command, none of the array's files can be open.

Syntax

```
container readwrite {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to make read-write.

Examples

Before making an array read-write, use the `container list` command (with the `/full` switch) to obtain information about any existing arrays. As the following example shows, there is an array 0 (a volume set) on this controller. Note that the **RO** column displays the value RO:

```
AAC0>container list /full
Executing: container list /full=TRUE
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    RO
-- ----- ------ ------ --- ------ ------- --------  ----
F: 0     Volume 10.0MB            NTFS    0:02:0    RO
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **RO** column.

The following example changes array 0 from read-only status to read-write status:

```
AAC0>container readwrite 0
Executing: container readwrite 0
```

Use the `container list` command (with the `/full` switch) after using the `container readwrite` command to display information about the array you just made read-write, as in the following example:

```
AAC0>container list /full
Executing: container list /full=TRUE
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    RO
-- ----- ------ ------ --- ------ ------- --------  ----
F: 0     Volume 10.0MB            NTFS    0:02:0
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **RO** column.

As a result of making an array (in this example, array 0) read-write with the `container readwrite` command, the **RO** column displays blank (instead of an RO) to indicate that the specified array is now read-write.

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container readonly` (page 3-66)

container reconfigure

To change the configuration of an array, use the `container reconfigure` command. The `container reconfigure` command allows you to perform the following operations:

- Transform an array from one type to another type

  You can transform an array from one type to another type. For example, you can transform a RAID 1 to a RAID 5 array. You specify the appropriate switches that the `container reconfigure` command provides to transform an array.

- Add more SCSI drives to an array

  You can add more SCSI drives to an array by specifying one or more SCSI devices. This action extends the array. The command uses the freespace(s) from the SCSI device(s) you specify to reconfigure the array.

  For example, you can extend a RAID array by specifying one or more devices. In this case, the array grows, but remains a RAID 1.

- Change an array's stripe size

  You can change an array's stripe size by specifying the appropriate switch with a valid stripe size. For example, you can change the stripe size of a stripe set from 16K to 32K.

Note: The stripe size for a RAID 50 array is always 64 KB.

- Change an array's partition size

  You can change an array's partition size by specifying the appropriate switch with a valid partition size. For example, you can change the partition size of a mirror set from 500 MB to 800 MB.

- Extend an existing file system

  You can extend an existing file system by specifying the appropriate switch. (The command extends the file system after completing the reconfigure operation.)

- Move partitions to other disks

  You can move an array's partitions to other disks by specifying the appropriate switches with a valid partition or partitions. For example, you can move three entries of a RAID 5 array to new disks.

You can combine the previously listed operations.

Notes

Some reconfigure operations result in destination sizes slightly different than what you specified.

When you perform an array reconfigure operation on a UNIX operating system you may see a resource conflict-related error. Typically, this error occurs when there is not enough space on the disk to perform the reconfigure operation. To correct the problem, use another disk (with more space) to perform the reconfigure operation.

Syntax

```
container reconfigure [/stripe_size{=integer}]
[/extend_fs{=boolean}][/mirror{=boolean}]
[/partition_move{=boolean}]
[/partition_size{=integer}] [/raid5{=boolean}]
[/raid10{=boolean}][/volume{=boolean}]
[/wait{=boolean}] {container} [{scsi_device}...]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array you want to reconfigure.

`{scsi_device}...`

Specifies one or more SCSI devices. Typically, you specify one or more SCSI devices when adding more drives to an array or moving an array's partitions. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), a SCSI device ID (0 through 15 inclusive), and a SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

The command uses the freespace(s) from the SCSI device(s) you specify to reconfigure the array.

Switches

`/stripe_size{=integer}`

Specifies the changed stripe size for the array. You can specify stripe sizes for the following types of arrays:

- RAID 5 array
- Multilevel stripe set:
  - Stripe set of mirror sets
  - Stripe set of volume sets
  - Stripe set of stripe sets

/extend_fs{=boolean}

Specifies whether to extend the file system so that it occupies the entire reconfigured array. If you specify TRUE, the command adds no new freespace and extends the file system so that it occupies the entire reconfigured array. (The command extends the file system after completing the reconfigure operation.) If you specify FALSE, the command adds freespace to the reconfigured array and does not extend the file system to occupy the entire reconfigured array.

This switch defaults to FALSE.

This switch applies only to those multilevel volume sets on which an NTFS file system resides. If you extend an NTFS file system, you must reboot your system in order for the extension to take effect.

If you do not specify this switch, you can later extend the file system with the container extend file_system command.

This switch is supported on Windows.

/mirror{=boolean}

Specifies whether to reconfigure the existing array into a mirror set. If you specify TRUE, the command reconfigures the existing array into a mirror set. If you specify FALSE, the command does not reconfigure the existing array into a mirror set.

This switch defaults to FALSE.

/partition_move{=boolean}

Specifies whether to move partitions instead of adding extra space (using additional disks). If you specify TRUE, the command moves partitions instead of adding extra space. If you specify FALSE, the command does not move partitions.

This switch defaults to FALSE.

/partition_size{=integer}

Specifies the partition size. All partitions must be the same size and you can specify only one partition per disk.

`/raid5{=boolean}`

Specifies whether to reconfigure the existing array into a RAID 5 array. If you specify `TRUE`, the command reconfigures the existing array into a RAID 5 array. If you specify `FALSE`, the command does not reconfigure the existing array into a RAID 5 array.

This switch defaults to `FALSE`.

`/raid10{=boolean}`

Specifies whether to reconfigure the existing array into a RAID 10 array. If you specify `TRUE`, the command reconfigures the existing array into a RAID 10 array. If you specify `FALSE`, the command does not reconfigure the existing array into a RAID 10 array.

This switch defaults to `FALSE`.

`/stripe{=boolean}`

Specifies whether to reconfigure the existing array into a stripe set. If you specify `TRUE`, the command reconfigures the existing array into a stripe set. If you specify `FALSE`, the command does not reconfigure the existing array into a stripe set.

This switch defaults to `FALSE`.

`/volume{=boolean}`

Specifies whether to reconfigure the existing array into a volume set. If you specify `TRUE`, the command reconfigures the existing array into a volume set. If you specify `FALSE`, the command does not reconfigure the existing array into a volume set.

This switch defaults to `FALSE`.

`/wait{=boolean}`

Specifies whether the command performs the array reconfiguration operation synchronously or asynchronously. If you set this switch to `TRUE`, the command performs the array reconfiguration operation synchronously, which means the command prompt does not return until the array reconfiguration operation completes. If you set this switch to `FALSE`, the command performs the array reconfiguration operation asynchronously, which means the command prompt returns immediately.

Examples

Use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a mirror set) and an array 1 (a volume set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi    Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Mirror 15.0MB            None     1:02:0 64.0KB:  15.0MB
   Tigris                                  1:03:0 64.0KB:  15.0MB
E: 1     Volume 15.0MB            NTFS     1:00:0 64.0KB:  15.0MB
```

The following example reconfigures a volume set to a stripe set:

```
AAC0>container reconfigure /partition_size=1 /stripe 1
Executing: container reconfigure /partition_size=1  /stripe=TRUE 1
```

As the command executes, note that the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!Task:100,Func:RCF Ctr:1,State:RUN 97.2%
```

For further details on status information, see page 1-13.

Use the `container list` command after using the `container reconfigure` command to display information about the array you just reconfigured, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi    Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Mirror 15.0MB            None     1:02:0 64.0KB:  15.0MB
   Tigris                                  1:03:0 64.0KB:  15.0MB
E: 1     Stripe 15.0MB            NTFS     1:00:0 64.0KB:  15.0MB
```

As the result of reconfiguring an array (in this example, array 1) with the `container reconfigure` command, the **Type** column displays **Stripe** (instead of **Volume**) to indicate that the specified array is now a stripe set instead of a volume set.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container create raid5` (page 3-21)
- `container create stripe` (page 3-27)
- `container create volume` (page 3-32)

- `container extend file_system` (page 3-40)
- `container list` (page 3-50)

container release_cache

To release the cache buffers associated with a specific array, use the `container release_cache` command. When a disk fails or you remove a disk, the cache buffers associated with specific arrays remain locked on the controller. This allows the opportunity for the disk to come back online or gives you the opportunity to replace the disk. In this case the arrays regain the previously locked buffers.

On the other hand, you may want to return these cache buffers to the global buffer pool. This command allows you to disassociate the cache buffers associated with specific arrays and release these cache buffers to the global pool.

Syntax

```
container release_cache {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array on which you want to release cache buffers.

Examples

The following example releases the cache buffers associated with array 0:

```
AAC0>container release_cache 0
Executing: container release_cache 0
```

Related Commands

`container` commands:

- `container set cache` (page 3-91)

container remove drive_letter

To remove a drive letter, use the `container remove drive_letter` command. To use the `container remove drive_letter` command, none of the array's files can be open.

Command Availability

This command is supported on Windows.

Syntax

```
container remove drive_letter {string}
```

Parameters

`{string}`

Specifies the drive letter to remove. The colon (:) after the drive letter is optional.

Examples

Before removing a drive letter from an array, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a volume set) on this controller with a drive letter `F`:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 10.0MB            NTFS    0:02:0 64.0KB: 10.0MB
```

The following example removes drive letter `F` from array 0:

```
AAC0> container remove drive_letter F
Executing: container remove drive_letter "F"
```

Use the `container list` command after using the `container remove drive_letter` command to display information about the array on which you just removed a drive letter, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Volume 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
```

As the result of removing a drive letter from an array (in this example, array 0) with the `container remove drive_letter` command:

- The **Dr** column no longer displays the drive letter assigned in a previous use of the `container assign drive_letter` command. In this example, the letter `F` no longer appears in the **Dr** column.

Related Commands

`container` commands:

- `container list` (page 3-50)

container remove failover

To remove one or more failover disks that were assigned to an array using the `container set failover` command, use the `container remove failover` command.

Syntax

```
container remove failover {container}
{scsi_device} [{scsi_device}...]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array whose assigned failover disk(s) you want to remove.

`{scsi_device}`

Specifies the ID for the SCSI device that you want to remove as a failover disk for the array specified in the `container` parameter. You previously assigned this SCSI device as a failover disk with the `container set failover` command.

For further details, see *scsi_device* on page 1-10.

`{scsi_device}...`

Specifies the ID(s) for any other SCSI device(s) you want to remove as failover disk(s) for the array specified in the `container` parameter.

Examples

Before removing one or more failover disks from an array, use the `container show failover` command to obtain information about any existing failover disks assigned to arrays.

As the following example shows, there is an array 0 that has two failover disks assigned to it on this controller:

```
AAC0>container show failover
Executing: container show failover
container Scsi C:ID:L
--------- ---------------------------------
  0       0:03:0  0:04:0
```

The following example removes SCSI device 0:03:0 as an assigned failover disk for array 0:

```
AAC0>container remove failover 0 (0,3,0)
Executing: container remove failover 0 (CHANNEL=0,ID=3,LUN=0)
```

Use the `container show failover` command after removing a failover disk from an array's failover disk list with the `container remove failover` command to display information about the array's failover disk list, as in the following example:

```
AAC0>container show failover
Executing: container show failover
container Scsi C:ID:L
--------- ---------------------------------
  0       0:04:0
```

As the display shows, SCSI device (0,03,0) is no longer in the failover disk list for array 0.

Related Commands

`container` commands:

- `container set failover` (page 3-95)
- `container show failover` (page 3-108)

container remove file_system

To remove the file system from an array, use the `container remove file_system` command. Use this command when the file system on the array is no longer needed.

This command fails if there are open files on the array.

Command Availability

This command is supported on Windows.

Syntax

```
container remove file_system {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array from which you want to remove the file system.

Examples

Before removing a file system from an array, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there is an array 0 (a volume set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
```

The following example removes the file system from array 0:

```
AAC0>container remove file_system 0
Executing: container remove file_system 0
```

Use the `container list` command after removing a file system from an array with the `container remove file_system` command to display information about the container, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 10.0MB            None    0:02:0 64.0KB:  10.0MB
```

As a result of removing a file system from an array, the **Usage** column displays **None** (instead of NTFS) to indicate that the specified array no longer has a file system.

Related Commands

`container` commands:

- `container format` (page 3-50)
- `container list` (page 3-50)

container remove global_failover

To remove one or more failover disks that were assigned to all arrays through the `container set global_failover` command, use the `container remove global_failover` command.

Syntax

```
container remove global_failover {scsi_device}
[{scsi_device}...]
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI device that you want to remove as a failover disk for all arrays. You previously assigned this SCSI device as a failover disk with the `container set global_failover` command.

For further details, see *scsi_device* on page 1-10.

`{scsi_device}...`

Specifies the ID(s) for any other SCSI device(s) you want to remove as failover disk(s) for all arrays. You previously assigned these SCSI device(s) as failover disk(s) with the `container set global_failover` command.

Examples

Before removing one or more failover disks from all arrays, use the `container show failover` command to obtain information about any existing failover disks assigned to arrays.

As the following example shows, there is an array 0 that has SCSI devices (1,2,0) and (1,3,0) assigned to it.

```
AAC0>container show failover
Executing: container show failover
container Scsi C:ID:L
--------- ---------------------------------
 GLOBAL    1:02:0  1:03:0
 0         --- No Devices Assigned ---
```

The following example removes SCSI device 1:02:0 as an assigned failover disk for all arrays:

```
AAC0>container remove global_failover (1,2,0)
Executing: container remove global_failover (CHANNEL=1,ID=2,LUN=0)
```

Use the `container show failover` command after removing a failover disk from an array's failover disk list with the `container remove failover` command to display information about the array's failover disk list, as in the following example:

```
AAC0>container show failover
Executing: container show failover
container Scsi C:ID:L
--------- --------------------------------
 GLOBAL   1:03:0
  0       --- No Devices Assigned ---
```

As the display shows, SCSI device (1,2,0) is no longer in the failover disk list for all arrays. However, SCSI device (1,3,0) remains in the failover disk list for all arrays.

Related Commands

`container` commands:

- `container set failover` (page 3-95)
- `container set global_failover` (page 3-97)
- `container show failover` (page 3-108)

container restore RAID5

To restore a RAID 5 array, use the `container restore RAID5` command. Typically, you use this command to restore a RAID 5 array that contains one or more dead partitions. This command does not always succeed; therefore, use it only if all other measures fail.

One situation where the command might prove useful is for those situations where you may have inadvertently pulled a drive and then almost immediately put it back. The act of pulling the drive may cause some dead partitions to appear. You can then restore the RAID 5 array using this command.

Syntax

```
container restore RAID5 {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array (a RAID 5 array) to restore.

Examples

Before restoring a RAID 5 array, use the `container list` command to display information about the RAID 5 array. If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point).

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     RAID-5 30.0MB     64KB   None    0:02:0 64.0KB!  10.0MB
                                          0:03:0 64.0KB:  10.0MB
                                          0:04:0 64.0KB:  10.0MB
                                          0:05:0 64.0KB:  10.0MB
```

The previous example shows that there is one dead partition on this RAID 5 array. The following example uses the `container restore RAID5` command to attempt to restore the RAID 5 array:

```
AAC0>container restore RAID5 0
Executing: container restore RAID5 0
```

Use the `container list` command after using the `container restore RAID5` command to display information about the RAID 5 array:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     RAID-5 30.0MB     64.0KB None    0:02:0 64.0KB:  10.0MB
                                          0:03:0 64.0KB:  10.0MB
                                          0:04:0 64.0KB:  10.0MB
                                         0:05:0 64.0KB:  10.0MB
```

Note that in this case the command successfully restored the RAID 5 array as indicated by the replacement of the exclamation point (“!”) with a colon (“:”).

Related Commands

`container` commands:

- `container create raid5` (page 3-21)
- `container list` (page 3-50)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)

container scrub

To scrub a redundant array, use the `container scrub` command. A mirror set, a multilevel array of mirror sets, and a RAID 5 array are examples of redundant arrays. For a mirror set or a multilevel array of mirror sets, this means the command reconstructs the data on both mirror halves (partitions), if found to be different. For RAID 5 arrays, the command recalculates and replaces, if necessary, the parity information.

Syntax

```
container scrub [/io_delay{=integer}] [/
no_repair{=boolean}][/wait{=boolean}] {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the redundant array to scrub.

Switches

`/io_delay{=integer}`

Specifies the number of milliseconds the controller waits between the I/O operations required to scrub the redundant array. The I/O delay value is not preserved between reboots of the operating system. Valid values are 0 through 100.

`/no_repair{=boolean}`

Specifies whether the command performs the scrub operation on the redundant array without repairing the error. If you set this switch to `TRUE`, the command performs the scrub action without repairing the error. If you set this switch to `FALSE`, the command performs the scrub action and repairs any errors.

`/wait{=boolean}`

Specifies whether the command performs the scrub action synchronously or asynchronously. If you set this switch to `TRUE`, the command performs the scrub action synchronously, which means the command prompt does not return until the scrub action completes. If you set this switch to `FALSE`, the command performs the scrub action asynchronously, which means the command prompt returns immediately.

Examples

Before scrubbing a redundant array, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a mirror set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
D: 0     Mirror 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
                                         0:02:0 10.0MB:  10.0MB
```

The following example synchronously scrubs array 0 using a delay time of 5 milliseconds:

```
AAC0>container scrub /io_delay=5 /wait=TRUE 0
Executing: container scrub /io_delay=5  /wait=TRUE 0
```

As the command executes, note the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!Task:100,Func:MSC Ctr:0,State:RUN 97.2%
```

For further details on status information, see page 1-13.

Use the `container list` command after using the `container scrub` command to display information about the scrubbed array, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
D: 0     Mirror 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
                                           0:02:0 10.0MB:  10.0MB
```

As the example shows, there is no difference between the array list display prior to and after the scrub operation.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container create mmirror` (page 3-8)
- `container create raid5` (page 3-21)
- `container list` (page 3-50)

container set cache

To set cache parameters for a specific array, use the `container set cache` command. You can use this command only if a native operating system's file system (for Windows, NTFS, or FAT) resides on the array.

The CLI prevents you from setting cache parameters if the array is involved in a reconfiguration operation. The CLI disables the cache parameters on the array involved in a reconfiguration operation. The CLI re-enables the cache parameters (assuming they were previously set) on the array when the reconfiguration operation completes.

The controller provides two global cache buffer pools available to arrays: a volatile read-ahead cache and a nonvolatile NVRAM write-back cache. Collectively, these global caches are referred to as the raw array cache.

The `container set cache` command allows you to set several characteristics associated with the raw array cache. These characteristics are embodied in the switches for the command.

Notes

Some controllers may not support the NVRAM write-back cache. Additionally, some controllers may not support the ability to enable the NVRAM write-back cache.

Syntax

```
container set cache [/
read_cache_enable{=boolean}} [/
unprotected{=boolean}]
[/write_cache_enable{=boolean}] {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array on which you want to set cache parameters.

Switches

`/read_cache_enable{=boolean}`

Specifies whether to enable the read-ahead cache. If you set this switch to `TRUE`, the command enables the read-ahead cache for the specified array. This switch should always be enabled to optimize performance, unless your application--which is unlikely--is doing completely random reads.

This switch defaults to `TRUE`.

To disable the read-ahead cache for the specified array, set this switch to `FALSE`. Note that if you disable the read-ahead cache, no other characteristics can be set.

`/unprotected{=boolean}`

Specifies whether to set the array's NVRAM write-back cache to disable, enable when protected, or enable always. You use this switch in conjunction with the `/write_cache_enable` switch to accomplish the desired setting. See the `write_cache_enable` switch for more information.

`/write_cache_enable{=boolean}`

Specifies whether to set the array's NVRAM write-back cache to disable, enable when protected, or enable always. You use this switch in conjunction with the `/unprotected` switch to accomplish the desired setting.

Table 3-5 summarizes the values for the `/write_cache_enable` and `/unprotected` switches:

Table 3-5 Switch Values and Results

/write_cache_enable Switch Setting	/unprotected Switch Setting	Result
FALSE	FALSE	The NVRAM write-back cache setting for the specified array is disabled.
FALSE	TRUE	Not allowed. An appropriate error message displays.
TRUE	FALSE	The NVRAM write-back cache setting for the specified array is enabled when protected. This means the controller enables the array's NVRAM write-back cache only if a battery is present and its status is OK.
TRUE	TRUE	The NVRAM write-back cache setting for the specified array is enabled always. This means the controller forces the enabling of the array's NVRAM write-back cache even if write data could be lost due to no battery or a bad battery.

Examples

Before setting cache parameters for a specific array, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there is an array 0 (a volume set) on this controller with no file system on it (as evidenced by **None** displayed in the **Usage** column):

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Volume 15.0MB            None    0:02:0 64.0KB:  15.0MB
```

The following example sets cache parameters for array 0 by accepting all of the defaults:

```
AAC0>container set cache 0
Executing: container set cache 0
```

After setting cache parameters for a specific array, use the `container show cache` command to display cache parameter information, as in the following example:

```
AAC0>container show cache 0
Executing: container show cache 0
Global container Read Cache Size  : 5345280
Global container Write Cache Size : 1970176
Read Cache Status         : ENABLED
Write Cache Status        : ENABLED
Stream Detection Status   : ENABLED
```

Related Commands

`container` commands:

- `container release_cache` (page 3-77)
- `container show cache` (page 3-103)

container set failover

To assign an automatic failover disk(s) for a single array, use the `container set failover` command. If the array was already assigned a failover disk(s), the command adds the specified disk(s) to the array's list of failover disk(s). Although all array types accept failover assignments, only mirror set and RAID 5 array (redundant) array types use the failover assignment if a disk fails.

Syntax

```
container set failover {container} {scsi_device}
[{scsi_device}...]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array on which to assign an automatic failover disk(s).

`{scsi_device}`

Specifies the ID for the SCSI device that you want to assign as a failover disk to the array specified in the `container` parameter. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), a SCSI device ID (0 through 15 inclusive), and a SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

`{scsi_device}...`

Specifies the ID(s) for additional SCSI device(s) that you want to assign as failover disk(s) to the array specified in the `container` parameter.

Examples

Before assigning a SCSI device as a failover disk to an array, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there is an array 0 (a mirror set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage    C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
D: 0     Mirror 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
                                            0:02:0 10.0MB:  10.0MB
```

The following example assigns two SCSI failover disks to array 0. If you assign only one SCSI disk as a failover device, ensure that there is sufficient freespace. If you assign more than one SCSI disk as failover devices, you need ensure that only one of these disks has sufficient space. The reason for this is that the failover operation will look for the disk that has enough space.

```
AAC0>container set failover 0 (0,3,0) (0,4,0)
Executing: container set failover 0 (CHANNEL=0,ID=3,LUN=0)
(CHANNEL=0,ID=4,LUN=0)
```

Use the `container show failover` command after using the `container set failover` command to display information about the array just assigned failover disk(s).

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container remove failover` (page 3-80)
- `container show failover` (page 3-108)

container set global_failover

To assign an automatic failover disk(s) for all arrays, use the `container set global_failover` command. If the array was already assigned a failover disk(s), the command adds the specified disk(s) to the array's list of failover disk(s). Although all array types accept failover assignments, only mirror set and RAID 5 array (redundant) array types use the failover assignment if a disk fails.

Syntax

```
container set global_failover {scsi_device}
[{scsi_device}...]
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI device that you want to assign as a failover disk. The command assigns this disk to all arrays. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), a SCSI device ID (0 through 15 inclusive), and a SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

`{scsi_device}...`

Specifies the ID(s) for additional SCSI device(s) that you want to assign as failover disk(s) to all arrays.

Examples

Before assigning a SCSI device as a failover disk to all arrays, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a mirror set) on this controller:

```
AAC0>container list

Executing: container list

   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Mirror 15.0MB            None    1:02:0 64.0KB:  15.0MB
   Tigris                                1:03:0 64.0KB:  15.0MB
```

The following example assigns two SCSI failover disks to all arrays. If you assign only one SCSI disk as a failover device, ensure that there is sufficient freespace. If you assign more than one SCSI disk as failover devices, you need ensure that only one of these disks has sufficient space. The reason for this is that the failover operation will look for the disk that has enough space.

```
AAC0>container set global_failover (1,2,0) (1,3,0)
Executing: container set global_failover (CHANNEL=1,ID=2,LUN=0)
(CHANNEL=1,ID=3,LUN=0)
```

Use the `container show failover` command (with or without the `/global` switch) after using the `container set failover` command to display information about the array(s) just assigned failover disk(s), as in the following example:

```
AAC0>container show failover /global
Executing: container show failover /global=TRUE
container Scsi C:ID:L
--------- ---------------------------------
 GLOBAL   1:02:0  1:03:0
```

As the example shows, the display indicates that SCSI devices (1,2,0) and (1,3,0) are assigned as failover devices for all arrays (the word **GLOBAL** appears in the **array** column).

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container remove failover` (page 3-80)
- `container remove global_failover` (page 3-84)
- `container set failover` (page 3-95)
- `container show failover` (page 3-108)

container set io_delay

To set the I/O delay for an array, use the `container set io_delay` command. You can specify an I/O delay for the following array-related tasks:

- Creating a mirror set
- Creating a multilevel array of mirror sets from a multilevel array of volume sets
- Scrubbing a redundant array

You use this command to change the I/O delay from the one you specified in the command for the previously listed tasks. The I/O delay value is not preserved between reboots of the operating system.

Syntax

```
container set io_delay {container} {integer}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array on which to set the I/O delay.

`{integer}`

Specifies the number of milliseconds the controller waits between the I/O operations required to perform the specified background task. The I/O delay value is not preserved between reboots of the operating system. Valid values are 0 through 100.

Examples

Before setting the I/O delay on an array, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there is an array 0 (a volume set) on this controller:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
```

The following example creates a mirror set asynchronously and sets an I/O delay of 20 milliseconds for array 0. You need to perform the create mirror task asynchronously. Otherwise, you cannot change the I/O delay:

```
container create mirror /io_delay=20 0 (0,2,0)
Executing container create mirror /io_delay=20 0 (CHANNEL=0,ID=2,LUN=0)
```

The following example shows how to change the I/O delay to 40 milliseconds on the currently running create mirror task:

```
AAC0>container set io_delay 0 40
Executing: container set io_delay 0 40
```

Use the `container list` command after using the `container set io_delay` command to display information about the array, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
```

As the example shows, there is no difference between the array list display prior to and after the set I/O delay operation.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container create mmirror` (page 3-8)
- `container list` (page 3-50)
- `container scrub` (page 3-88)

container set label

To assign a new label to the specified array or to assign a label to an array that has no label, use the `container set label` command.

The following commands have a `/label` switch that allows you to assign a label to the array when you create it:

- `container create mstripe` (a multilevel stripe set)
- `container create mvolume` (a multilevel volume set)
- `container create raid5` (a RAID 5 array)
- `container create stripe` (a stripe set)
- `container create volume` (a volume set)

The label you specify with the `container set label` command and the `/label` switch associated with the previously listed `container create` commands is not the label that displays in Windows Explorer. The label displayed by Windows Explorer comes from the label specified with the `container format` command.

Syntax

```
container set label {container} {string}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to which you want to assign a label.

`{string}`

Specifies the label you want to assign to the array. The label is a string of up to sixteen characters.

Examples

Before assigning a label to a specific array, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is an array 0 (a volume set) on this controller with no label assigned to it (as evidenced by no label in the **Num Label** column):

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Volume 15.0MB            None    0:02:0 64.0KB:  15.0MB
```

The following example assigns the label `Tigris` to array 0:

```
AAC0>container set label 0 Tigris
Executing: container set label 0 "Tigris"
```

Use the `container list` command after using the `container set label` command to display the new label, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
   0     Volume 15.0MB            None    0:02:0 64.0KB:  15.0MB
   Tigris
```

As the example shows, the new label `Tigris` now appears in the **Num Label** column.

Related Commands

`container` commands:

- `container create mstripe` (page 3-12)
- `container create mvolume` (page 3-17)
- `container create raid5` (page 3-21)
- `container create stripe` (page 3-27)
- `container create volume` (page 3-32)
- `container format` (page 3-50)

container show cache

To display the current cache parameters associated with a specific array, use the `container show cache` command. Typically, you use this command after setting cache parameters for a specific array using the `container set cache` command.

Syntax

```
container show cache {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array whose associated cache parameters you want to display.

Examples

Before setting cache parameters for a specific array, use the `container list` command to obtain information about any existing arrays.

As the following example shows, there is an array 0 (a volume set) on this controller with no file system on it (as evidenced by **None** displayed in the **Usage** column). Assume the cache parameters for this array were previously set with the `container set cache` command:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
    0     Volume 15.0MB            None    0:02:0 64.0KB:  15.0MB
```

The following example displays cache parameters for array 0:

```
AAC0>container show cache 0
Executing: container show cache 0
Global container Read Cache Size  : 5345280
Global container Write Cache Size : 1970176
Read Cache Setting        : ENABLED
Write Cache Setting       : ENABLED
Write Cache Status        : ENABLED
```

The following list provides a brief description of the items that appear as a result of using the `container show cache` command:

- Global array Read Cache Size

  This item indicates the number of blocks devoted to the read cache.

- Global array Write Cache Size

  This item indicates the number of blocks devoted to the write cache.

- Read Cache Setting

  This item indicates whether the read cache is set for the array. The value `ENABLED` indicates the read cache is set for this array. The value `DISABLED` indicates the read cache is disabled for this array.

- Write Cache Setting

  This item indicates whether the write cache is set for the array. Table 3-6 displays the possible values for this setting.

Table 3-6 Values for the Write Cache Setting

Value	Meaning
`DISABLE`	The write cache is disabled for this array
`ENABLE ALWAYS`	The write cache setting for the specified array is enabled always. This means the controller forces the enabling of the array's write cache even if write data could be lost due to no battery or a bad battery.
`ENABLE WHEN PROTECTED`	The write cache setting for the specified array is enabled when protected. This means the controller enables the array's write cache only if a battery is present and its status is OK.

- Write Cache Status

  This item indicates the current status of the write cache for the array and, where appropriate, the status of the battery.

  Table 3-7 displays the values for Write Cache Status:

Table 3-7 Values for Write Cache Status

Value	Meaning
Active, not protected	The status of the write cache for this array is as follows: ■ Active – The write cache can accept write operations from the array. ■ Not protected – The write cache is force-enabled. Thus, the write cache accepts write operations even if write data could be lost due to no battery present or a bad battery on the controller. ■ This status also indicates that the battery status is OK.
Active, not protected, battery low	The status of the write cache for this array is as follows: ■ Active – The write cache can accept write operations from the array. ■ Not protected – The write cache is force-enabled. Thus, the write cache accepts write operations even if write data could be lost due to no battery present or a bad battery on the controller. ■ This status also indicates that the battery status is low.
Active, not protected, battery not present	The status of the write cache for this array is as follows: ■ Active – The write cache can accept write operations from the array. ■ Not protected – The write cache is force-enabled. Thus, the write cache accepts write operations even if write data could be lost due to no battery present or a bad battery on the controller. ■ This status also indicates that the battery is not present on the controller.

Table 3-7 Values for Write Cache Status (Continued)

Value	Meaning
Active, not protected, battery reconditioning	The status of the write cache for this array is as follows: ■ Active – The write cache can accept write operations from the array. ■ Not protected – The write cache is force-enabled. Thus, the write cache accepts write operations even if write data could be lost due to no battery present or a bad battery on the controller. ■ This status also indicates that the battery is being reconditioned.
Active, protected	The status of the write cache for this array is as follows: ■ Active – The write cache can accept write operations from the array. ■ Protected – The write cache is enabled only if a battery is present and its status is OK. ■ This status also indicates that the controller's battery is present and its status is OK.
Inactive, battery low	The status of the write cache for this array is as follows: ■ Inactive – The write cache cannot accept write operations from the array. ■ Battery low – This controller's battery power is low.
Inactive, battery not present	The status of the write cache for this array is as follows: ■ Inactive – The write cache cannot accept write operations from the array. ■ Battery not present – This controller has no battery.
Inactive, battery reconditioning	The status of the write cache for this array is as follows: ■ Inactive – The write cache cannot accept write operations from the array. ■ Battery low – The controller's battery is being reconditioned.

Table 3-7 Values for Write Cache Status (Continued)

Value	Meaning
Inactive, cache disabled	The status of the write cache for this array is as follows: ■ Inactive – The write cache cannot accept write operations from the array. ■ Cache disabled – The user disabled the write cache by using the `container set cache` command and setting the appropriate switches.
Inactive, write not supported	The status of the write cache for this array is as follows: ■ Inactive – The write cache cannot accept write operations from the array. ■ Battery not supported – This controller does not support a battery.

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container set cache` (page 3-91)
- `container show cache` (page 3-103)

container show failover

To display a list of failover disks assigned to an array, use the `container show failover` command. You assign failover disks to an array with the `container set failover` command.

Syntax

```
container show failover [{container}]
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array whose assigned failover disk(s) you want to display. If you do not specify this parameter, the command displays all arrays and their assigned failover disk(s).

Examples

The following example shows the failover disks assigned to array 0:

```
AAC0>container show failover 0
Executing: container show failover 0
container Scsi C:ID:L
--------- --------------------------------
  0       0:03:0  0:04:0
```

The example shows that array 0 was assigned disks 0:03:0 and 0:04:0 as failover disks.

Related Commands

`container` commands:

- `container remove failover` (page 3-80)
- `container set failover` (page 3-95)

container split

To split a mirror set or a multilevel array of mirror sets into two separate single-partition volume sets or two multilevel arrays of single-partition volume sets, use the `container split` command. Once this command completes execution, the two volume sets cannot be merged.

Before using this command to split an array, shut down system applications (such as, databases) in order to flush application data to the controller.

When you split a mirror set or a multilevel array of mirror sets, the system creates a new (referred to as the split) array. The split array contains half of the storage data from the original (referred to as the master) array (mirror set). In addition, the split array has an identical copy of the data from the old disk at the time of the split operation.

In Windows, the split array's file system may be inconsistent and you should verify and repair it as follows:

- Assign a drive letter to the split array using the `container assign drive_letter` command.
- Run the DOS `CHKDSK` command with the `/F` switch. For details, refer to your Windows documentation.

On UnixWare, before splitting a mirror set that contains a volume table of contents, run the `edvtoc` command and change the volume label to a series of twelve spaces. You can then use the `container split` command to split the mirror set.

Syntax

```
container split {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array (mirror set or multilevel array of mirror sets) to split into two separate arrays.

Examples

Before splitting a mirror set or a multilevel array of mirror sets, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is one existing array (array 0, a mirror set) on this controller at the time the mirror set is split:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Mirror 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
                                         0:02:0 10.0MB:  10.0MB
```

The following example shows how to split a mirror set:

```
AAC0>container split 0
Executing: container split 0
container 1 created
```

On UNIX systems, the message displayed after you execute the `container split` command includes the root special file associated with the newly created split array.

Use the `container list` command after using the `container split` command to display information about the split mirror set, as in the following example:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi   Partition
Dr Label Type   Size   Ctr Size   Usage   C:ID:L Offset:Size
-- ----- ------ ------ --- ------ ------- ------ -------------
F: 0     Volume 10.0MB            NTFS    0:02:0 64.0KB:  10.0MB
   1     Volume 10.0MB            NTFS    0:02:0 10.0MB:  10.0MB
```

The following list describes the change to the display as the result of splitting a mirror set with the `container split` command:

- The **Num Label** column displays the IDs (in the example, 0 and 1) of the split mirror set. array 0 is the master array; array 1 is the split array.

  This column also displays the label(s) assigned to the array(s) when the array(s) was created. If no label(s) was assigned to the array(s) then no label(s) appears in the column.

  On UNIX systems, the root special file associated with the array also appears in this column.

- The **Type** column displays **Volume** as the array type for a split mirror set.

- The **Usage** column displays NTFS to indicate that file systems exist on both arrays. You can create an NTFS or FAT file system on an array by using the `container format` command.
- The **Scsi C:ID:L** column displays the SCSI device IDs for the disk on which the split mirror set (now two volume sets) reside. In the example, the SCSI device ID is 0:02:0.
- The **Partition Offset:Size** column displays the partition offsets and sizes for the partition associated with the split mirror set (now two volume sets).

  If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container create mmirror` (page 3-8)
- `container list` (page 3-50)
- `container unmirror` (page 3-114)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)

container unlock

Caution: Use the `container unlock` command only under the direction of Technical Support.

To unlock an array so it can be moved, deleted, made read-only, and used to create a multilevel array, use the `container unlock` command.

Syntax

```
container unlock {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the array to unlock.

Examples

Before unlocking an array, use the `container list` command (with the `/full` switch) to obtain information about any existing arrays. As the following example shows, there is an array 0 (a volume set) on this controller. Note that the **Lk** column displays an L, which indicates that array 0 was previously locked with the `container lock` command:

```
AAC0>container list /full
Executing: container list /full=TRUE
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    Lk
-- ----- ------ ------ --- ------ ------- --------  --
F: 0     Volume 10.0MB               NTFS    0:02:0    L
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **Lk** column.

The following example shows how to unlock array 0:

```
AAC0>container unlock 0
Executing: container unlock 0
```

Use the `container list` command (with the `/full` switch) after using the `container unlock` command to display information about the unlocked array, as in the following example:

```
AAC0>container list /full
Executing: container list /full=TRUE
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    Lk
-- ----- ------ ------ --- ------ ------- --------  --
F: 0     Volume 10.0MB            NTFS    0:02:0
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **Lk** column.

As a result of unlocking an array (in this example, array 0) with the `container unlock` command, the **Lk** column displays a blank (instead of an L) to indicate that the specified array is now unlocked.

Related Commands

`container` commands:

- `container list` (page 3-50)
- `container lock` (page 3-59)

container unmirror

To unmirror a mirror set, resulting in a single-partition volume set and freespace, use the `container unmirror` command. Note that there is no equivalent command for a multilevel array of mirror sets. To unmirror a multilevel array of mirror sets, use this command for each underlying mirror set.

Syntax

```
container unmirror {container}
```

Parameters

`{container}`

Specifies the ID number (0 to 63) of the mirror set to unmirror. You previously created the mirror set by using the `container create mirror` or `container create mmirror` command.

Examples

Before unmirroring a mirror set, use the `container list` command to obtain information about any existing arrays. As the following example shows, there is one existing array (array 0, a mirror set) on this controller at the time the mirror set is unmirrored:

```
AAC0>container list
Executing: container list
   Num          Total  Oth Stripe          Scsi
Dr Label Type   Size   Ctr Size   Usage   C:ID:L    State
-- ----- ------ ------ --- ------ ------- --------  -----
   0     Mirror 10.0MB            NTFS    0:02:0    Normal
                                          0:03:0
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **State** column.

The following example shows how to unmirror a mirror set from the mirror set that resides on SCSI device (0,3,0):

```
AAC0>container unmirror 0
Executing: container unmirror 0
```

Note that the example eliminates some items in the `container list` display so that you can see an example of the **State** column.

The following list describes the changes to the display as a result of using the `container unmirror` command:

- The **Type** column displays **Volume**, which indicates that the mirror set was created from a Volume set.
- The **State** column displays a blank space instead of Normal. The Normal state is displayed only for mirror sets.
- The **Scsi C:ID:L** column displays only one SCSI ID.
- The **Partition Offset:Size** column displays only one partition offset and size.

  If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

Related Commands

`container` commands:

- `container create mirror` (page 3-5)
- `container list` (page 3-50)

`disk` commands:

- `disk remove dead_partitions` (page 5-10)

4

controller Commands

In this Chapter

controller details

To display details about the currently opened controller, use the `controller details` command. These details include the controller type and software revision levels.

Syntax

```
controller details
```

Examples

The following example shows sample output as a result of using the `controller details` command:

```
AAC0> controller details
Executing: controller details
Controller Information
----------------------
         Remote Computer: YETI
             Device Name: aac0
         Controller type: Adaptec 5400S
             Access Mode: READ-WRITE
Controller serial number: Last six digits = 8A277A
      Number of Channels: 4
     Devices per Channel: 15
          Controller CPU: Strong Arm 110
    Controller CPU speed: 233 Mhz
       Controller Memory: 144 Mbytes
           Battery State: Ok

Component Revisions
-------------------
                CLI: 2.1-0  (Build #2881)
                API: 2.1-0  (Build #2874)
    Miniport Driver: 2.1-0  (Build #2874)
Controller Software: 2.1-0  (Build #2874)
    Controller BIOS: 2.0-1  (Build #2874)
Controller Firmware:        (Build #2874)
Controller Hardware: 1.0
```

The following sections provide more information on the items that the `controller details` command displays.

Device Name Item

The `Device Name:` item displays the name of the controller. This is the controller name specified when you opened the controller with the `open` command. In the example, the controller name is aac0.

The Controller type Item

The `Controller type:` item displays the controller type. In the example, the controller type is Adaptec 5400S.

The Access Mode Item

The `Access Mode:` item displays the access mode of the controller. The command displays the value `READ-WRITE` if you opened the controller for read-write operations. Otherwise, it displays `READ-ONLY` if you opened the controller for read-only operations.

The Controller serial number Item

The `Controller serial number:` item displays the controller serial number. The command displays the last six hexadecimal characters of the controller serial number. These six hexadecimal characters match the last six characters displayed on the "SN =" sticker located on the controller itself.

The Number of Channels and Devices per Channel Items

The `Number of Channels:` item displays the number of channels actually found on the controller. In the example, the number of channels on the controller is four.

The `Devices per Channel:` item displays the number of SCSI disk devices that can be put onto each SCSI channel for this controller. In the example, 15 SCSI disks can be put onto each SCSI channel.

The Controller CPU Item

The `Controller CPU:` item displays the type of central processing unit (CPU) that the controller uses. The `controller details` command can display the controller CPU type listed in Table 4-1.

Table 4-1 Controller CPU Types

Controller CPU Type	Meaning
Strong Arm 110	The CPU type on the controller is a Digital Strong Arm 110. The example uses this CPU type.

The Controller CPU Speed and Controller Memory Items

The `Controller CPU Speed:` item (if available) displays the clock speed (in megahertz) of the CPU that resides on the controller. In the example, the speed is 233 Mhz.

The `Controller Memory:` item displays the total amount of memory on the controller that programs and buffer data can use. In the example, the total memory is 48 MB.

The Battery State Item

The `Battery State:` item displays the state of the controller's battery. The `controller details` command can display the battery states listed in Table 4-2.

Table 4-2 Battery States

State	Meaning
Ok	The battery is in good working condition.
Reconditioning	The battery is in the recondition state.
Low	The battery is low on power.
Not Present	The battery has no power or there is no battery present on the controller.

The Component Revisions Items

Table 4-3 describes each of the `Component Revisions:` items. This information is useful if you ever need to contact a technical support representative.

Table 4-3 Component Revision Items

Component Revision Item	Meaning
CLI:	Displays the revision level of the Command Line Interface. In the example, the revision level is 2.1-0 (Build #2881).
API:	Displays the revision level of the controller's application programming interface (API). In the example, the revision level is 2.1-0 (Build #2881).

Table 4-3 Component Revision Items (Continued)

Component Revision Item	Meaning
Service:	Displays the revision level of the network services software. This item displays only if the open controller resides on a remote computer. In the example, the revision level is 2.1-0 (Build #2881).
Remote API:	Displays the revision level of the API that the network services software uses on the remote computer. This item displays only if the open controller resides on a remote computer. In the example, the revision level is 2.1-0 (Build #2881).
Miniport Driver:	Displays the version number of the miniport portion of the AAC.SYS device driver. This item displays only if the controller runs the AAC.SYS device driver. In the example, the revision level is 2.1-0 (Build #2874).
Controller Software:	Displays the revision level of the controller software. In the example, the revision level is 2.1-0 (Build #2874).
Controller BIOS:	Displays the revision level of the controller BIOS. In the example, the revision level is 2.0-1 (Build #2874).
Controller Firmware:	Displays the revision level of the firmware. In the example, the revision level is (Build #2874).
Controller Hardware:	Displays the version of the controller hardware. In the example, the version is 1.0.

Related Commands

General control commands:

- `open` (page 2-6)

controller firmware compare

To compare the contents of each of the flash components on a controller to the corresponding image in a pair of user flash image (UFI) files and indicate whether they match, use the `controller firmware compare` command.

Syntax

```
controller firmware compare
[/C[{controller_ID}][{controller_ID}...]
[/D{UFI_file_path}]
```

Switches

`/C{controller_ID}`

Specifies the controller ID representing the set of controllers on which to perform the firmware comparison.

If you do not specify this switch, this command compares the flash components on controller 0.

`/D{UFI_file_path}`

Specifies the path where the pair of UFI files are located.

If you do not specify this switch, this command compares the flash components against the pair of UFI files in the current default drive and directory.

Examples

The following example compares the contents of a controller's flash components to the corresponding image in a pair of UFI files.

```
AAC0> controller firmware compare D:\
Executing: controller firmware compare "D:\"
The controller's firmware matches the firmware in the specified flash
image files.
```

Related Commands

`controller` commands:

- `controller firmware save` (page 4-7)
- `controller firmware update` (page 4-9)

controller firmware save

Note: This command is not supported in Linux.

To save the contents of a controller's flash in a pair of user flash image (UFI) files, use the `controller firmware save` command. The names of the pair of UFI files are based on the controller type and cannot be changed.

Syntax

```
controller firmware save
[/C[{controller_ID}][{controller_ID}...] {/
D{UFI_file_path}]
```

Switches

`/C{controller_ID}`

Specifies the controller ID representing the set of controllers on which to perform the firmware save.

If you do not specify this switch, this command saves the flash components on controller 0.

`/D{UFI_file_path}`

Specifies the path where the pair of UFI files are located. Use this switch to specify the drive and directory where you want to create the pair of UFI files.

If you do not specify this switch, this command creates the pair of UFI files in the current default drive and directory.

Examples

The following example saves the contents of the opened controller's flash components to a pair of UFI files. In the example, one file is specified. A second file is also created with the same name, except that the number of file name suffix is incremented by 1.

```
AAC0> controller firmware save D:\aac\backup...
Executing: controller firmware save "D:\aac\backup..."
```

Related Commands

`controller` commands:

- `controller firmware compare` (page 4-6)
- `controller firmware update` (page 4-9)

controller firmware update

To update a controller's flash components from the flash image data in a pair of user flash image (UFI) files, use the `controller firmware update` command. This command can update the flash components on a single controller or multiple controllers.

Syntax

```
controller firmware update
[/C[{controller_ID}][{controller_ID}...]
{/D{UFI_file_path}]
```

Switches

`/C{controller_ID}`

Specifies the controller ID representing the set of controllers on which to perform the firmware update.

If you do not specify this switch, the firmware update is performed on controller 0.

`/D{UFI_file_path}`

Specifies the path where the pair of UFI files are located.

If you do not specify this switch, this command looks for or creates a pair of UFI files in the current default drive and directory.

Examples

The following example updates a controller's firmware.

```
AAC0> controller firmware update D:\...
Executing: controller firmware compare "D:\..."
!!The controller's firmware has been successfully updated; you need to
restart the system!!.
```

Related Commands

`controller` commands:

- `controller firmware compare` (page 4-6)
- `controller firmware save` (page 4-7)

controller list

To list all controllers on a specified computer or display specific information about the currently opened controller, use the `controller list` command.

Syntax

```
controller list [/domain{=string}] [{string}]
```

Parameters

`{string}`

Specifies the computer name from which you want to display a list of controllers or specific information about the currently opened controller. If you do not specify a computer name, the command displays only the controllers on the local computer.

Switches

`/domain{=string}`

Specifies the domain in which the specified computer resides.

If you do not specify this switch, the command assumes the local domain.

This switch is supported on Windows.

Examples

The following example shows how to list all controllers on a local computer:

```
AAC0> controller list
Executing: controller list
Adapter Name      Adapter Type       Availability
------------      ------------       ------------
\\.\AAC0          Adaptec 5400S      read/write
```

The following sections provide descriptions of the items in the display.

The Adapter Name and Adapter Type Items

The `Adapter Name` item displays the name(s) of all controllers on the local or remote computers. In the example, there is only one controller on the local computer and it is called AAC0.

The `Adapter Type` item displays the controller type. In the example, the controller type is Adaptec 5400S.

The Availability Item

The `Availability` item displays the availability of the controller. Table 4-4 lists the values that can display in this item.

Table 4-4 Controller Availability Status

Controller Availability	Meaning
Unavailable	The controller is not available.
Unknown	The controller availability is unknown.
read only	The controller is available for read-only access.
read/write	The controller is available for read/write access. In the example, the controller is available for read/write access.

Related Commands

`controller` commands:

- `controller details` (page 4-2)

controller pause_io

To pause all I/O activity on the currently opened controller, use the `controller pause_io` command. While I/O activity is paused, you can make changes to devices attached to the controller without rebooting. For example, you can add, remove, or change SCSI channel assignments.

Before using this command, you should consider the following points:

- The `controller pause_io` command allows a maximum delay of 150 seconds (that is, 2 1/2 minutes) for all I/O activity on the currently opened controller. This delay is based on an internal driver timeout (not on any network timeout) of three minutes.
- The network timeout is *client-based*, and different clients are likely to have different timeout values. Thus, pausing all I/O activity on the open controller can cause network timeouts, particularly if the specified delay is for more than one minute.
- If you find it necessary to pause the system for more than a minute, then network timeouts are usually preferable to rebooting.
- Taking the maximum delay of 2 1/2 minutes increases the chances that a network timeout will occur. If you cannot perform the hardware reconfiguration changes in less than 2 1/2 minutes, then you should shut down the system. Even if you take the maximum delay, you can usually resume the I/O sooner. This is the reason for making the pause I/O default time 2-1/2 minutes.
- The safest amount of time (that is, timeouts are unlikely to occur) to delay I/O operations is no more than 30 seconds.

Command and Switch Availability

This command is supported in Windows and NetWare.

Notes

The controller automatically performs a rescan of the SCSI channel before the I/O is resumed.

Using this command when there is a pagefile on an array could cause unexpected behavior.

Use of the `controller pause_io` command is not permitted while an array task is running on the controller. (Use the `task list` command to display a list of currently running array tasks.)

Syntax

```
controller pause_io [{integer}]
```

Parameters

`{integer}`

Specifies the amount of time, in seconds, to cause the controller to wait before automatically resuming I/O.

The parameter defaults to the maximum value of 150 seconds. This is necessary because the `controller resume_io` command may not be able to access the controller under some circumstances.

Examples

The following example pauses all I/O activity on the currently opened controller for 100 seconds:

```
AAC0> controller pause_io 100
Executing: controller pause_io 100
```

Related Commands

`controller` commands:

- `controller resume_io` (page 4-16)

`task` commands:

- `task list` (page 8-2)

controller rescan

To rescan the SCSI channels on the currently opened controller and update all underlying structures, use the `controller rescan` command. The effect of this command is to verify currently connected disks or to recognize new disks added to the channel.

This command rescans the disks for the arrays and loads the volatile array and the partition tables from disks again. If nothing on the disks has changed, `controller rescan` should have no visible effect. However, if any disk partition structures have changed, this command causes those changes to be reflected in the array and partition lists in the controller memory. If you add disks to the controller, the rescan enables you to use those disks.

Syntax

```
controller rescan
```

controller reset_scsi_channel

To reset a specific SCSI channel, use the `controller reset_scsi_channel` command.

Syntax

```
controller reset_scsi_channel {integer}
```

Parameters

`{integer}`

Specifies the controller channel (for example, 0, 1, 2, 3, etc.) on which you want to reset this SCSI channel. See the installation guide for your controller to determine the number of channels it actually supports.

Examples

The following example resets the SCSI channel on channel 1:

```
AAC0>controller reset_scsi_channel 1
Executing: controller reset_scsi_channel 1
```

controller resume_io

To rescan the SCSI channels and resume all I/O activity on the currently opened, previously paused controller, use the `controller resume_io` command. You use this command after pausing the controller with the `controller pause_io` command.

Syntax

```
controller resume_io
```

Examples

The following example resumes all I/O activity on the currently paused controller:

```
AAC0> controller resume_io
Executing: controller resume_io
```

Related Commands

`controller` commands:

- `controller pause_io` (page 4-12)

controller set automatic_failover

Automatic failover allows you to replace a failed disk with a replacement disk. The controller then automatically assigns the disk you insert as a failover disk without your having to first assign it with the `array set failover` or `array set global_failover` command.

Note that the automatic failover feature works only with disks that reside in a SAF-TE enclosure management device.

To turn on or off automatic failover for the specified controller, use the `controller set automatic_failover` command.

Caution: The controller deletes any data on the replacement disk when automatic failover is enabled and you remove the failed disk and insert the replacement disk in the failed disk's slot.

Notes

To assign one or more failover disks for a single array, you use the `array set failover` command. To assign one or more failover disks for all arrays, you use the `array set global_failover` command. Although all array types accept failover assignments, only mirror sets and RAID 5 arrays use the failover assignment if a disk fails. (Mirror sets and RAID 5 arrays are often referred to as redundant arrays). The main characteristic of these commands is that you must "manually" assign the failover disk to one or more arrays. In fact, you might consider the previously described failover mechanism as "manual failover".

The controller will assign a replacement drive as a failover disk only if the serial number of the drive is different from the serial number of the drive it replaces.

Syntax

```
controller set automatic_failover
[/failover_enabled{=boolean}]
/failover_enabled{=boolean}
```

Specifies whether to turn on or off automatic failover on the specified controller. You must set this switch to `TRUE` if you want to turn on automatic failover on the specified controller.

To turn off automatic failover on the specified controller, set this switch to `FALSE`. The command defaults to `TRUE`, which means the command turns on automatic failover on the specified controller.

Examples

Before turning the automatic failover feature on or off, check its status by using the `controller show automatic_failover` command.

The following example shows that the controller's automatic failover is currently disabled:

```
AAC0> controller show automatic_failover
Executing: controller show automatic_failover
Automatic failover DISABLED
```

The following example turns on (enables) the automatic failover feature for the currently opened controller:

```
AAC0> controller set automatic_failover /failover_enabled
Executing: controller set automatic_failover /failover_enabled=TRUE
```

The `controller show automatic_failover` command indicates that the controller's automatic failover is currently enabled:

```
AAC0> controller show automatic_failover
Executing: controller show automatic_failover
Automatic failover ENABLED
```

Related Commands

`array` commands:

- `array set failover` (page 3-95)
- `array set global_failover` (page 3-97)

`controller` commands:

- `controller show automatic_failover` (page 4-21)

controller set array_verify

To enable or disable the array verify operation, use the `controller set verify` command. This command allows you to enable or disable the array verify operation for all arrays controlled by the specified controller.

Syntax

```
controller set array_verify
[/verify_enabled{=boolean}]
/verify_enabled{=boolean}
```

Specifies whether to turn on or off the array verify feature on the specified controller. You must set this switch to `TRUE` if you want to turn on array verify for all arrays associated with the specified controller.

To turn off the array verify feature on the specified controller, set this switch to `FALSE`. The command defaults to `TRUE`, which means the command turns on the array verify feature on the specified controller.

Examples

Before turning the array verify feature on or off, check its status by using the `controller show array_verify` command. The following example shows that the controller's array verify feature is currently disabled:

```
AAC0> controller show array_verify
Executing: controller show array_verify
Array verify DISABLED
```

The following example turns on (enables) the array verify feature for the currently opened controller:

```
AAC0> controller set array_verify /verify_enabled
Executing: controller set array_verify /verify_enabled=TRUE
```

The `controller show array_verify` command indicates that the controller's array verify feature is currently enabled. The command also displays the number of errors (if any) found as a result of the array verify operation.

```
AAC0> controller show array_verify
Executing: controller show array_verify
Array verify ENABLED
Errors found 0
```

Related Commands

`controller` commands:

- `controller show array_verify` (page 4-25)

controller show automatic_failover

To display the automatic failover status (enabled or disabled) for the specified controller, use the `controller show automatic_failover` command.

Syntax

```
controller show automatic_failover
```

Examples

The following example shows that the controller's automatic failover is currently disabled:

```
AAC0> controller show automatic_failover
Executing: controller show automatic_failover
Automatic failover DISABLED
```

Related Commands

`array` commands:

- `array set failover` (page 3-95)
- `array set global_failover` (page 3-97)

`controller` commands:

- `controller set automatic_failover` (page 4-17)

controller show channels

To show all of the channels on a controller and the associated characteristics of each channel, use the `controller show channels` command. See the installation guide for your controller to determine the number of channels it actually supports.

Syntax

```
controller show channels
```

Examples

In the following example, the system displays the channels on the currently opened controller:

```
AAC0> controller show channels
Executing: controller show channels
Ch# Host ID Targets Type      Max Usage
--- ------- ------- --------- ---------
0   7       15      NoInfo    NoInfo
1   7       15      NoInfo    NoInfo
2   7       15      NoInfo    NoInfo
3   7       15      NoInfo    NoInfo
```

The following example displays the channels on the currently opened controller:

```
AAC0> controller show channels
Executing: controller show channels
Ch# Host ID Targets Type      Max Usage
--- ------- ------- --------- ---------
0   7       15      NoInfo    NoInfo
1   7       15      NoInfo    NoInfo
2   7       15      NoInfo    NoInfo
3   7       15      NoInfo    NoInfo
```

The following sections provide more information on the items that the `controller show channels` command displays.

The Ch# and Host ID Items

The Ch# item displays the number of the channel on the controller. In the example there are four channels on the currently opened controller. See the installation guide for your controller to determine the number of channels it actually supports.

The Host ID item displays the SCSI device ID of the host controller, which in the example is 7 for all channels.

The Targets and Type Items

The Targets item displays the number of SCSI device IDs not including the controller ID, which in the example is 15 for all channels.

The Type item displays the SCSI channel type the channel is capable of supporting, which in the example is NoInfo for all channels. The `controller show channels` command can display the SCSI channel types listed in Table 4-5.

Table 4-5 SCSI Channel Types

Type	Meaning
Fast	The controller supports SCSI channel type Fast. This channel type is not currently supported.
FastWide	The controller supports SCSI channel type FastWide. This channel type is not currently supported.
FibreChnl	The controller supports SCSI channel type Fibrechannel. This channel type is not currently supported.
NoInfo	No information is available on the SCSI channel type supported by this controller.
Slow	The controller supports SCSI channel type Slow. This channel type is not currently supported.
SlowWide	The controller supports SCSI channel type SlowWide. This channel type is not currently supported.
Ultra	The controller supports SCSI channel type Ultra. This channel type is not currently supported.
Ultra2LVD	The controller supports SCSI channel type Ultra2LVD.
UltraWide	The controller supports SCSI channel type UltraWide.
Unknown	The controller SCSI channel type that the controller supports is unknown.

The Max Usage Item

The Max Usage item displays the best speed the channel is running at, which in the example is NoInfo for all channels. If drives other than Ultra2 are present on the system, the entire channel is limited to UltraWide.

Related Commands

`controller` commands:

- `controller details` (page 4-2)

controller show array_verify

To display the array verify status (enabled or disabled) for the specified controller, use the `controller show array_verify` command.

Syntax

```
controller show array_verify
```

Examples

Before turning the array verify feature on or off, check its status by using the `controller show array_verify` command.

The following example shows that the controller's automatic failover is currently enabled. The command also displays the number of errors (if any) found as a result of the array verify operation.

```
AAC0> controller show array_verify
Executing: controller show array_verify
Array verify ENABLED
Errors found 0
```

Related Commands

`controller` commands:

- `controller set array_verify` (page 4-19)

5

disk Commands

In this Chapter

Use the following syntax for `disk` commands:

```
disk command [object] [/switch{=value}]
[parameter]
```

disk blink

To cause a SCSI disk access light to blink (or stop blinking), use the `disk blink` command.

Syntax

```
disk blink [/wait{=boolean}] {scsi_device}
{integer}
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI disk you want to blink. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

`{integer}`

Specifies the number of seconds you want the SCSI disk to blink. A value of zero (0) stops the SCSI disk from blinking.

Switches

`/wait{=boolean}`

Specifies whether to perform verification synchronously or asynchronously. If you set this switch to `TRUE`, the command performs the block verification synchronously and the command prompt does not return until the block verification operation completes.

The default is `FALSE`; the command performs the block verification asynchronously and the command prompt returns immediately.

Examples

The following example causes SCSI channel number 1, SCSI device ID 2, and SCSI device logical unit number 0 to blink for fifty-five seconds:

```
AAC0>disk blink (1,2,0) 55
Executing: disk blink  (CHANNEL=1,ID=2,LUN=0) 55
```

disk initialize

To initialize a SCSI disk for use with the currently opened controller, use the `disk initialize` command. This command writes data structures to the disk so that the controller can use the disk.

Syntax

```
disk initialize [/always{=boolean}]
[/unconditional{=boolean}]{scsi_device}
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI disk you want to prepare and initialize for controller use. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

Switches

`/always{=boolean}`

Specifies whether to initialize the disk even if the disk has data on it. If you specify `TRUE` for this switch, the command initializes the disk even if it has data on it. If arrays on the disk have open files, this command cannot be used. In other words, you need to close all open files before you use this command to initialize a SCSI disk.

The default behavior for the command is `FALSE`; that is, the `disk initialize` command does not initialize a disk that has data on it. Specifying `/always` overrides this behavior.

`/unconditional{=boolean}`

Specifies whether to initialize the disk even if arrays on the disk have open files. If you specify `TRUE` for this switch, the command initializes the disk even if the disk has open files.

The default is `FALSE`; that is, the `disk initialize` command does not initialize a disk that has open files on it.

WARNING: Unconditionally deleting an array that is in use can cause a system crash under some circumstances.

Examples

The following example initializes a SCSI disk device (SCSI channel number 0, SCSI device ID 2, and SCSI device logical unit number 0) that has data on it:

```
AAC0>disk initialize /always (0,2,0)

Executing: disk initialize /always=TRUE (CHANNEL=0,ID=2,LUN=0)
```

disk list

To display a list of the disks available on the currently opened controller, use the `disk list` command.

Syntax

```
disk list [/all{=boolean}] [/full{=boolean)]
[{scsi_device}]
```

Parameters

`{scsi_device}`

Specifies the ID for a specific SCSI device for which you want to display information. A SCSI device consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

Switches

`/all{=boolean}`

Specifies whether to display a list of all SCSI devices. If you set this switch to `TRUE`, the command displays a list of all SCSI devices on the controller.

This switch defaults to `FALSE` if you specify `scsi_device`. Otherwise, the switch defaults to `TRUE`.

`/full{=boolean}`

Specifies whether to display detailed information. If you specify `TRUE`, the command displays detailed information. If you specify `FALSE`, the command does not display detailed information.

This switch defaults to `FALSE`.

Examples

The following example shows how to display nondetailed information for all of the SCSI disks on the currently opened controller:

```
AAC0>disk list

Executing: disk list
C:ID:L  Device Type Blocks    Bytes/Block  Usage      Shared Rate
------  ----------  --------- ----------- ---------- ------ ----
2:00:0   Disk       8887200   512         Initialized   NO   40
2:01:0   Disk       8887200   512         Initialized   NO   40
2:02:0   Disk       8496960   512         Initialized   NO   40
2:03:0   Disk       8887200   512         Initialized   NO   40
```

The detailed `disk list` display contains the previous columns of information plus the following columns:

- Removable media
- Vendor-ID
- Product-ID
- Rev

The following sections provide more information on the columns of information that the `disk list` command displays.

The C:ID:L Column

The **C:ID:L** column displays the SCSI device channel number, the SCSI device ID, and the SCSI device logical unit number for each disk on the currently opened controller. (The display shows only one disk if you specified a particular disk.)

In the example, the command displays the SCSI channel number, the SCSI device ID, and the SCSI device logical unit number for the four disks on the open controller.

The Device Type Column

The **Device Type** column displays the SCSI device type. The `disk list` command can display one of the values listed in Table 5-1.

Table 5-1 Device Type Values

Value	Meaning
Disk	The device type is a SCSI direct access device. Typically, these are disk drives. This value appears in the example.
Sequential	The device type is a SCSI sequential access device; typically, tape drives.
Printer	The device type is a SCSI printer device.
Processor	The device type is a processor device.
Write once	The device type is a SCSI write once and read many times device. Typically, these devices are referred to as WORM disk drives.
CDROM	The device type is a SCSI CD-ROM (read-only direct access device).
Scanner	The device type is a SCSI scanner device.
Optical	The device type is a SCSI optical disk device.
Medium changer	The device type is a SCSI medium changer device; typically, a changer for a Jukebox.
Network	The device type is a SCSI network communication device.

The Removable media Column

The **Removable media** column displays the value **Y** if the media in the device is removable and the value **N** if the media in the device is not removable.

The Vendor-ID and Product-ID Columns

The **Vendor-ID** column displays a string of characters that identifies the vendor of the SCSI device.

The **Product-ID** column displays a string of characters that identifies the product line associated with the SCSI device.

The Rev Column

The **Rev** column displays the revision number of the SCSI device.

The Block Column

The **Block** column displays the number of blocks available on the SCSI device.

The Bytes/Block Column

The **Bytes/Block** column displays the number of bytes for each block on the SCSI device.

The Usage Column

The **Usage** column displays the usage of the SCSI device. The `disk list` command can display one of the values listed in Table 5-2.

Table 5-2 SCSI Disk Device Usage Values

Value	Meaning
Detached	The system detected that the diskset in the detached state. When a diskset is detached, it is not available for use.
DOS	The SCSI device was prepared for use (initialized) with MS-DOS partitions.
Initialized	The SCSI device was prepared for use (initialized) with arrays.
Not Initialized	The SCSI device was not prepared for use (initialized) with arrays.
Offline	The SCSI device was present at boot time. However, the device is either removed from the controller or it failed.
Unowned	The controller does not own the SCSI device.

The Shared Column

The **Shared** column displays the value **YES** if the device resides on a shared channel and the value **NO** if the device does not reside on a shared channel. In the example, the devices do not reside on a shared channel.

The Rate Column

The **Rate** column displays the negotiated speed of the SCSI device, in megabytes per second.

Related Commands

`disk` commands:

- `disk initialize` (page 5-3)
- `disk show partition` (page 5-19)
- `disk show space` (page 5-26)

disk remove dead_partitions

To remove all dead partitions from a SCSI disk, use the `disk remove dead_partitions` command. A dead partition is a partition that is no longer used by any array.

Typically, you use this command only under specific circumstances. For example, if you remove a disk from a redundant array and then later add the disk to the controller, the partition on the reinserted disk (which was previously part of the redundant array) is no longer useful. In this case, you use this command and specify the disk that was added to the controller again.

If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column (displayed with the `array list` command) changes to a "!" (exclamation point).

Syntax

```
disk remove dead_partitions {scsi_device}
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI disk from which you want to remove all dead partitions. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

Examples

The following example removes all dead partitions that reside on a SCSI disk device (SCSI channel number 0, SCSI device ID 2, and SCSI device logical unit number 0):

```
AAC0>disk remove dead_partitions (0,02,0)
Executing: disk remove dead_partitions (CHANNEL=0,ID=2,LUN=0)
```

Related Commands

`array` commands:

- `array list` (page 3-50)

disk set default

To set the default SCSI ID for use in subsequent CLI commands, use the `disk set default` command. This command allows you to set defaults for a specific SCSI device's SCSI device channel number and SCSI device logical unit number. Then, in subsequent commands, you need only specify the SCSI device ID on the command line.

Syntax

```
disk set default {scsi_device}
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI device. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), a SCSI device ID (0 through 15 inclusive), and a SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

If you do not specify a SCSI device ID, the command displays an appropriate error message.

Examples

The following example sets the SCSI channel number and SCSI default logical unit number for SCSI device (0,3,0) as the default for use in subsequent CLI commands:

```
AAC0>disk set default (1,3,0)
Executing: disk set default (CHANNEL=1,ID=3,LUN=0)
```

You need only specify the SCSI device ID in subsequent CLI commands, and the CLI assumes the default SCSI channel number and SCSI logical unit number as follows:

```
AAC0>disk list 3
Executing: disk list (ID=3)
C:ID:L  Device Type Blocks    Bytes/Block Usage          Shared
------  ----------- --------- ---------   -------------- ------
2:03:0   Disk       4197405     512       Initialized     NO
```

For further details, see *disk list* on page 5-5.

Related Commands

`disk` commands:

- `disk list` (page 5-5)
- `disk show default` (page 5-16)

disk set smart

To change a device's S.M.A.R.T. configuration, use the `disk set smart` command. The acronym S.M.A.R.T. stands for Self-Monitoring, Analysis and Reporting Technology. This technology is an industry standard for hard drives that monitors a variety of disk parameters, such as the rate of read-write errors. In addition, S.M.A.R.T. can send an alert to system administrators about potential problems caused by disk errors.

This command allows you to make the following changes related to S.M.A.R.T.:

- Enable or disable S.M.A.R.T. configuration for all disks on the system
- Clear the S.M.A.R.T. error count for the specified disk
- Enable or disable S.M.A.R.T. exception reporting

Syntax

```
disk set smart [/all{=boolean}][/clear{=boolean}]
[/enable_exceptions{=boolean}][/logerr{=boolean}]
[/perf{=boolean}][/report_count{=integer}]
[/update{=boolean}] [{scsi_device}]
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI disk on which you want to change S.M.A.R.T. configurations. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

You do not need to specify a SCSI ID if you use the `/all` switch.

Switches

`/all{=boolean}`

Specifies whether to enable S.M.A.R.T. configurations (enable exception reporting) for all disks on the system. If you specify `TRUE` for this switch, the command enables S.M.A.R.T. configurations for all disks on the system.

The default is `FALSE`; the `disk set smart` command does not enable S.M.A.R.T. configurations for all disks on the system. In this case, you would want to specify a SCSI ID to enable S.M.A.R.T. configurations for a specific disk on the system.

`/clear{=boolean}`

Specifies whether to clear S.M.A.R.T. error counts for the specified disk or disks. If you specify `TRUE` for this switch, the command clears error counts for the specified disk or disks.

The default is `FALSE`; that is, the `disk set smart` command does not clear S.M.A.R.T. error counts for disks on the system.

`/enable_exceptions{=boolean}`

Specifies whether to enable S.M.A.R.T. exceptions reporting for the specified disk or disks. If you specify `TRUE` for this switch, the command enables exception reporting for the specified disk or disks.

The default is `FALSE`; that is, the `disk set smart` command does not enable S.M.A.R.T. exception reporting for disks on the system.

`/interval_timer{=integer} <?>`

`/logerr{=boolean}`

Specifies whether to enable logging of S.M.A.R.T. exception reporting for the specified disk or disks. If you specify `TRUE` for this switch, the command enables logging of S.M.A.R.T. exception reporting for the specified disk or disks.

If you specify `FALSE` for this switch, the command disables logging of S.M.A.R.T. exception reporting for the specified disk or disks.

`/mrie={integer} <?>`

`/perf{=boolean}`

Specifies whether to report exceptions according to the MRIE mode taking into account performance. If you specify `TRUE` for this switch, the command performs exception reporting as long as performance is not an issue. If performance is an issue, the command does not report exceptions.

If you specify `FALSE` for this switch, exceptions are reported according to the MRIE mode regardless of performance issues.

`/report_count{=integer}`

Specifies the number of times an exception can be reported. The value 0 (zero) indicates that there is no limit to the number of times an exception can be reported.

`/update{=boolean}`

Specifies whether to update the number of device errors found on the specified SCSI device. If you specify `TRUE` for this switch, the command updates the number of device errors found on the specified SCSI device.

If you specify `FALSE` for this switch, the command does not update the number of device errors found on the specified SCSI device.

Examples

The following example enables S.M.A.R.T. configurations on one SCSI disk device (SCSI channel number 1, SCSI device ID 2, and SCSI device logical unit number 0):

```
AAC0>disk set smart (1,2,0)

Executing: disk set smart (CHANNEL=1,ID=2,LUN=0)
```

Related Commands

`disk` commands:

- `disk show smart` (page 5-22)

disk show default

To display the current default for the SCSI device ID, use the `disk show default` command. You previously set this default using the `disk set default` command.

Syntax

```
disk show default
```

Examples

The following example shows the default for a SCSI device ID that was set in a previous use of the `disk set default` command:

```
AAC0>disk show default

Executing: disk show default
Default Scsi: (CHANNEL=0,ID=3,LUN=0)
```

If no default for a SCSI device was previously specified, the command displays an appropriate message.

Related Commands

`disk` commands:

- `disk set default` (page 5-11)

disk show defects

To show the number of defects and/or the defect list for a specific SCSI disk drive, use the `disk show defects` command.

Syntax

```
disk show defects [/full{=boolean}]{scsi_device}
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI device for which you want to display defect information. A SCSI device consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

Switches

`/full{=boolean}`

Specifies whether to display the defect count and the list of disk defects. If you set this switch to `TRUE`, the command displays the list of defects and the number of defects.

This switch defaults to `FALSE`, which means the command displays only the number of defects.

Examples

The following example lists the number of defects and the list of defects for a SCSI disk device (SCSI channel number 1, SCSI device ID 0, and SCSI logical unit number 0):

```
AAC0> disk show defects /full=TRUE (1,0,0)
Number of PRIMARY defects on drive: 2
Defect 1 at cylinder 12, head 5, sector 59
Defect 2 at cylinder 550, head 1, sector 44
Number of GROWN defects on drive: 0
```

The items in the display are described as follows:

- Number of PRIMARY defects on drive:

  This item shows the details regarding any primary disk defects. In the example, there are two defects.

- Defect 1 at cylinder 12, head 5, sector 59

  This item shows where defect 1 occurred. In the example, the defect occurred on head 5 of cylinder 12 located in sector 59.

- Defect 2 at cylinder 550, head 1, sector 44

  This item shows where defect 2 occurred. In the example, the defect occurred on head 1 of cylinder 550 located in sector 44.

  The command shows any additional defects.

- Number of GROWN defects on drive: 0

  This item shows the list of grown defects (if any). In the example, there are no grown defects, so the command displays the value 0 (zero).

disk show partition

To display a list of partitions on the disks attached to the currently opened controller, use the `disk show partition` command.

Syntax

```
disk show partition
```

Examples

The following example shows a sample output from the `disk show partition` command:

```
AAC0>disk show partition
Executing: disk show partition
Scsi   Partition     Array     MultiLevel
C:ID:L Offset:Size   Num Type   Num Type   R/W
------ ------------- --- ------ --- ------ ---
0:02:0 64.0KB:20.0MB 0   Volume 0   None   RW
```

The following sections provide more information on the columns that the `disk show partition details` command displays.

The SCSI C:ID:L Columns

The **C:ID:L** column displays the SCSI channel number, the SCSI device ID, and the SCSI device logical unit number for each disk on the currently opened controller.

In the example, the command displays the SCSI channel number, the SCSI device ID, and the SCSI device logical unit number for one disk (0:02:0) on the currently opened controller.

The Partition Offset:Size Column

The **Partition Offset:Size** column displays the offset (in bytes) into the SCSI device and the size of the partition (in bytes). In the example, the offset is **64.0 KB** and the size is **20.0 MB**.

If a partition is dead, the ":" (colon) in the **Partition Offset:Size** column changes to a "!" (exclamation point). See the `disk remove dead_partitions` (page 5-10) command for more information on dead partitions.

The Array Num and Array Type Columns

The **Array Num** column displays the ID of the primary array in the partition. In the example, the array ID is **0**.

The **Array Type** column displays the type of array in which the partition resides. Table 5-3 lists the possible values that the command can display in the Type column.

Table 5-3 Primary Array Values

Value	Meaning
None	This partition is not in an array.
Mirror	This partition is part of a mirror set.
Stripe	This partition is part of a stripe set.
Volume	This partition is part of a volume set.
RAID-5	This partition is part of a RAID 5 array.
Reconf	This partition is part of an array reconfiguration operation. An array reconfiguration operation occurs when you use the `array reconfigure` command to change an array from one type to another. The Reconf value is not used in UNIX.

The MultiLevel Num and MultiLevel Type Columns

The **MultiLevel Num** column displays the ID of the multilevel array the partition is associated with (if any). In the example, the array ID is **0**.

The **MultiLevel Type** column displays the type of multilevel array in which the partition resides. Table 5-4 lists the possible values that the command can display in the Type column.

Table 5-4 Multilevel Array Values

Value	Meaning
None	This partition is not in a secondary array.
Stripe	This partition is part of a stripe set.
Volume	This partition is part of a volume set.

The R/W Column

The **R/W** column displays whether the partition is read-only or read-write. Specifically, the column can display the values listed in Table 5-5.

Table 5-5 R/W Array Values

Value	Meaning
RO	The partition is read-only.
RW	The partition is read-write.

disk show smart

To display S.M.A.R.T. configuration information for one or all disks, use the `disk show smart` command. The acronym S.M.A.R.T. stands for Self-Monitoring, Analysis and Reporting Technology. This technology is an industry standard for hard drives that monitors a variety of disk parameters, such as the rate of read-write errors. In addition, S.M.A.R.T. can send an alert to system administrators about potential problems caused by disk errors.

Syntax

```
disk show smart [/all{=boolean}] [/
full{=boolean)] [/
view_changeable{=boolean}][{scsi_device}]
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI disk for which you want to display S.M.A.R.T. configuration information. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

You do not need to specify a SCSI ID if you use the `/all` switch.

Switches

`/all{=boolean}`

Specifies whether to display S.M.A.R.T. configuration information for all disks on the system. If you specify `TRUE` for this switch, the command displays S.M.A.R.T. configuration information for all disks on the system.

This switch defaults to `TRUE`.

`/full{=boolean}`

Specifies whether to display detailed S.M.A.R.T. configuration information for one or all SCSI disks. If you specify `TRUE`, the command displays detailed S.M.A.R.T. configuration information. If you specify `FALSE`, the command does not display detailed S.M.A.R.T. configuration information.

This switch defaults to `FALSE`.

`/view_changeable{=boolean}`

Specifies whether to display the configuration information that can be set on a S.M.A.R.T. disk. If you specify `TRUE` for this switch, the command displays the letter "X" in the column for the configuration information that can be set. For example, an "X" appears in the Enable Exception Control column for those S.M.A.R.T. disks on which this value can be set.

This switch defaults to `FALSE`.

Examples

The following example shows how to display nondetailed S.M.A.R.T. configuration information for all disks on the system:

```
AAC0>disk show smart
Executing: disk show smart
        Smart    Method of          Enable
        Capable  Informational      Exception  Performance  Error
C:ID:L  Device   Exceptions(MRIE)   Control    Enabled      Count
------  -------  ----------------   ---------  -----------  ----
3:00:0     N
3:01:0     N
3:02:0     N
3:03:0     N
```

The detailed `disk show smart` display contains the previous columns of information plus the following columns:

- Log Errors
- Interval Timer (secs.)
- Report Count

The following sections provide more information on the columns that the `disk show smart` command displays.

The C:ID:L Column

The **C:ID:L** column displays the SCSI channel number, the SCSI device ID, and the SCSI device logical unit number for each disk on the currently opened controller. (The display shows only one disk if you specified a particular disk.)

In the example, the command displays the SCSI channel number, the SCSI device ID, and the SCSI device logical unit number for the four disks on the open controller.

The Smart Capable Device Column

The **Smart Capable Device** column displays the value **Y** if the device is S.M.A.R.T. capable or the value **N** if the device is not S.M.A.R.T. capable.

The Method of Informational Exceptions (MRIE) Column <?>

The **Method of Informational Exceptions (MRIE)** column displays the MRIE value (one of a possible six values). However, you cannot set the MRIE. Your devices report exceptions only when polled (level 6) and you cannot change the setting for this switch.

The Enable Exception Control Column

The **Enable Exception Control** column displays the value **Y** if you enabled S.M.A.R.T. exception control reporting or the value **N** if you did not enable S.M.A.R.T. exception control reporting.

If you specified the `/view_changeable` switch, the command displays an **X** in this column for those disks on which you can enable S.M.A.R.T. exception control reporting.

The Performance Enabled Column

The **Performance Enabled** column displays the value **Y** if you enabled performance or **N** if you did not enable performance. You enable performance by using the `disk set smart` command's `/perf` switch.

If you specified the `/view_changeable` switch, the command displays an **X** in this column for those disks on which you can enable performance.

The Log Errors Column

The **Log Errors** column displays the value **Y** if you enabled log errors or **N** if you did not enable log errors. You enable log errors by using the `disk set smart` command's `/logerr` switch.

If you specified the `/view_changeable` switch, the command displays an **X** in this column for those disks on which you can enable log errors.

The Interval Timer (secs.) Column

The **Interval Timer (secs.)** column displays the number of seconds specified for the exception reporting interval. If you specified the `/view_changeable` switch, the command displays an **X** in this column for those disks on which you can set the number of seconds for the exception reporting interval.

The Report Count Column

The **Report Count** column displays the number of times you specified for exceptions to be reported. You specify this number by using the `disk set smart` command's `/report_count` switch.

The Error Count Column

The **Error Count** column displays the number of errors that S.M.A.R.T. encountered on the disk.

If you specified the `/view_changeable` switch, the command displays an **X** in this column for those disks on which you can enable the recording of errors that S.M.A.R.T. encounters on the disk.

Related Commands

`disk` commands:

- `disk set smart` (page 5-13)

disk show space

To display space usage information on a SCSI disk, use the `disk show space` command.

Syntax

```
disk show space [/all{=boolean}] [{scsi_device}]
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI device for which you want to display space usage information. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), a SCSI device ID (0 through 15 inclusive), and a SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

Switches

`/all{=boolean}`

Specifies whether to show space usage information for all SCSI disks on the currently opened controller. If you set this switch to `TRUE`, the command shows space usage information for all SCSI disks on the currently opened controller.

This switch defaults to `FALSE` if you specify the `scsi_device` parameter; otherwise, the switch defaults to `TRUE`.

Examples

The following example shows space usage information for all of the SCSI disks on the currently opened controller:

```
AAC0>disk show space
Executing: disk show space
Scsi C:ID:L Usage       Size
----------- ---------   -------------
  0:02:0    Array       64.0KB:  20.0MB
  0:02:0    Free        20.0MB:  29.0MB
  0:03:0    Free        64.0KB:  49.0MB
  0:04:0    Free        64.0KB:  49.0MB
  0:05:0    Free        64.0KB:  49.0MB
```

The following example shows space usage information for SCSI disk device (0,2,0):

```
AAC0>disk show space (0,2,0)
Executing: disk show space (ID=2)
Scsi C:ID:L Usage       Size
----------- ---------   -------------
  0:02:0     Array      64.0KB:  20.0MB
  0:02:0     Free       20.0MB:  29.0MB
```

The following sections provide more information on the columns that the `disk show space` command displays.

The Scsi C:ID:L Column

The **Scsi C:ID:L** column displays the SCSI channel number, the SCSI device ID, and the SCSI device logical unit number for each disk on the currently opened controller. (The display shows only one disk if you specified a particular disk.)

In the first example, the command displays the SCSI channel number, the SCSI device ID, and the SCSI device logical unit number for the six disks on the open controller. In the second example, the command displays information for the specified SCSI device (0:02:0).

The Usage Column

The **Usage** column displays the type of partition or some characteristic related to disk space usage. Specifically, the command can display the values listed in Table 5-6.

Table 5-6 Disk Usage Values

Value	Meaning
Free	This area is unused space on the disk. In the example, a specific area is unused space on the 0:02:0 disk. This unused space is typically referred to as freespace.
Array	This partition is used as part of an array. In the example, a partition on the 0:02:0 disk is used as part of an array.
Orphan	This area was part of an array that could not be configured.
Dead	This area had an error and was declared dead.
Conflict	This area is not configured because of a conflict.

The Size Column

The **Size** column displays the offset (in bytes) into the SCSI device and the size of the partition or space (in bytes). In the example, the offset and sizes for the first two disks are as follows:

- For the 0:02:0 disk

  The offset is **64.0 KB** for the partition and **20.0 MB** for the unused space. The size is **20.0 MB** for the partition and **29.0 MB** for the unused space.

- For the 0:03:0 disk

  The offset is **64.0 KB** and the size is **49.0 MB** for the unused space. (This disk has no partitions on it.)

  The offset is **64.0 KB** and the size is **49.0 MB** for the unused space. (This disk has no partitions on it.)

disk verify

To verify all blocks on a SCSI disk device and, optionally, repair any bad blocks, use the `disk verify` command.

Syntax

```
disk verify [/repair{=boolean}] [/wait{=boolean}]
{scsi_device}
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI disk device that you want to verify. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), a SCSI device ID (0 through 15 inclusive), and a SCSI logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

Switches

`/repair{=boolean}`

Specifies whether to automatically repair bad blocks. If you set this switch to `TRUE`, the command attempts to repair any bad blocks. If you do not specify the switch, the command only reports failures.

This switch defaults to `FALSE`.

`/wait{=boolean}`

Specifies whether to perform verification synchronously or asynchronously. If you set this switch to `TRUE`, the command performs the block verification synchronously and the command prompt does not return until the block verification operation completes.

The default is `FALSE;` that is, the command performs the block verification asynchronously and the command prompt returns immediately.

Examples

The following example synchronously verifies and repairs all blocks on disk (0,2,0):

```
AAC0>disk verify /repair=TRUE /wait=TRUE (0,2,0)
Executing: disk verify (CHANNEL=0,ID=2,LUN=0)
```

As the command executes, note the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!Task:100,Func:SCV,State:RUN 97.2%
```

For further details on status information, see page 1-13.

disk zero

To clear an entire SCSI disk, use the `disk zero` command. When you clear a disk, all data is erased and cannot be recovered.

Command and Switch Availability

This command is supported on Windows.

Syntax

```
disk zero [/always{=boolean}] [/wait{=boolean}]
{scsi_device}
```

Parameters

`{scsi_device}`

Specifies the ID for the SCSI disk you want to clear. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive).

For further details, see *scsi_device* on page 1-10.

Switches

`/always{=boolean}`

Specifies whether to clear the disk, even if it has data on it. If you specify `TRUE`, the command clears the disk even if it has data on it.

The default behavior for the command is `FALSE`; the command clears the disk only if it has no data on it. Specifying `/always` overrides this behavior.

In both cases, all user files must be closed. The `/always` switch cannot override this restriction.

`/wait{=boolean}`

Specifies whether to clear the disk synchronously or asynchronously. If you set this switch to `TRUE`, the command clears the disk synchronously and the command prompt does not return until the clear disk task completes.

The default is `FALSE`; the command clears the disk asynchronously and the command prompt returns immediately.

Examples

The following example clears SCSI disk (0,2,0). There is data on the disk. However, the data is not deemed important and thus the `/always` switch is used:

```
AAC0> disk zero /always /wait (0,2,0)
Executing: disk zero /always=TRUE /wait=TRUE (CHANNEL=0,ID=2,LUN=0)
```

As the command executes, note the title bar of the DOS window displays the status of the command. For example:

```
Stat:OK!Task:100,Func:SCZ,State:RUN 97.2%
```

For further details on status information, see page 1-13.

6

diagnostic Commands

In this Chapter

The `diagnostic` commands are used to help fix problems that can occur during controller operation. The `diagnostic` commands should be used only under the direction of technical support.

Use the following syntax for `diagnostic` commands:

```
diagnostic command [/switch{=value}] [parameter]
```

diagnostic clear boot_parameters

To clear all boot-time parameters contained in the currently opened controller's parameters, use the `diagnostic clear boot_parameters` command. Under normal conditions, no boot-time parameters are set.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic clear boot_parameters
```

Examples

The following example clears all boot-time parameters:

```
AAC0> diagnostic clear boot_parameters
Executing: diagnostic clear boot_parameters
All boot time parameters cleared to default values.
```

Related Commands

`diagnostic` commands:

- `diagnostic set boot_parameter` (page 6-11)
- `diagnostic show boot_parameter` (page 6-12)

diagnostic dump structures

To save internal data structures to a file for use by technical support, use the `diagnostic dump structures` command.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic dump structures
```

Parameters

`{string}`

Specifies the name of the file to contain the internal data structures. Use the filename conventions that apply to your operating system.

Examples

The following example saves the internal data structures to a file called `c:\aac0loginfo\dump.txt` on a Windows system:

```
AAC0>diagnostic dump structures c:\aac0loginfo\dump.txt
Executing: diagnostic dump structures "c:\aac0loginfo\dump.txt"
```

Related Commands

`diagnostic` commands:

- `diagnostic dump text` (page 6-4)

`logfile` commands:

- `logfile start` (page 7-3)

diagnostic dump text

To display diagnostic information on the console display for use by technical support during bug reporting, use the `diagnostic dump text` command.

Caution: Use this command only under the direction of technical support.

Notes

Before using this command in DOS, verify that the window is set to scroll data. For details, refer to your DOS documentation.

Before using this command on other operating systems, you might want to make sure your windowing system is set up to scroll data. See your operating system documentation.

You might also want to use the `logfile start` command to make sure the diagnostic-related information gets sent to the logfile.

Syntax

```
diagnostic dump text
```

Examples

The following example displays diagnostic information on the console display:

```
AAC0>diagnostic dump text
Executing: diagnostic dump text
Partitions:6
```

The command displays a variety of information in table format. It also displays:

```
 *** HISTORY BUFFER FROM LAST RUN ***
[00]:
 *** HISTORY BUFFER FROM CURRENT CONTROLLER RUN ***
[00]:
==============
Dump Complete.
```

Related Commands

`diagnostic` commands:

- `diagnostic dump structures` (page 6-3)

diagnostic load_arrays

To load arrays when the system is in maintenance mode, use the `diagnostic load_arrays` command. This command differs from `controller rescan` in that it does not rescan the SCSI channel.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic load_arrays
```

Examples

The following example allows arrays to be loaded:

```
AAC0>diagnostic load_arrays

Executing: diagnostic load_arrays
All boot time parameters cleared to default values.
```

Related Commands

`controller` commands:

- `controller rescan` (page 4-14)

diagnostic moderation set count

To set the default interrupt count on the controller, use the `diagnostic moderation set count` command.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic moderation set count {integer}
```

Parameters

`{integer}`

Specifies the value (for example, 700) you want to set as the default interrupt count on the controller. The interrupt count is the number of outstanding requests on the controller needed before the delay time (the time specified by the `diagnostic set timer` command) takes effect. For example, if the interrupt count is set to 32 and the delay time is set to 10, the controller batches I/O request responses for one millisecond only if there are more than 32 I/Os outstanding in the one millisecond interval.

Related Commands

`diagnostic` commands:

- `diagnostic moderation set timer` (page 6-8)
- `diagnostic moderation show count` (page 6-9)
- `diagnostic moderation show timer` (page 6-10)

diagnostic moderation set timer

To set the default interrupt timer on the controller, use the `diagnostic moderation set timer` command.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic moderation set timer {integer}
```

Parameters

`{integer}`

Specifies the value (for example, 10) you want to set as the default interrupt timer on the controller. The interrupt timer is the delay time in 100 microsecond units before the controller issues an interrupt to the host computer. For example if the interrupt timer is set to 10, the controller batches I/O request responses for one millisecond and issues an interrupt to the host computer.

The default interrupt timer is 0 (zero).

Related Commands

`diagnostic` commands:

- `diagnostic moderation set count` (page 6-7)
- `diagnostic moderation show count` (page 6-9)
- `diagnostic moderation show timer` (page 6-10)

diagnostic moderation show count

To display the number of outstanding I/Os necessary to allow the delay of I/O request responses to the host computer, use the `diagnostic moderation show count` command. You may have previously specified the default interrupt count (the number of outstanding I/Os) with the `diagnostic moderation set count` command.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic moderation show count
```

Related Commands

`diagnostic` commands:

- `diagnostic moderation set count` (page 6-7)
- `diagnostic moderation set timer` (page 6-8)
- `diagnostic moderation show timer` (page 6-10)

diagnostic moderation show timer

To display the time in 100 microsecond units that the controller batches I/O request responses before issuing an interrupt to the host computer, use the `diagnostic moderation show timer` command. You may have previously specified the default interrupt timer (the time in 100 microsecond units) with the `diagnostic moderation set timer` command.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic moderation show timer
```

Related Commands

`diagnostic` commands:

- `diagnostic moderation set count` (page 6-7)
- `diagnostic moderation set timer` (page 6-8)
- `diagnostic moderation show count` (page 6-9)

diagnostic set boot_parameter

To set boot-time parameters that a technical support representative might need, use the `diagnostic set boot_parameter` command.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic set boot_parameter
```

Parameters

`{string}`

Specifies the name of the boot-time parameter to be set. If you need to use this command, a technical support representative will tell you the name of the boot-time parameter to use.

`{integer}`

Specifies the value for the boot-time parameter specified in the `string` parameter. If you need to use this command, a technical support representative will tell you the value to use.

Related Commands

`diagnostic` commands:

- `diagnostic clear boot_parameters` (page 6-2)
- `diagnostic show boot_parameter` (page 6-12)

diagnostic show boot_parameter

To display a specific boot-time parameter (if one exists) that a technical support representative might need, use the `diagnostic show boot_paramter` command.

Caution: Use this command only under the direction of technical support.

Syntax

```
diagnostic show boot_parameter
```

Parameters

`{string}`

Specifies the boot-time parameter character string whose associated value you want to display. If you need to use this command, a technical support representative will tell you the name of the boot-time parameter to use.

Related Commands

`diagnostic` commands:

- `diagnostic clear boot_parameters` (page 6-2)
- `diagnostic set boot_parameter` (page 6-11)

diagnostic show history

To display an internal history log of operations kept in the currently opened controller that a technical support representative might need, use the `diagnostic show history` command.

Syntax

```
diagnostic show history [/old{=boolean}]
[/current{=boolean}]
```

Switches

`/old{=boolean}`

Specifies whether to display the controller's operations history log from the previous boot of the operating system. If you set this switch to `TRUE`, the command displays the history log from the previous boot of the operating system.

If you do not specify any switches, the command displays the history log from the previous boot of the operating system (in other words this switch defaults to `TRUE)` .

If you set this switch and the `/current` switch to `FALSE`, the command displays an appropriate error message and stops executing.

`/current{=boolean}`

Specifies whether to display the controller's operations history log from the currently running system. If you set this switch to `TRUE`, the command displays the log from the currently running system.

This switch defaults to `FALSE`.

7

logfile Commands

In this Chapter

Use the following syntax for `logfile` commands:

```
logfile command [/switch{=value}] [parameter]
```

logfile end

To end the logging of all output, use the `logfile end` command.

Syntax

```
logfile end
```

Examples

The following example stops logging information to the file `aac0log17Nov01.txt` (which was previously started with the `logfile start` command):

```
AAC0> logfile end
Executing: logfile end


Log file closed.
```

Related Commands

`logfile` commands:

- `logfile start` (page 7-3)

logfile start

To begin the logging of all CLI command line activity to a specified file, use the `logfile start` command. CLI command line activity includes the associated output the CLI command displays.

The CLI continues to log command output until you close the CLI or you explicitly end the logfile with the `logfile end` command.

Syntax

```
logfile start [/append{=boolean}] {string}
```

Parameter

`{string}`

Specifies the name of the file to contain CLI command line activity. Use the filename conventions that apply to your operating system.

Switches

`/append{=boolean}`

Specifies whether to append all CLI command line activity to an existing output file, if one exists. If you set this switch to `TRUE`, the command appends the CLI command line activity to the file specified in `{string}`.

This switch defaults to `FALSE` (that is, the command does not append CLI command line activity to an existing output file).

Examples

The following Windows example logs all output to a file called `c:\ctrloginfo\aac0log17Nov01.txt`.

```
AAC0> logfile start c:\ctrloginfo\aac0log17Nov01.txt
Executing: logfile start "c:\ctrloginfo\aac0log17Nov01.txt"
File c:\ctrloginfo\aac0log17Nov01.txt receiving all output.
```

The following Windows example logs subsequent output to an existing file:

```
AAC0> logfile start /append c:\ctrloginfo\aac0log17Nov01.txt
Executing: logfile start /append=TRUE
"c:\ctrloginfo\aac0log17Nov01.txt"

File c:\ctrloginfo\aac0log17Nov01.txt receiving all output.
```

Related Commands

`logfile` commands:

- `logfile end` (page 7-2)

8

task Commands

In this Chapter

Use the following syntax for `task` commands:

```
task command [/switch{=value}] [parameter]
```

task list

To display a list of one or all tasks running on the currently opened controller, use the `task list` command. A task is an operation that occurs only on the controller, asynchronous to all other operations. Clearing a disk, creating a file system, and creating a mirror set are examples of tasks done on the controller.

The command displays an appropriate message if there are no tasks running on the controller.

Syntax

```
task list [/all{=boolean}]
```

Parameters

`{integer}`

Specifies the task ID for the task whose information you want to display. You need to perform a `task list /all` to display all tasks and their associated IDs.

Switches

`/all{=boolean}`

Specifies whether to display all currently running tasks. If you set this switch to `TRUE`, the command displays a list of all tasks running on the currently opened controller.

This switch defaults to `FALSE` if you specify the `integer` parameter; otherwise, the switch defaults to `TRUE`.

Examples

The following example shows typical output as a result of using the `task list` command.

```
AAC0> task list
Executing: task list
Controller Tasks
TaskId  Function  Done%  Array      State  Specific1 Specific2
------  --------  ------ ---------  -----  ---------  --------
101     Scrub     24.6%  0          SUS    00000000   00000000
```

As the example shows, the typical display shows the following columns of information:

- TaskId
- Function
- Done%
- Array
- State
- Specific1
- Specific2

The following sections describe each column that the `task list` command displays.

The Task ID Column

The **Task ID** column displays the ID number associated with a specific task. The controller assigns each task a unique ID number.

The Function Column

The **Function** column displays the type of task running on the controller. Table 8-1 describes the types of tasks that the `task list` command can display.

Table 8-1 Task Types

Task Type	Meaning
Create	A create mirror set or create multilevel mirror set task is running on the specified array. When the create mirror set or create multilevel mirror set task completes, the specified array is a mirror set or a multilevel array of mirror sets. The create a mirror set or create a multilevel mirror set task runs as a result of using the `array create mirror` or `array create mmirror` command.
FATfmt	An array format FAT file system task is running on the specified array. The format FAT file system task runs as a result of using the `array format` command with the `/file_system` switch set to `FAT`.

Table 8-1 Task Types (Continued)

Task Type	Meaning
NTFSfmt	An array format NTFS file system task is running on the specified array. The format NTFS file system task runs as a result of using the `array format` command with the `/file_system` switch set to `NTFS`.
Rebuild	A rebuild task is running on the specified array. Typically, the rebuild task runs when the controller is in a rebuild of a redundant array (RAID 5 array, mirror set, or multilevel array of mirror sets).
Reconfg	An array reconfigure task is running on the specified array. When the array reconfigure task completes, the specified array becomes a different array (for example, from a volume set to a stripe set). The array reconfigure task runs as a result of using the `array reconfigure` command.
Scrub	A scrub task is running on the specified redundant array. When the scrub task completes, the specified redundant array has reconstructed data on one partition based on data found on the other partition (for mirror sets and multilevel arrays of mirror sets). Or, the specified redundant array recalculates and replaces, if necessary, the parity information (for RAID 5 arrays). The scrub task runs as a result of the `array scrub` command. The scrub task also runs as a result of the `array create raid5` command with the `/scrub` switch specified. This is the task type displayed in the example.
Unknown	An unknown task is running.

Table 8-1 Task Types (Continued)

Task Type	Meaning
Verify	A verify with no repair of bad blocks task is running on the specified SCSI disk. When the verify with no repair of bad blocks task completes, the specified SCSI disk's blocks were verified without repairing any detected defects. The verify with no repair of bad blocks task runs as a result of using the `disk verify` command without specifying the `/repair` switch. Or, a verify with repair of bad blocks task is running on the specified SCSI disk. When the verify with repair of bad blocks task completes, the specified SCSI disk's blocks were verified and any detected bad blocks were repaired. The verify with repair of bad blocks task runs as a result of using the `disk verify` command with the `/repair` switch.
Zero	A clear disk task is running on the specified SCSI disk. When the clear disk task completes, the specified SCSI disk is cleared (that is, all data is erased and cannot be recovered). The clear disk task runs as a result of using the `disk zero` command.

The Done% Column

The **Done%** column displays a running percentage value that indicates the progress of the currently running task. The display shows the percentage in tenths of a percent increments. The currently running task is complete when the **Done %** column displays **100%**.

The Array Column

The **Array** column displays the ID number of the array associated with the task (if any). In the example, the array's ID is **0** (zero).

The State Column

The **State** Column displays the state of the task. Table 8-2 describes the state values that the `task list` command can display.

Table 8-2 State Values

State	Meaning
`BAD`	The task failed and is no longer running.
`DNE`	The task successfully completed or the task is no longer running (that is, the task is done).
`RUN`	The task is running.
`SUS`	The task was suspended. Typically, you suspend a task with the `task suspend` command.
`UNK`	The controller reported an unknown status for the task.

The Specific1 Column

The **Specific1** column displays additional information for certain tasks that appear in the **Task Id** column. Table 8-3 lists the tasks for which the `task list` command can display additional information. Table 8-3 also describes the additional information that `task list` can display.

Table 8-3 Additional Specific1 Information

Task	Additional Information
`Verify`. Refers to the verify with no repair of bad blocks task, which runs as a result of using the `disk verify` command without the `/repair` switch.	Displays the number of bad blocks found with no replacement. The example displays zeroes, which signifies no errors found.
`Scrub`. Refers to the scrub task, which runs as a result of using the `array scrub` command.	Displays the number of sectors that were different.

The Specific2 Column

The **Specific2** column displays additional information for certain tasks that appear in the **Task Id** column. Table 8-4 lists the tasks for which the `task list` command can display additional information. Table 8-4 also describes the additional information that `task list` can display.

Table 8-4 Additional Specific2 Information

Task	Additional Information
`Verify`. Refers to the verify with repair of bad blocks task, which runs as a result of using the `disk verify` command with the `/repair` switch.	Displays the number of bad blocks found that were replaced.

Related Commands

`array` commands:

- `array create mirror` (page 3-5)
- `array scrub` (page 3-88)

`disk` commands:

- `disk verify` (page 5-29)
- `disk zero` (page 5-31)

`task` commands:

- `task resume` (page 8-8)
- `task stop` (page 8-10)
- `task suspend` (page 8-12)

task resume

To resume one or all tasks previously suspended on the currently opened controller, use the `task resume` command. You suspend tasks with the `task suspend` command.

Syntax

```
task resume [/all{=boolean}] [{integer}]
```

Parameters

`{integer}`

Specifies the task ID for the task to resume. Perform a `task list /all` to display the task IDs for all tasks currently suspended on the open controller before resuming a specific task or all tasks.

Switches

`/all{=boolean}`

Specifies whether to resume all suspended tasks. If you set this switch to `TRUE`, the command resumes all tasks that were suspended on the currently opened controller.

If you do not specify this switch or a valid task ID, the command displays an appropriate error message.

Examples

Before resuming a suspended task, use the `task list` command to identify any suspended tasks, as in the following example.

```
AAC0>task list
Executing: task list
TaskId  Function  Done% Array      State  Specific1  Specific2
------  --------  ----- ---------  -----  ---------  ---------
101     Scrub     24.6% 0          SUS    00000000   00000000
```

The following example shows the use of the `task resume` command to resume task 101, which was previously stopped with the `task suspend` command:

```
AAC0>task resume 101
Executing: task resume 101
```

Use the `task list` command after using the `task resume` command to display information about the task, as in the following example:

```
AAC0>task list
Executing: task list
TaskId  Function  Done% Array      State  Specific1  Specific2
------  --------  ----- ---------  -----  ---------  ---------
101     Scrub     24.6% 0          RUN    00000000   00000000
```

As a result of using the `task resume` command, the State column now displays `RUN` (running on the controller) instead of `SUS` (suspended on the controller).

Related Commands

`task` commands:

- `task list` (page 8-2)
- `task suspend` (page 8-12)

task stop

To stop one or all tasks running on the currently opened controller, use the `task stop` command.

Syntax

```
task stop [/all{=boolean}]
```

Parameters

`{integer}`

Specifies the task ID for the task to stop. Perform a `task list /all` to display the task IDs for all tasks currently running on the open controller before stopping a specific task or all tasks.

Switches

`/all{=boolean}`

Specifies whether to stop all running tasks. If you set this switch to `TRUE`, the command stops all tasks that were running on the currently opened controller.

If you do not specify this switch or a valid task ID, the command displays an appropriate error message.

Examples

Before stopping a running task, use the `task list` command to identify any currently running tasks, as in the following example:

```
AAC0>task list
Executing: task list
TaskId  Function  Done% Array      State  Specific1  Specific2
------  --------  ----- ---------  -----  ---------  ---------
101     Scrub     24.6%  0         RUN    00000000   00000000
```

The following example shows the use of the `task stop` command to stop task 101:

```
AAC0> task stop 101
Executing: task stop 101
```

Use the `task list` command after using the `task stop` command to display information about the task, as in the following example:

```
AAC0>task list
Executing: task list
No tasks currently running on controller.
```

Related Commands

`task` commands:

- `task list` (page 8-2)

task suspend

To suspend one or all tasks running on the currently opened controller, use the `task suspend` command.

Syntax

```
task suspend [/all{=boolean}]
```

Parameters

`{integer}`

Specifies the task ID for the task to suspend. Perform a `task list /all` to display the task IDs for all tasks currently running on the open controller before suspending a specific task or all tasks.

Switches

`/all{=boolean}`

Specifies whether to suspend all running tasks. If you set this switch to `TRUE`, the command suspends all tasks that were running on the currently opened controller.

If you do not specify this switch or a valid task ID, the command displays an appropriate error message.

Examples

Before suspending a running task, use the `task list` command to identify any currently running tasks, as in the following example:

```
AAC0>task list
Executing: task list
TaskId  Function  Done% Array      State  Specific1  Specific2
------  --------  ----- ---------  -----  ---------  ---------
101     Scrub     24.6%  0         RUN    00000000   00000000
```

The following example shows the use of the `task suspend` command to suspend task 101:

```
AAC0> task suspend 101

Executing: task suspend 101
```

Use the `task list` command after using the `task suspend` command to display information about the task, as in the following example:

```
AAC0>task list
Executing: task list
TaskId  Function  Done%  Array      State  Specific1  Specific2
------  --------  ------ ---------  ----  ---------  ---------
101     Scrub     24.6%  0          SUS   00000000   00000000
```

As a result of using the `task suspend` command, the State column now displays `SUS` (suspended on the controller) instead of `RUN` (running on the controller).

Related Commands

`task` commands:

- `task list` (page 8-2)
- `task resume` (page 8-8)

9

enclosure Commands

In this Chapter

These `enclosure` commands operate on enclosure management devices that support Version 1.0 of the SCSI Accessed Fault-Tolerant Enclosures (SAF-TE) specification. Use the following syntax for `enclosure` commands:

```
enclosure command [/switch{=value}] [parameter]
```

enclosure activate slot

To activate a device in a specified slot of the specified SAF-TE enclosure management device, use the `enclosure activate slot` command. Specifically, this command makes the device that resides in the slot available on the SCSI channel.

Syntax

```
enclosure activate slot {enclosure} {slot}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device on which you want to activate a device. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{slot}`

Specifies the unit number associated with the slot. This is the slot that contains the device you want to activate. This unit number can range from 0 to 15 inclusive. Typically, the slot unit number is limited by the physical number of device slots in the enclosure management device.

Examples

Before activating a device, you might want to use the `enclosure show slot` command to determine the current device state.

As the following example shows, the current device state setting is **NOTACTIVATE**:

```
AAC0>enclosure show slot 0
Executing: enclosure show slot 0
Enclosure Slot scsiId Insert  Status
--------- ---- ------ -------  ----------------------------
   0       0   0:00:0    4     OK UNCONFIG NOTACTIVATE
   0       1   0:01:0    0     OK UNCONFIG ACTIVATE
   0       2   0:10:0    0     OK UNCONFIG ACTIVATE
   0       3   0:03:0    0     OK UNCONFIG ACTIVATE
   0       4   0:04:0     0   OK UNCONFIG  ACTIVATE
   1       0   1:00:0     0   OK CONFIG    ACTIVATE
   1       1   1:01:0     0   OK UNCONFIG  ACTIVATE
   1       2   1:10:0     0   OK UNCONFIG  ACTIVATE
   1       3   1:03:0     0   OK UNCONFIG ACTIVATE
   1       4   1:04:0     0   OK UNCONFIG ACTIVATE
```

The following example shows how to activate a device in slot **0** of enclosure management device **0** using the enclosure activate slot command:

```
AAC0>enclosure activate slot 0 0
Executing: enclosure activate slot 0 0
```

You may want to use the `enclosure show slot` command again to see the change in the device slot's status, as in the following example. Note that the **OK ACTIVATE** in the status column indicates that the device is activated and working properly.

```
AAC0>enclosure show slot 0
Executing: enclosure show slot 0
Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ----------------------------------------
   0       0   0:00:0     4   OK UNCONFIG ACTIVATE
   0       1   0:01:0     0   OK UNCONFIG ACTIVATE
   0       2   0:10:0     0   OK UNCONFIG ACTIVATE
   0       3   0:03:0     0   OK UNCONFIG ACTIVATE
   0       4   0:04:0     0   OK UNCONFIG ACTIVATE
   1       0   1:00:0     0   OK CONFIG   ACTIVATE
   1       1   1:01:0     0   OK UNCONFIG ACTIVATE
   1       2   1:10:0     0   OK UNCONFIG ACTIVATE
   1       3   1:03:0     0   OK UNCONFIG ACTIVATE
   1       4   1:04:0     0   OK UNCONFIG ACTIVATE
```

Related Commands

`enclosure` commands:

- `enclosure identify slot` (page 9-4)
- `enclosure list` (page 9-6)
- `enclosure prepare slot` (page 9-11)
- `enclosure show slot` (page 9-32)

enclosure identify slot

To identify a slot on a specified SAF-TE enclosure management device, use the `enclosure identify slot` command. Typically, an enclosure management device indicates to the operator which slot is currently being identified by flashing an adjacent light.

Syntax

```
enclosure identify slot {enclosure} {slot}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device on which you want to identify a slot. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{slot}`

Specifies the unit number associated with the slot. This is the slot that you want the enclosure management device to identify. This unit number can range from 0 to 15 inclusive. Typically, the slot unit number is limited by the physical number of device slots in the enclosure management device.

Switches

`/stop{=boolean}`

Specifies whether to stop the identification of the specified slot. If you set this switch to `TRUE`, the command stops identification of the specified slot. Typically, you use this switch to stop the identification of a slot that you previously identified with the `enclosure identify slot` command.

This switch defaults to `FALSE`.

Examples

The following example shows how to identify slot **1** in enclosure management device **0** using the enclosure identify slot command:

```
AAC0>enclosure identify slot 0 1

Executing: enclosure identify slot 0 1
```

Related Commands

`enclosure` commands:

- `enclosure activate slot` (page 9-2)
- `enclosure list` (page 9-6)
- `enclosure prepare slot` (page 9-11)
- `enclosure show slot` (page 9-32)

enclosure list

To display the list of components associated with one or all SAF-TE enclosure management devices connected to the controller, use the `enclosure list` command.

Syntax

```
enclosure list [/all{=boolean}] [/full{=boolean}]
[{enclosure}]
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device whose associated components you want to display. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

Switches

`/all{=boolean}`

Specifies whether to display the components for all currently connected enclosure devices. If you set this switch to `TRUE`, the command displays a list of all enclosure devices connected to the controller and their associated components.

This switch defaults to `FALSE` if you specify a unit number; otherwise, the switch defaults to `TRUE`.

`/full{=boolean}`

Specifies whether to display detailed component information. If you specify `TRUE`, the command displays detailed component information. If you specify `FALSE`, the command displays non-detailed component information.

This switch defaults to `FALSE`.

Examples

The following example shows a non-detailed list of components for enclosure management devices **0** and **1**:

```
AAC0>enclosure list /all=TRUE

Executing: enclosure list /all=TRUE
Enclosure Fan Power Slot Sensor Door ScsiId Speaker  Standard
Diagnostic
--------- --- ----- ---- ------ ---- ------ -------- -------- ---------
-
   0       1    1    5     1     5  0:06:0   No    SAF-TE    PASSED
   1       1    1    5     1     5  1:06:0   No     SAF-TE   PASSED
```

The non-detailed `enclosure list` display contains the following columns of information:

- Enclosure
- Fan
- Power
- Slot
- Sensor
- Door
- ScsiId
- Speaker
- Standard
- Diagnostic

The detailed `enclosure list` display contains the previous columns of information plus the following columns:

- EnclosureId
- VendorId
- ProductId
- Firmware

The following sections provide brief descriptions of each `enclosure list` column.

The Enclosure Column

The **Enclosure** column displays the ID of a specific enclosure management device. This is the ID that you specify for the `enclosure` parameter on the command line. In the example, the `enclosure list` command displays enclosure management device IDs **0** and **1** (because the `/all` switch was specified).

The Fan and Power Columns

The **Fan** column displays the number of fans on the enclosure management device. Typically, enclosure management devices support a maximum of sixteen fans. In the example, enclosure management devices **0** and **1** each have one fan.

The **Power** column displays the number of power supplies on the enclosure management device. Typically, enclosure management devices support a maximum of sixteen power supplies. In the example, enclosure management devices **0** and **1** each have one power supply.

Note: The "hot swapping" of enclosures is not supported. If you perform a hot swap, the Fan and Power columns may display incorrect information. Either perform a `controller pause_io` before the hot swap, or a `controller rescan` after the hot swap to display the correct information.

The Slot and Sensor Columns

The **Slot** column displays the number of device slots on the enclosure management device. Typically, enclosure management devices support a maximum of sixteen device slots. In the example, enclosure management devices **0** and **1** each have five device slots.

The **Sensor** column displays the number of temperature sensors on the enclosure management device. Typically, enclosure management devices support a maximum of sixteen temperature sensors. In the example, enclosure management devices **0** and **1** each have one temperature sensor.

The Door Column

The **Door** column displays the number of doors on the enclosure management device. Typically, enclosure management devices support a maximum of sixteen doors. In the example, enclosure management devices **0** and **1** each have five doors.

The Scsild Column

The **ScsiId** column displays the SCSI channel number (in the example, **0** and **1**), the SCSI ID (in the example, **06**), and the SCSI logical unit number (in the example, **0**) for the enclosure management device.

The Speaker Column

The **Speaker** column displays whether there is a speaker installed on the enclosure management device. Specifically, the column can display the items listed in Table 9-1.

Table 9-1 Speaker Status

Status	Meaning
Yes	Speaker is installed.
No	Speaker is not installed. This is the status displayed in the example.

The Standard Column

The **Standard** column displays a value that indicates whether the enclosure management device is compliant with the SAF-TE Interface Specification. This column should display the value **SAF-TE**, as in the example.

The Diagnostic Column

The **Diagnostic** column displays the status of the enclosure management device's boot-time diagnostic tests. Typically at boot time, an enclosure management device performs diagnostic self-tests on itself.

Specifically, the column can display the items listed in Table 9-2.

Table 9-2 Diagnostic Status

Status	Meaning
PASSED	No errors were detected and the diagnostic self-tests were successful. This is the value displayed in the example for both enclosure management devices.
FAILED	Errors were detected and the diagnostic self-tests were not successful.
UNKNOWN	The status of the diagnostic self-tests is unknown.

The EnclosureId, VendorId, and ProductId Columns

The **EnclosureId** column displays an identification number that is unique for each enclosure management device product sold by a particular manufacturer.

The **VendorId** column displays a string that identifies the manufacturer of the enclosure management device. In the example, the string that identifies the manufacturer of enclosure management devices **0** and **1** is **ESG-SHV**.

The **ProductId** column displays a string that identifies the enclosure management device's product ID. In the example, the product ID for enclosure management devices **0** and **1** is **SCA HSBP M2**.

The Firmware Column

The **Firmware** column displays the current firmware revision of the enclosure management device. In the example, the firmware revision for enclosure management devices **0** and **1** is **0.14**.

Related Commands

`enclosure` commands:

- `enclosure show fan` (page 9-26)
- `enclosure show power` (page 9-29)
- `enclosure show slot` (page 9-32)
- `enclosure show status` (page 9-36)
- `enclosure show temperature` (page 9-41)

enclosure prepare slot

To indicate to the SAF-TE enclosure management device that it should take the appropriate action to prepare the device in a specific slot for removal or insertion, use the `enclosure prepare slot` command.

This command may perform such operations as shutting off power to the device slot. This allows users to safely remove a disk device from or insert a disk device into the device slot.

Syntax

```
enclosure prepare slot {enclosure} {slot}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device on which you want to prepare a slot. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{slot}`

Specifies the unit number associated with the slot. This is the slot that contains the device you want to prepare to make available. This unit number can range from 0 to 15 inclusive.

Examples

Before preparing a device in the slot, you might want to use the `enclosure show slot` command to determine the current device slot status:

```
AAC0>enclosure show slot
Executing: enclosure show slot
Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ----------------------------------------
   0       0   0:00:0     4   OK UNCONFIG ACTIVATE
   0       1   0:01:0     0   OK UNCONFIG ACTIVATE
   0       2   0:10:0     0   OK UNCONFIG ACTIVATE
   0       3   0:03:0     0   OK UNCONFIG ACTIVATE
   0       4   0:04:0     0   OK UNCONFIG ACTIVATE
   1       0   1:00:0     0   OK CONFIG   ACTIVATE
   1       1   1:01:0     0   OK UNCONFIG ACTIVATE
   1       2   1:10:0     0   OK UNCONFIG ACTIVATE
   1       3   1:03:0     0   OK UNCONFIG ACTIVATE
   1       4   1:04:0     0   OK UNCONFIG ACTIVATE
```

The following example prepares the device in slot **0** on enclosure management device **0** for insertion or removal as a result of using the `enclosure prepare slot` command:

```
AAC0>enclosure prepare slot 0 0
Executing: enclosure prepare slot 0 0
```

You may want to use the `enclosure show slot` command again to see the change in the device slot's status, as in the following example:

```
AAC0>enclosure show slot
Executing: enclosure show slot
Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ----------------------------------------
   0       0   0:00:0     4   OK UNCONFIG I/R READY NOTACTIVATE
   0       1   0:01:0     0   OK UNCONFIG ACTIVATE
   0       2   0:10:0     0   OK UNCONFIG ACTIVATE
   0       3   0:03:0     0   OK UNCONFIG ACTIVATE
   0       4   0:04:0     0   OK UNCONFIG ACTIVATE
   1       0   1:00:0     0   OK CONFIG   ACTIVATE
   1       1   1:01:0     0   OK UNCONFIG ACTIVATE
   1       2   1:10:0     0   OK UNCONFIG ACTIVATE
   1       3   1:03:0     0   OK UNCONFIG ACTIVATE
   1       4   1:04:0     0   OK UNCONFIG ACTIVATE
```

Note that the **Status** column now displays **I/R READY** and **NOTACTIVATE**, which means the device slot is off and ready for the insertion or removal of a device.

Related Commands

`enclosure` commands:

- `enclosure activate slot` (page 9-2)
- `enclosure identify slot` (page 9-4)
- `enclosure list` (page 9-6)
- `enclosure show slot` (page 9-32)

enclosure set alarm

To turn the audible alarm on a specific SAF-TE enclosure management device on or off, use the `enclosure set alarm` command.

Syntax

```
enclosure set alarm [/on{=boolean}] {enclosure}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device whose audible alarm you want to set. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

Switches

`/on{=boolean}`

Specifies whether to turn the enclosure management device's audible alarm on or off. If you set this switch to `TRUE`, the command turns on the audible alarm. If you set this switch to `FALSE`, the command turns off the audible alarm.

The default is `FALSE`.

Examples

The following example turns on the audible alarm on enclosure management device **0**:

```
AAC0>enclosure set alarm /on=TRUE 0
Executing: enclosure set alarm /on=TRUE 0
```

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)

enclosure set door

To lock or unlock a specific door on a specific SAF-TE enclosure management device, use the `enclosure set door` command.

Syntax

```
enclosure set door [/lock{=boolean}] {enclosure}
{door}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device on which you want to lock the door. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{door}`

Specifies the unit number of the door for which you want to set the lock. This number can range from 0 to 15 inclusive.

Switches

`/lock{=boolean}`

Specifies whether to lock the door associated with the specified unit number. If you set this switch to `TRUE`, the command locks the door associated with the specified unit number. If you set this switch to `FALSE`, the command unlocks the door associated with the specified unit number.

This switch defaults to `FALSE`.

Examples

The following example locks the door associated with unit **1** on enclosure management device **0**:

```
AAC0>enclosure set door /lock=TRUE 0 1
Executing: enclosure set door /lock=TRUE 0 1
```

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)

enclosure set fan

To set the speed of a specific fan on a specific SAF-TE enclosure management device, use the `enclosure set fan` command.

Syntax

```
enclosure set fan [/off{=boolean}] {enclosure}
{fan} [{speed}]
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device on which you want to set the speed of a specific fan. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{fan}`

Specifies the unit number associated with the fan whose speed you want to set. The unit number for the fan ranges from 0 to 15 inclusive.

`{speed}`

Specifies the speed at which you want to set this fan. Table 9-3 lists the fan speed values you can specify.

Table 9-3 Fan Speed Values

Value	Meaning
0	Turns off the fan
1	Runs the fan at 1/4 the full speed.
2	Runs the fan at 1/2 the full speed.
3	Runs the fan at 3/4 the full speed.
4	Runs the fan at full speed.

The fan speed defaults to full speed.

Switches

`/off{=boolean}`

Specifies whether to turn off the fan associated with the specified unit number. If you set this switch to `TRUE`, the command turns off the fan associated with the specified unit number. You must turn on the fan if you want to set its speed.

This switch defaults to `FALSE`, which means the fan remains on and you can set its speed.

Examples

The following example sets fan unit **0** on enclosure management device **0** to full speed:

```
AAC0>enclosure set fan 0 0 4
Executing: enclosure set fan 0 0 4
```

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure set alarm` (page 9-13)
- `enclosure set power` (page 9-19)

enclosure set interval

To set the monitor reporting interval for a specific SAF-TE enclosure management device, use the `enclosure set interval` command. The monitor reporting interval indicates how often the controller checks the status of the enclosure management device and its associated components. In setting this value, consider the needs of your computing environment. Setting the monitor reporting level to less than ten seconds could cause slower system performance.

Syntax

```
enclosure set interval {second}
```

Parameters

`{second}`

Specifies the number of seconds for the monitor reporting interval. You can specify 1 to n seconds where n represents the number of seconds for the monitor reporting interval. (The maximum value is 86,400, the number of seconds in a day.)

The default is 10 seconds.

Examples

Before setting the monitor reporting interval, you might want to use the `enclosure show status` command to determine the current monitor reporting interval setting.

As the following example shows, the current monitor reporting interval setting is **10**:

```
AAC0>enclosure show status
Enclosure UpTime D:H:M   PowerCycle Interval Door     Alarm
--------- -------------- ---------- -------- -------- -------
   0             0:00:00        0        10  UNLOCKED OFF
.
.
.
Enclosure UpTime D:H:M   PowerCycle Interval Door     Alarm
--------- -------------- ---------- -------- -------- -------
   1             0:00:00        0       10   UNLOCKED OFF
```

The following example sets the monitor reporting interval for enclosure management devices **0** and **1** to **20** seconds:

```
AAC0>enclosure set interval 20
Executing: enclosure set interval 20
```

You may want to use the `enclosure show status` command again to see the change in the power supply's status, as in the following example:

```
Enclosure UpTime D:H:M   PowerCycle Interval Door     Alarm
--------- -------------- ---------- -------- -------- -------
   0             0:00:00         0       20   UNLOCKED OFF
.
.
.
Enclosure UpTime D:H:M   PowerCycle Interval Door     Alarm
--------- -------------- ---------- -------- -------- -------
   1             0:00:00         0       20   UNLOCKED OFF
```

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure show status` (page 9-36)

enclosure set power

To turn the power of a specific power supply on a specific SAF-TE enclosure management device on or off, use the `enclosure set power` command. Typically, most enclosure management devices turn on their associated power supplies by default after power up.

Syntax

```
enclosure set power [/off{=boolean}] {enclosure}
{powersupply}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device on which you want to turn the power of a specific power supply on or off. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{powersupply}`

Specifies the unit number associated with the power supply whose power you want to turn on or off. The unit number for the power supply can range from 0 to 15 inclusive.

Switches

`/off{=boolean}`

Specifies whether to turn the power supply associated with the specified unit number on or off. If you set this switch to `TRUE`, the command turns off the power supply associated with the specified unit number. If you specify `FALSE`, the command turns on the power supply associated with the specified unit number.

This switch defaults to `FALSE`, which means the command turns on the power supply if you do not specify the switch.

Examples

Before turning a power supply on or off, you might want to use the `enclosure show power` command to determine the current setting for the power supply or supplies. As the following example shows, power supply **0** on enclosure management device **1** is off and operational:

```
AAC0>enclosure show power
Executing: enclosure show power
Enclosure Power State        Status
--------- ----- ------------ -------
   0       0     ON           OK
   1       0     ON           OK
```

The following example turns on power supply **0** on enclosure management device **1**:

```
AAC0>enclosure set power 1 0
Executing: enclosure set power 1 0
```

You may want to use the `enclosure show power` command again to see the change in the power supply's status, as in the following example:

```
AAC0>enclosure show power
Executing: enclosure show power
Enclosure Power State        Status
--------- ----- ------------ -------
   0       0     ON           OK
   1       0     ON           OK
```

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure show power` (page 9-29)

enclosure set scsiid

To set the SCSI device ID for a specific device slot on a specific SAF-TE enclosure management device, use the `enclosure set scsiid` command. As discussed in other chapters, a SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, and so on), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive). The `enclosure set scsiid` command allows you to set the specified device slot to a specific SCSI device ID (the middle number of a SCSI ID).

Notes

Before setting the SCSI device ID for a specific device slot on a specific SAF-TE enclosure management device, use the `controller pause_io` command. This command pauses all I/O activity on the currently opened controller.

You cannot set the SCSI device ID for a specific device slot to a SCSI device ID that is already in use. The command checks for SCSI device ID conflicts and will display an appropriate error message if you try to assign a SCSI device ID that is already in use.

Syntax

```
enclosure set scsiid {enclosure} {slot}
{device_id}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device. The device slot to be assigned a SCSI channel number is located on this enclosure management device. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{slot}`

Specifies the unit number associated with the device slot that you want to assign a SCSI channel number. This integer can range from 0 to 15 inclusive.

`{device_id}`

Specifies the ID (0 through 15 inclusive) of the SCSI device you want to assign to the specified device slot.

Examples

Before setting a device slot to a specific SCSI ID, you might want to use the `enclosure show slot` command to determine the current SCSI ID setting for the device slots, as in the following example:

```
AAC0>enclosure show slot
Executing: enclosure show slot
Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ----------------------------------------
--
   0       0   0:00:0     4   OK NOTACTIVATE
   0       1   0:01:0     0   OK UNCONFIG ACTIVATE
   0       2   0:10:0     0   OK UNCONFIG ACTIVATE
   0       3   0:03:0     0   OK UNCONFIG ACTIVATE
   0       4   0:04:0     0   OK UNCONFIG ACTIVATE
   1       0   1:00:0     0   OK CONFIG ACTIVATE
   1       1   1:01:0     0   OK UNCONFIG ACTIVATE
   1       2   1:10:0     0   OK UNCONFIG ACTIVATE
   1       3   1:03:0     0   OK UNCONFIG ACTIVATE
   1       4   1:04:0     0   OK UNCONFIG ACTIVATE
```

The following example sets device slot **2** on enclosure management device **0** to SCSI device ID **05**. The example also shows how to pause all I/O activity on the controller by using the `controller pause_io` command:

```
AAC0>controller pause_io 30
Executing: controller pause_io 30
AAC0>enclosure set scsiid 0 2 5
Executing: enclosure set scsiid 0 2 5
```

You may want to use the `enclosure show slot` command again to see the change in the SCSI ID setting, as in the following example:

```
AAC0>enclosure show slot
Executing: enclosure show slot
Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ----------------------------------------
--
   0       0   0:00:0     4   OK NOTACTIVATE
   0       1   0:01:0     0   OK UNCONFIG ACTIVATE
   0       2   0:05:0     0   OK UNCONFIG ACTIVATE
   0       3   0:03:0     0   OK UNCONFIG ACTIVATE
   0       4   0:04:0     0   OK UNCONFIG ACTIVATE
   1       0   1:00:0     0   OK CONFIG ACTIVATE
   1       1   1:01:0     0   OK UNCONFIG ACTIVATE
   1       2   1:10:0     0   OK UNCONFIG ACTIVATE
   1       3   1:03:0     0   OK UNCONFIG ACTIVATE
   1       4   1:04:0     0   OK UNCONFIG ACTIVATE
```

Related Commands

`controller` commands:

- `controller pause_io` (page 4-12)

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure show slot` (page 9-32)

enclosure set temperature

To set the temperature threshold on a specific SAF-TE enclosure management device, use the `enclosure set temperature` command. Typically, an enclosure management device contains individual temperature sensors.

With this command, you can set a temperature threshold so that the enclosure management device alerts you if one or more of the temperature sensors exceeds the threshold.

Syntax

```
enclosure set temperature {enclosure} {degree}
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device on which you want to set a temperature threshold. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

`{degree}`

Specifies the temperature (in degrees Fahrenheit) at which you want to set the threshold for this enclosure management device. You can specify from 0 to 255 degrees Fahrenheit.

The default is 120 degrees Fahrenheit.

Examples

Before setting the threshold temperature, you might want to use the `enclosure show temperature` command to determine the current temperature(s). As the following example shows, the current temperature is **70** degrees Fahrenheit and the threshold setting is **120** degrees Fahrenheit:

```
Enclosure Sensor Temperature Threshold Status
--------- ------ ----------- --------- --------
   0       0       70 F         120    NORMAL
```

The following example sets the threshold temperature associated with enclosure management device **0** to **100** degrees Fahrenheit:

```
AAC0>enclosure set temperature 0 100
Executing: enclosure set temperature 0 100
```

You may want to use the `enclosure show temperature` command again to see the change in threshold temperature, as in the following example:

```
AAC0>enclosure show temperature 0
Executing: enclosure show temperature 0
Enclosure Sensor Temperature Threshold Status
--------- ------ ----------- --------- --------
   0       0       70 F         100    NORMAL
```

The **Threshold** column now displays **100** degrees.

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure show temperature` (page 9-41)

enclosure show fan

To display the status of a specific fan on a specific SAF-TE enclosure management device, use the `enclosure show fan` command. You can also use this command to display the status of all fans on the enclosure management device or devices.

Syntax

```
enclosure show fan [{enclosure}] [{fan}]
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device for which you want to display fan status information. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

If you do not specify a unit number, the command displays the fan status for all enclosure management devices connected to the controller.

`{fan}`

Specifies the unit number associated with the fan whose status you want to display. This integer can range from 0 to 15 inclusive.

If you do not specify a unit number, the command displays the status for all fans on the enclosure management device or devices.

Examples

The following example displays the status of all fans on enclosure management devices **0** and **1**:

```
AAC0>enclosure show fan
Executing: enclosure show fan
Enclosure Fan Status
--------- --- -------------
   0       0   OK
   1       0   NOT INSTALLED
```

As the example shows, the fan status display contains the following columns:

- Enclosure
- Fan
- Status

The following sections describe the information that the `enclosure show fan` command can display in these columns.

The Enclosure and Fan Columns

The **Enclosure** column displays the ID associated with the enclosure management device that you specified on the command line. If you did not specify an ID, the `enclosure show fan` command displays the IDs for all enclosure management devices connected to the controller. In the example, the `enclosure show fan` command displays enclosure management device IDs **0** and **1**.

The **Fan** column displays the unit number associated with the fan that you specified on the command line. If you did not specify a unit number, the `enclosure show fan` command displays the unit numbers for all fans on the enclosure management device or devices. In the example, there is one fan (unit **0**) each on enclosure management devices **0** and **1**.

The Status Column

The **Status** column displays the status for each fan listed in the display. The `enclosure show fan` command can display the fan status values listed in Table 9-4.

Table 9-4 Fan Status Values

Status	Meaning
OK	The fan is operational. In the example, fan 0 on enclosure management device 0 is operational.
BAD	The fan is not working properly.
NOT INSTALLED	The fan is not installed. In the example, fan 0 on enclosure management device 1 is not installed.
UNKNOWN	The status of the fan is not reportable. Or, the fan status is unknown.

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure set fan` (page 9-15)
- `enclosure show status` (page 9-36)

enclosure show power

To display the status of a specific power supply on a specific SAF-TE enclosure management device, use the `enclosure show power` command. You can also use this command to display the status of all power supplies on the enclosure management device or devices.

Syntax

```
enclosure show power [{enclosure}]
[{powersupply}]
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device for which you want to display power supply status information. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

If you do not specify an ID, the command displays the power supply status for all enclosure management devices connected to the controller.

`{powersupply}`

Specifies the unit number associated with the power supply whose status you want to display. This integer can range from 0 to 15 inclusive.

If you do not specify a unit number, the command displays the status for all power supplies on the enclosure management device or devices.

Examples

The following example displays the status of the power supplies on enclosure management devices **0** and **1**:

```
AAC0>enclosure show power
Executing: enclosure show power
Enclosure Power State        Status
--------- ----- ------------ -------
   0       0     OK           ON
   1       0     OK           OFF
```

As the example shows, the power supply status display contains the following columns:

- Enclosure
- Power
- State
- Status

The following sections describe the information that the `enclosure show power` command can display in these columns.

The Enclosure and Power Columns

The **Enclosure** column displays the ID associated with the enclosure management device that you specified on the command line. If you did not specify an ID, the `enclosure show power` command displays the IDs for all enclosure management devices connected to the controller. In the example, **0** is specified.

The **Power** column displays the unit number associated with the power supply that you specified on the command line. If you did not specify a unit number, `enclosure show power` command displays the unit numbers for all power supplies on the enclosure management device or devices. In the example, there is one power supply (unit **0**) each on enclosure management devices **0** and **1**.

The State and Status Columns

The **State** column displays the state and the **Status** column displays the status of the power supply. Specifically, the column can display the items listed in Table 9-5.

Table 9-5 State and Status Columns

Status	State	Meaning
`ON`	`OK`	The power supply is on and operational. In the example, power supply 0 on enclosure management device 0 has these values.
`OFF`	`OK`	The power supply is off and operational. In the example, power supply 0 on enclosure management device 1 has these values.

Table 9-5 State and Status Columns

Status	State	Meaning
ON	BAD	The power supply is on and not working properly.
OFF	BAD	The power supply is off and not working properly.
INSTALLED	UNKNOWN	The power supply is installed and its status is not reportable or unknown.
NOT INSTALLED	Not Applicable	The power supply is not installed.
UNKNOWN	UNKNOWN	The state and status of the power supply is not reportable or unknown.

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure set power` (page 9-19)

enclosure show slot

To display the status of a specific device slot on a specific SAF-TE enclosure management device, use the `enclosure show slot` command. You can also use this command to display the status of all device slots on the enclosure management device or devices.

Syntax

```
enclosure show slot [{enclosure}] [{slot}]
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device for which you want to display device slot status information. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

If you do not specify an ID, the command displays the device slot status for all enclosure management devices connected to the controller.

`{slot}`

Specifies the unit number associated with the device slot whose status you want to display. This integer can range from 0 to 15 inclusive. Typically, the slot unit number is limited by the physical number of device slots in the enclosure management device.

If you do not specify a unit number, the command displays the status for all device slots on the enclosure management device or devices.

Examples

The following example displays the status of all device slots on enclosure management devices **0** and **1**:

```
AAC0>enclosure show slot
Executing: enclosure show slot
Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ----------------------------------
   0       0   0:00:0     4   OK
   0       1   0:01:0     0   UNCONFIG
   0       2   0:10:0     0   UNCONFIG
   0       3   0:03:0     0   UNCONFIG
   0       4   0:04:0     0   UNCONFIG
   1       0   1:00:0     0   UNCONFIG
   1       1   1:01:0     0   UNCONFIG
   1       2   1:10:0     0   UNCONFIG
   1       3   1:03:0     0   UNCONFIG
   1       4   1:04:0     0   UNCONFIG
```

As the example shows, the device slot status display contains the following columns:

- Enclosure
- Slot
- scsiId
- Insert
- Status

The following sections describe the information that the `enclosure show slot` command can display in these columns.

The Enclosure and Slot Columns

The **Enclosure** column displays the ID associated with the enclosure management device that you specified on the command line. If you did not specify an ID, the `enclosure show slot` command displays the IDs for all enclosure management devices connected to the controller. In the example, the `enclosure show slot` command displays enclosure management device IDs **0** and **1**.

The **Slot** column displays the unit number associated with the device slot that you specified on the command line. If you did not specify a unit number, the `enclosure show slot` command displays the unit numbers for all device slots on the enclosure management device or devices. In the example, the `enclosure show slot` command displays the device slot status for device slots **0** through **4** for enclosure management devices **0** and **1**.

The scsiId Column

The **scsiId** column displays the SCSI ID associated with the device slot. A SCSI ID consists of a SCSI channel number (for example, 0, 1, 2, 3, etc.), SCSI device ID (0 through 15 inclusive), and SCSI device logical unit number (0 through 7 inclusive). In the example, the `enclosure show slot` command displays the SCSI IDs for device slots **0** through **4**.

The Insert Column

The **Insert** column displays a number that indicates how many times a user inserted a device into the device slot. In the example, a user inserted a device into device slot 0 four times.

The Status Column

The **Status** column displays status information about a device slot. Specifically, the column can display the values listed in Table 9-6.

Table 9-6 Status Column

Status	Meaning
OK	The device that resides in this slot is operational. In the example, the device that resides in device slot 0 on enclosure management device 0 is operational.
ERROR	The device that resides in this slot has error conditions.
FAULTY	The device that resides in this slot has some hardware or data fault.
FAILOVER	The device that resides in this slot is in a rebuild.
FAILED	The device that resides in this slot is in a device array. One member of this device array has an error.
CRITICAL	The device that resides in this slot is in a device array that was previously fault-tolerant and is now non-fault-tolerant.
SCRUB	The device that resides in this slot is in a device array that is undergoing a parity check operation.
WARNING	The device that resides in this slot is likely to fail in the near future.

Table 9-6 Status Column

Status	Meaning
UNCONFIG	The device that resides in this slot is not currently configured as a member of a device array or as a hot spare.
HOTFAILOVER	The device that resides in this slot is configured as a hot spare.
FAILOVERSTOP	The device that resides in this slot was in a rebuild operation, but the rebuild operation terminated abnormally or unsuccessfully.
EMPTY	No physical device resides in this slot.
I/R READY	The slot is ready for the physical insertion or removal of a device.
NOTACTIVATE	The slot is not activated. The device that resides in the slot cannot be accessed on the SCSI channel.
ACTIVATE	The slot is activated. The device that resides in the slot can be accessed on the SCSI channel.

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure set scsiid` (page 9-21)

enclosure show status

To display the general status of a specific SAF-TE enclosure management device, use the `enclosure show status` command. You can also use this command to display the general status of all enclosure management devices connected to the controller.

In addition to general status information, this command also displays status information available from the following commands:

- `enclosure show fan`
- `enclosure show power`
- `enclosure show slot`
- `enclosure show temperature`

Syntax

```
enclosure show status [{enclosure}]
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device for which you want to display status information. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

If you do not specify an ID, the command displays status information for all enclosure management devices connected to the controller.

Examples

The following example displays status information for enclosure management devices **0** and **1**:

```
AAC0>enclosure show status
Executing: enclosure show status
Enclosure UpTime D:H:M   PowerCycle Interval Door     Alarm
--------- -------------- ---------- -------- -------- -------
   0             0:04:25        2       20   UNLOCKED OFF

Enclosure Fan Status
--------- --- -------------
   0       0   OK

Enclosure Power State        Status
--------- ----- ------------ -------
   0       0     UNKNOWN       UNKNOWN

Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ---------------------------------------
--
   0       0   0:00:0     0   OK UNCONFIG ACTIVATE
   0       1   0:01:0     0   OK UNCONFIG ACTIVATE
   0       2   0:10:0     0   OK UNCONFIG ACTIVATE
   0       3   0:03:0     0   OK UNCONFIG ACTIVATE
   0       4   0:04:0     0   OK UNCONFIG ACTIVATE

Enclosure Sensor Temperature Threshold Status
--------- ------ ----------- --------- --------
   0       0       72 F         120    NORMAL

Enclosure UpTime D:H:M   PowerCycle Interval Door     Alarm
--------- -------------- ---------- -------- -------- -------
   1             2:20:25        3       20   UNLOCKED OFF

Enclosure Fan Status
--------- --- -------------
   1       0   NOT INSTALLED

Enclosure Power State        Status
--------- ----- ------------ -------
   1       0     UNKNOWN       UNKNOWN

Enclosure Slot scsiId Insert  Status
--------- ---- ------ ------- ---------------------------------------
--
   1       0   1:00:0     0   UNCONFIG
   1       1   1:01:0     0   UNCONFIG
   1       2   1:10:0     0   UNCONFIG
   1       3   1:03:0     0   UNCONFIG
   1       4   1:04:0     0   UNCONFIG

Enclosure Sensor Temperature Threshold Status
--------- ------ ----------- --------- --------
   1       0       75 F         120    NORMAL
```

The general status of an enclosure management device is encapsulated in the following columns:

- Enclosure
- UpTime D:H:M
- PowerCycle
- Interval
- Door
- Alarm

The following sections describe the information that the `enclosure show status` command can display in these columns. For information about the status displays for fans, power supplies, device slots, and temperature sensors see the *Related Commands* section.

The Enclosure Column

The **Enclosure** column displays the ID associated with the enclosure management device that you specified on the command line. If you did not specify an ID, the `enclosure show status` command displays the IDs for all enclosure management devices connected to the controller.

The UpTime D:H:M Column

The **UpTime D:H:M** column displays the total time (days:hours:minutes) that the enclosure management device has been running. This count is cumulative over the life of the enclosure management device.

The PowerCycle Column

The **PowerCycle** column displays the number of times a user has powered on the enclosure management device. In the example, a user powered on enclosure management device 0 two times and enclosure management device 1 three times. This count is cumulative over the life of the enclosure management device.

The Interval Column

The **Interval** column displays the monitor interval. You can set the monitor interval with the `enclosure set interval` command. In the example, the monitor interval reporting level for both enclosure management devices is **20** seconds.

The Door and Alarm Columns

The **Door** column displays the status of the door lock. The `enclosure show status` command can display the door lock status values listed in Table 9-7.

Table 9-7 Door Lock Status Values

Status	Meaning
LOCKED	The door is currently locked.
UNLOCKED	The door is currently unlocked or there is no controllable door lock installed. This is the value displayed in the example.
UNKNOWN	The status of the door lock is not reportable. Or, the door lock status is unknown.

The **Alarm** column displays the status of the audible alarm. The `enclosure show status` command can display the audible alarm status values listed in Table 9-8.

Table 9-8 Audible Alarm Status Values

Status	Meaning
ON	The alarm is currently on.
OFF	The alarm is currently off or there is no alarm installed. This is the value displayed in the example.

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure set interval` (page 9-17)
- `enclosure show fan` (page 9-26)

- `enclosure show power` (page 9-29)
- `enclosure show slot` (page 9-32)
- `enclosure show temperature` (page 9-41)

enclosure show temperature

To display the temperature status of a specific temperature sensor on a specific SAF-TE enclosure management device, use the `enclosure show temperature` command. You can also use this command to display the temperature status of all temperature sensors on the enclosure management device or devices.

Syntax

```
enclosure show temperature [{enclosure}]
[{sensor}]
```

Parameters

`{enclosure}`

Specifies the ID associated with the enclosure management device for which you want to display temperature status information. For this version, the controller supports a maximum of eight enclosure management devices on each bus.

If you do not specify an ID, the command displays the temperature status information for all enclosure management devices connected to the controller.

`{sensor}`

Specifies the unit number associated with the temperature sensor whose status you want to display. This integer can range from 0 to 15 inclusive.

If you do not specify a unit number, the command displays the status for all temperature sensors on the enclosure management device or devices.

Examples

The following example displays the temperature status of temperature sensor unit **0** on enclosure management device **0**:

```
AAC0>enclosure show temperature 0 0
Executing: enclosure show temperature 0 0
Enclosure Sensor Temperature Threshold Status
--------- ------ ----------- --------- --------
   0       0        70          120    NORMAL
```

As the example shows, the temperature sensor status display contains the following columns:

- Enclosure
- Sensor
- Temperature
- Threshold
- Status

The following sections describe the information that the `enclosure show temperature` command can display in these columns.

The Enclosure Column

The **Enclosure** column displays the ID associated with the enclosure management device that you specified on the command line. If you did not specify an ID, the `enclosure show temperature` command displays the IDs for all enclosure management devices connected to the controller. In the example, **0** is specified.

The Sensor Column

The **Sensor** column displays the unit number associated with the temperature sensor that you specified on the command line. If you did not specify a unit number, the `enclosure show temperature` command displays the unit numbers for all temperature sensors on the enclosure management device or devices.

The Temperature Column

The **Temperature** column displays the internal enclosure temperature for this temperature sensor, in degrees Fahrenheit. The temperature can range from -10 degrees to 245 degrees Fahrenheit. In the example, the temperature is **70** degrees Fahrenheit.

The Threshold Column

The **Threshold** column displays the temperature at which an alarm will be sounded if the temperature goes above the threshold. In the example, the temperature threshold is **120** degrees Fahrenheit. You can change the temperature threshold by using the `enclosure set temperature` command.

The Status Column

The **Status** column displays the status of the temperature displayed in the **Temperature** column. The `enclosure show temperature` command can display the temperature status values listed in Table 9-9.

Table 9-9 Temperature Status Values

Status	Meaning
NORMAL	The temperature is within the normal range for this temperature sensor. In this case, the enclosure management device does not send an alarm. This is the value that appears in the example.
ABNORMAL	The temperature exceeds the normal range for this temperature sensor. In this case, the enclosure management device may send an alarm.

Related Commands

`enclosure` commands:

- `enclosure list` (page 9-6)
- `enclosure set temperature` (page 9-24)
- `enclosure show status` (page 9-36)

A

Automated Scripts

This appendix describes how to write DOS batch files and CLI command scripts to perform the `controller details` task.

You can use the CLI commands alone in CLI command scripts or in combination with DOS commands in DOS batch files. Also, you can invoke CLI command scripts in a DOS batch file. You should already understand how to write DOS batch files and be familiar with Windows-related backup commands.

To execute a CLI command script, type the @ (at sign) followed by the file name that contains the CLI commands, for example:

```
AAC0>@array.txt
```

Creating an Automated Script

In its simplest form, a CLI command script is a text file that contains valid CLI commands. A carriage return linefeed follows each command. The following CLI command script:

1. Opens controller `aac0` by invoking the CLI command `open`.
2. Invokes `controller details`.
3. Closes controller `aac0` by invoking the CLI command `close`.
4. Exits the example CLI command script.

```
open AAC0
controller details
close AAC0
exit
```

Adaptec, Inc.
691 South Milpitas Boulevard
Milpitas, CA 95035 USA

Part Number: 513852-06, Ver. AA 03/04