AMCC STORAGE

3ware®

Serial ATA RAID Controller

Command Line Interface

Supports the 9000 Series

PN: 720-0115-01
March 2005

CLI Guide

Copyright

Trademarks

Disclaimer

Table of Contents

About This Guide

3ware 9000 Series Serial ATA Controller CLI Guide provides instructions for configuring and maintaining your 3ware controller using 3ware's command line interface (CLI).

This guide assumes that you have already installed your controller in your system. If you have not yet done so, see *3ware 9000 Series Serial ATA RAID Controller Installation Guide* for instructions.

How this Guide is Organized

There are often multiple ways to accomplish the same configuration and maintenance tasks for your 3ware controller. While this manual includes instructions for performing tasks using the command line interface, two additional tools are available:

3ware BIOS Manager

3DM®2 (3ware Disk Manager)

For information about these tools, see *3ware 9000 Series Serial ATA RAID Controller User Guide*.

Table 1: Sections in this Guide

Section	Description
Introduction to 3ware Command Line Interface	Installation, features, concepts
Primary CLI Syntax Reference	Describes individual commands using the primary syntax
Legacy CLI Syntax Reference	Describes individual commands using the legacy syntax

Introduction to the 3ware Command Line Interface

The 3ware SATA RAID Controller Command Line Interface (CLI) for Linux, Windows, and FreeBSD is provided to manage 7000, 8000, and 9000-series 3ware ATA and Serial ATA RAID controllers. Multiple 3ware RAID controllers can be managed using the CLI via a command line or script.

Note: All information contained in this document that describes usage for the 3ware 9000 series products will not work with 3ware 7000 or 8000 series controllers.

Warning!

For all of the functions of the 3ware CLI to work properly, you must have the proper CLI, firmware, and driver versions installed. Check www.3ware.com for the latest versions and upgrade instructions.

Features

3ware CLI is a command line interface storage management application for 3ware RAID Controllers. It provides controller, logical unit, drive, and BBU (Battery Backup Unit) management. It can be used in both interactive and batch mode, providing higher level API functionalities.

The 3ware CLI provides the functionality of the 3ware Disk Management (3DM®2) utility through a command line interface. You can use it to view unit status and version information and perform maintenance functions such as adding or removing drives. 3ware CLI also includes advanced features for creating and deleting RAID units online.

Note: For complete information on 3DM 2 and for information about configuring or upgrading your computer, refer to the *3ware 9000 Series Serial ATA RAID Controller User Guide*.

Supported Operating Systems

- **Windows.** Windows 2000 with SP3 or newer, Windows XP with SP1 or newer, and Windows Server 2003, both 32-bit and 64-bit x64.
- **Linux**. Redhat, SuSE
- **FreeBSD**

For specific versions of Linux and FreeBSD that are supported for the 3ware CLI, see the Release Notes.

Terminology

This document uses the following terminology:

Logical Units. Usually shortened to “units.” These are block devices presented to operating systems. A logical unit can be a one-tier, two-tier, or three-tier arrangement. JBOD, Spare, and Single logical units are examples of one-tier units. RAID 1 and RAID 5 are examples of two-tier units and as such will have sub-units. RAID 10 and RAID 50 are examples of three-tier units and as such will have sub-sub-units.

Port. A controller has one or many ports (typically 4, 8, 12). Each port can be attached to a single disk drive.

For additional information about 3ware controller concepts and terminology, see *3ware 9000 Series Serial ATA RAID Controller User Guide*.

Installing the 3ware CLI

Warning!

AMCC does not recommend installing both 3DM 2 and CLI on the same system. Conflicts may occur. For example, if both are installed, alarms will be captured only by 3DM. You should use either CLI or 3DM 2 to manage your 3ware RAID controllers.

Installing the 3ware CLI on Windows

3ware CLI can be installed or run directly from the 3ware software CD, or the latest version can be downloaded from the 3ware web site, www.3ware.com. Online manual pages are also available in nroff and html formats. These are located in `/packages/cli/tw_cli.8.html` or `tw_cli.8.nroff`.

To install 3ware CLI on Windows, copy the file `tw_cli.exe` to the directory from which you want to run the program. CLI is located on the 3ware CD in the directory `\packages\cli\windows`

Note: CLI comes in both 32-bit and 64-bit versions. Be sure to copy the correct version for the version of the operating system you are using.

CLI can only be run by an administrator or a user with administrator rights. Without the correct privileges, CLI will prompt and then exit when the application is executed.

To start CLI, do one of the following:

- Start the 3ware CD and at the 3ware Escalade menu, click **Run CLI.**
- Or, open a console window and at the command prompt, enter `tw_cli`
- OR, double-click the CLI icon in a folder.

  The CLI prompt is displayed in a DOS console window.

Installing the 3ware CLI on Linux and FreeBSD

3ware CLI can be installed or run directly from the 3ware software CD, or the latest version can be downloaded from the 3ware web site, www.3ware.com.

To install the 3ware CLI, copy tw_cli to the directory from which you want to run the program. CLI is located on the 3ware CD in `/packages/cli/freebsd` or `/packages/cli/linux`.

Online manual pages are also available in nroff and html formats. These are located in `/packages/cli/tw_cli.8.html` or `tw_cli.8.nroff`.

You will need to be root or have root privileges to install the CLI to `/usr/sbin` and to run the CLI.

```
Filename: tw_cli
```

To install the CLI to a different location, change `/usr/sbin/` to the desired location.

Notes:

The installation location needs to be in the environment path for root to execute the CLI without using complete paths (i.e., if installed to `/usr/sbin/`, you can type **`tw_cli`** on the command line, otherwise you will have to type the complete path: **`/home/user/tw_cli`**

The 3ware CLI comes in both 32-bit and 64-bit versions. Be sure to copy the correct version for the version of the operating system you are using.

Working with 3ware CLI

You can work with the 3ware CLI in different ways:

- Interactively, entering commands at the main prompt
- As a series of single commands
- By creating a script—an input file with multiple commands

This first section shows examples of each of these ways.

Examples shown in the *CLI Reference* chapters reflect the interactive method.

There are two command syntaxes available for the 3ware CLI:

- **The primary command syntax.**

  The current CLI version of the 3ware CLI includes a new command syntax to improve usability. This is now considered the primary syntax for use in the 3ware CLI, and includes commands for features new in the 3ware RAID controller software version 9.1.5, such as those that are used with the Battery Backup Unit (BBU).

  Details of the primary syntax are described under "Primary CLI Syntax Reference" on page 17.

- **Legacy command syntax.**

  In the current CLI version, the command syntax used in previous versions of the 3ware CLI is still supported, to make sure that scripts written with the old syntax will still operate. New functions (such as BBU-related commands) are not available in the legacy syntax. The legacy command syntax will be supported for a limited time.

  Details of the legacy syntax are described under "Legacy CLI Syntax Reference" on page 75.

Using the command interface interactively

You can use 3ware CLI interactively, entering commands at the main prompt and observing the results on the screen.

To use the CLI interactively

1 Enter the following command:

```
# tw_cli
```

The main prompt is displayed, indicating that the program is awaiting a command.

```
//localhost>
```

2 At the CLI prompt, you can enter commands to show or act on 3ware controllers, units, and drives.

For example,
```
//localhost> show
```

displays all controllers in the system and shows details about them, like this:
```
Ctl  Model   Ports  Drives Units NotOpt RRate  VRate  BBU
------------------------------------------------------------
c0   7500-12  12     8      3    1      2       -       -
c1   9506S-12 12     6      1    0      3       5  TESTING
```

Using a single command with output

You can use 3ware CLI with line arguments, processing a single command at a time. To do so, simply enter the command and the arguments.

Single commands can be useful when you want to perform a task such as redirecting the output of the command to a file. It also allows you to use the command line history to eliminate some typing.

Syntax

```
tw_cli <command line arguments>
```

Example

```
tw_cli /c0 show diag > /tmp/3w_diag.out
```

Using an input file to execute a script

You can operate 3ware CLI scripts by executing a file. The file is a text file containing a list of CLI commands which you have entered in advance. Each command must be on a separate line.

Syntax

```
tw_cli -f <filename>
```

Where `<filename>` is the name of the text file you want to execute.

Example

```
tw_cli -f clicommand.txt
```

This example executes the file `clicommand.txt`, and runs the CLI commands included in that file.

Scripting example

Following is a a scripting example using a text file called config_array.txt, containing three commands. This example sets up a 12-port controller with two units: one with the first 2 drives mirrored, and another with the remaining drives in a RAID 5 array. In then prints the configurations for verification. The commands included in the script file are:

```
/c0 add type=raid1 disk=0-1
/c0 add type=raid5 disk=2-11
/c0 show
```

To run the script, enter:

```
tw_cli -f config_array.txt
```

Outputting the CLI to a Text File

You can have the output of the 3ware CLI, including errors, sent to a text file by adding 2>&1 to the end of the line. This could be useful, for example, if you want to email the output to AMCC Technical Support.

Examples

tw_cli /c2/p0 show >> controller2port0info.txt 2>&1

or

tw_cli /c0 show diag >> Logfile.txt 2>&1

Understanding RAID Concepts and Levels

The next few pages introduce RAID concepts you may find useful. For additional information about installing and managing your 3ware controller, see the *3ware 9000 Series Serial ATA RAID Controller User Guide*.

3ware controllers use a Redundant Array of Inexpensive Disks (RAID) to increase your storage system's performance and provide fault tolerance (protection against data loss).

The following concepts are important to understand when working with a RAID controller:

- **Arrays and Units**. In the storage industry, the term "array" is used to describe two or more disk drives that appear to the operating system as a single unit. When you work with 3ware software, "unit" is the term used to refer to an array of disks that is configured and managed through the 3ware software. Single-disk units can also be configured in the 3ware software.
- **Mirroring**. Mirrored arrays write data to paired drives simultaneously. If one drive fails, the data is preserved on the paired drive. Mirroring provides data protection through redundancy. In addition, mirroring using a 3ware controller provides improved performance because 3ware's TwinStor technology reads from both drives simultaneously.
- **Striping**. Striping across disks allows data to be written and accessed on more than one drive, at the same time. Striping combines each drive's capacity into one large volume. Striped disk arrays achieve highest transfer rates and performance at the expense of fault tolerance.
- **Distributed Parity**. Parity works in combination with striping on RAID 5 and RAID 50. Parity information is written to each of the striped drives, in rotation. Should a failure occur, the data on the failed drive can be reconstructed from the data on the other drives.
- **Hot Swap**. The process of swapping out a drive without having to shut down the system. This is useful when you need to swap out a degraded drive, manually or automatically, with a pre-designated spare.
- **Array Roaming.** The process of swapping out or swapping in a configured unit without having to shut down the system. This is useful if you need to move the unit to another controller.
- **Disk Roaming.** The process of removing a unit from a controller and putting it back later, either on the same controller, or a different one, and having it recognized as a unit. The disks may be in a different order than they initially occupied, without harm to the data. The disks may be attached to the same ports or different ports on the controller.

RAID Configurations

The following RAID levels and configurations are available for drives attached to a 3ware controller:

- RAID 0
- RAID 1
- RAID 5
- RAID 10
- RAID 50
- Single Disk
- JBOD
- Hot Spare

RAID 0

Provides striping, but no mirroring. Striped disk arrays achieve high transfer rates because they can read and write data on more than one drive simultaneously. The stripe size is configurable in the 3ware CLI, 3ware BIOS Manager (3BM) and in the 3ware Disk Manager (3DM 2). Requires a minimum of two drives.

When drives are configured in a striped disk array (see Figure 1), large files are distributed across the multiple disks using RAID 0 techniques.

Striped disk arrays give exceptional performance, particularly for data intensive applications such as video editing, computer aided design and geographical information systems.

RAID 0 arrays are not fault tolerant. The loss of any drive results in the loss of all the data in that array, and can even cause a system hang, depending on your operating system. RAID 0 arrays are not recommended for high availability systems unless additional precautions are taken to prevent system hangs and data loss.

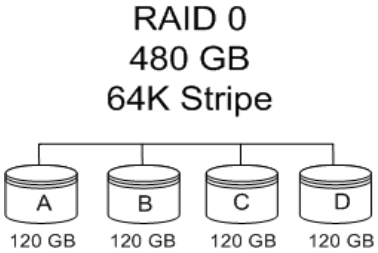

Figure 1. RAID 0 Configuration Example

RAID 1

Also known as a mirrored array. Mirroring is done on pairs of drives. Mirrored disk arrays write data to two drives using RAID 1 algorithms (see

Figure 2). This gives your system fault tolerance by preserving the data on one drive if the other drive fails. Fault tolerance is a basic requirement for mission critical systems like web and database servers.

3ware uses a patented technology, TwinStor®, on RAID 1 arrays for improved performance during sequential read operations. With TwinStor technology, read performance is twice the speed of a single drive during sequential read operation.

The adaptive algorithms in TwinStor technology boost performance by distinguishing between random and sequential read requests. For the sequential requests generated when accessing large files, both drives are used, with the heads simultaneously reading alternating sections of the file. For the smaller random transactions, the data is read from a single optimal drive head.

Figure 2. RAID 1 Configuration Example

RAID 5

Combines striping data with parity (exclusive OR) to restore data in case of a drive failure. This array type provides performance, fault tolerance, high capacity, and storage efficiency. Requires a minimum of three drives.

Parity information is distributed across all drives rather than being concentrated on a single disk (see Figure 3). This avoids throughput loss due to contention for the parity drive.

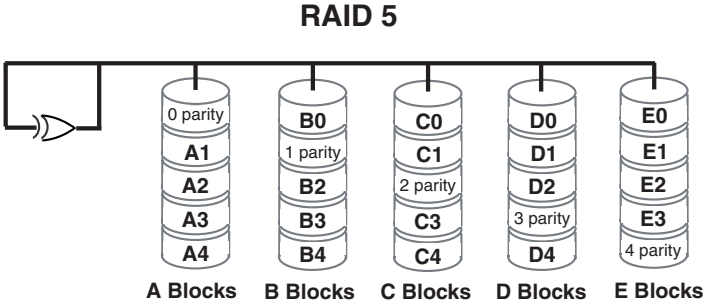

Figure 3. RAID 5 Configuration Example

RAID 10

This array is a combination of RAID 1 with RAID 0. Striped and mirrored arrays for fault tolerance and high performance. Requires a minimum of four drives to use both RAID 0 and RAID 1 techniques.

When drives are configured as a striped mirrored array, the disks are configured using both RAID 0 and RAID 1 techniques, thus the name RAID 10 (see Figure 4). A minimum of four drives are required to use this technique. The first two drives are mirrored as a fault tolerant array using RAID 1. The third and fourth drives are mirrored as a second fault tolerant array using RAID 1. The two mirrored arrays are then grouped as a striped RAID 0 array using a two tier structure. Higher data transfer rates are achieved by leveraging TwinStor and striping the arrays. RAID 10 is available on the four, eight, and twelve port 3ware Serial ATA RAID Controllers.

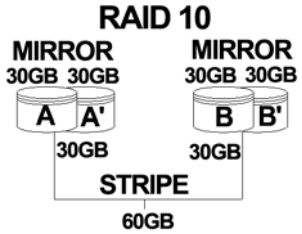

Figure 4. RAID 10 Configuration Example

RAID 50

This array is a combination of RAID 5 with RAID 0. This array type provides fault tolerance and high performance. Requires a minimum of six drives.

Several combinations are available with RAID 50. For example, on a 12-port controller, you can have a grouping of 3, 4, or 6 drives. A grouping of 3 means that the RAID 5 arrays used have 3 disks each; four of these 3-drive RAID 5 arrays are striped together to form the 12-drive RAID 50 array.

Single Disk

A single drive that has been configured as a unit through 3ware software (3BM, 3DM2, or CLI). Like disks in other RAID configurations, single disks contain 3ware Disk Control Block (DCB) information and are seen by the OS as available units.

Single drives are not fault tolerant and therefore not recommended for high availability systems unless additional precautions are take to prevent system hangs and data loss.

JBOD

A JBOD is an unconfigured disk attached to your 3ware RAID controller. 3ware recommends that you use Single Disk as a replacement for JBOD, to take advantage of future advanced features such as RAID level migration (RLM).

JBOD units are not fault tolerant and therefore not recommended for high availability systems unless additional precautions are taken to prevent system hangs and data loss.

Hot Spare

A single configured drive, available so that a redundant array can be automatically rebuilt in case of drive failure.

For additional information about RAID levels, see the article “RAID Primer” on the 3ware website, at: http://www.3ware.com/products/pdf/RAID_Primer.pdf.

Determining What RAID Level to Use

Select the RAID configuration to use based on the applications to be used on the system, whether performance or data protection is of primary importance, and the number of disk drives available for use.

Review the information under “Understanding RAID Concepts and Levels” on page 10 to determine the type of RAID configuration most appropriate for your needs and use the tables below to determine what RAID levels are available, based on your particular controller model and the number of available drives.

The RAID configurations available to you are determined by the number of ports on your controller, and the number drives attached to those ports. You can configure all drives in one unit, or you can configure multiple units, if you have enough drives.

Table 2: Possible Configurations Based on Number of Drives

# Drives	Possible RAID Configurations
1	Single or spare drive
2	RAID 0 or RAID 1
3	RAID 0, RAID 5, or RAID 1 + spare
4	RAID 5 + hot spare RAID 10 Combination of RAID 0, RAID 1, single disk
5	RAID 5 + hot spare RAID 10 + hot spare Combination of RAID 0, RAID 1, hot spare
6 or more	RAID 50 Depending on the number of drives, a RAID 50 may contain from 2 to 4 subunits. For example, with 12 drives, possible RAID 50 configurations include 2 subunits of 6, 3 subunits of 4, or 4 subunits of 3. With 10 drives, a RAID 50 will contain 2 subunits of 5 drives each. Combination of RAID 0, 1, 5, 10, hot spare, and single disk

Drive Capacity Considerations

The capacity of each drive is limited to the capacity of the smallest drive in the array. The total array capacity is defined as follows:

Table 3: Drive Capacity

RAID Level	Capacity
RAID 0	(number of drives) X (capacity of the smallest drive)
RAID 1	capacity of the smallest drive
RAID 5	(number of drives - 1) X (capacity of the smallest drive) Storage efficiency increases with the number of disks: storage efficiency = (number of drives -1) / (number of drives)
RAID 10	(number of drives / 2) X (capacity of smallest drive)
RAID 50	(number of drives - number of groups of drives) X (capacity of the smallest drive)

Support for Over 2 Terabytes

Windows 2000, Windows XP, and Linux 2.4 do not currently recognize unit capacity in excess of 2 TB.

If the combined capacity of the drives to be connected to a unit exceeds 2 Terabytes (TB), you can enable auto-carving using the `set autocarve` command.

Auto-carving divides the available unit capacity into multiple chunks of 2 TB or smaller that can be addressed by the operating systems as separate volumes.

Primary CLI Syntax Reference

Note: Information contained in this document that describes usage only for the 3ware 9000 series products does not work with 3ware 7000 or 8000 series controllers.

This chapter provides detailed information about using the primary command syntax for the 3ware CLI. (The legacy syntax is still supported for a limited time. For details see, “Legacy CLI Syntax Reference” on page 75.)

Throughout this chapter the examples reflect the interactive method of executing 3ware CLI.

Common Tasks Mapped to CLI Commands

The table below lists many of the tasks people use to manage their RAID controllers and units, and lists the primary CLI command associated with those tasks.

Table 4: Common Tasks Mapped to CLI Commands

Task	CLI Command	Page
Controller Configuration Tasks		
View information about different controllers	/cx show	26
View controller policies	/cx show [attribute] [attribute]	27

Table 4: Common Tasks Mapped to CLI Commands

Task	CLI Command	Page
Set policies for a controller ■ Export JBODs ■ Enable staggered spinup ■ Disable write cache on unit degrade ■ Enable autocarving	 /cx set exportjbod /cx set stagger and /cx set spinup /cx set ondegrade /cx set autocarve	 46 47 46 47
Unit Configuration Tasks		
Create a new unit	/cx add	33
Create a hot spare	/cx add	33
Enable/disable unit write cache	/cx/ux set cache	55
Changes to an Existing Configuration		
Change RAID level	/cx/ux migrate	56
Change stripe size	/cx/ux migrate	56
Expand unit capacity	/cx/ux migrate	56
Delete a unit	/cx/ux del	53
Remove a unit (export)	/cx/ux export	52
Name a unit	/cx/ux set name	56
Units Mainenance Tasks		
Add a time slot to a rebuild schedule	/cx add rebuild	42
Add a time slot to a verify schedule	/cx add verify	43
Add a time slot to a selftest schedule	/cx add selftest	44
Enable/disable the rebuild/migrate schedule and set the task rate	/cx set rebuild	45
Enable/disable the verify schedule and set the task rate	/cx set verify	46
Enable/disable the selftest schedule	/cx set selftest	46
Start a rebuild	/cx/ux start rebuild	53
Start a verify	/cx/ux start verify	54
Pause/resume rebuild	/cx/ux pause rebuild and /cx/ux resume rebuild	54

Table 4: Common Tasks Mapped to CLI Commands

Task	CLI Command	Page
Stop verify	/cx/ux stop verify	54
Enable/disable autoverify	/cx/ux set autoverify	55
Allow/disallow write caching	/cx/ux set cache	55
View Alarms	/cx show alarms	37

Primary Syntax Overview

The primary command syntax uses the general form:
`Object Command Attributes`

Objects are shell commands, controllers, units, ports (drives), and BBUs (battery backup units).

Commands can either select (show, get, present, read) attributes or alter (add, change, set, alter, write) attributes.

Attributes are either Boolean Attributes or Name-Value Attributes.

- The value of a boolean attribute is deduced by presence or lack of—that is, the attribute is either specified, or not. For example, the command *show alarms* by default lists alarms with the most recent alarm first. If you include the attribute *reverse* (*show alarms reverse*), alarms are listed in reverse order.
- The value of *name*-value attributes are expressed in the format *attribute=value*.

Example: When adding (creating) a unit to the controller with the following command string,
`/c1 add type=raid1 disk=0-1`

`c1` is the object, `add` is the command, `type` (for type of array) is an attribute with `raid1` is the value of the attribute, and `disk` is another attribute with `0-1` as the value (ports 0 through 1).

Information about commands is organized by the object on which commands act:

Shell Object Commands. Shell object commands set the focus or provide information (such as alarms, diagnostics, rebuild schedules, and so forth) about all controllers in the system. For details, see “Shell Object Commands” on page 21.

Controller Object Commands. Controller object commands provide information and perform actions related to a specific controller. For example, you use controller object commands for such tasks as seeing alarms specific to a controller, creating schedules during which background tasks are run, and

setting policies for the controller. You also use the controller object command `/cx add type` to create RAID arrays. For details, see “Controller Object Commands” on page 25.

Unit Object Commands. Unit object commands provide information and perform actions related to a specific unit on a specific controller. For example, you use unit object commands for such tasks as seeing the rebuild, verify, or initialize status of a unit, starting, stopping, and resuming verifies, starting and stopping rebuilds, and setting policies for the unit. You also use the controller object command `cx/ux migrate type` to change the configuration of a RAID array. For details, see “Unit Object Commands” on page 49.

Port Object Commands. Port object commands provide information and perform actions related to a drive on a specific port. For example, you use port object commands for such tasks as seeing the status, model, or serial number of the drive. For details, see “Port Object Commands” on page 61.

BBU Object Commands. BBU object commands provide information and perform actions related to a Battery Backup Unit on a specific controller. For details, see “BBU Object Commands” on page 64.

Help Commands. Help commands allow you to display help information for all commands and attributes. For details, see “Help Commands” on page 68.

Conventions

The following conventions are used through this guide:

- In text, `monospace font` is used for code and for things you type.
- In commands, an italic font indicates items that are variable, but that you must specify, such as a controller ID, or a unit ID.
- In commands, brackets around an item indicates that it is optional.
- In commands, ellipses (. . .) indicate that more than one parameter can be included.
- In commands, a vertical bar (|) indicates an 'or' situation where the user has a choice between more than one attribute, but only one can be specified.

For example, in the command to rescan all ports and reconstitute all units, the syntax appears as rescan [*cid* ...] [noscan]. The italic cid indicates that you need to supply a controller ID. The ellipses indicate that you can specify more than one controller ID, separated by spaces. The brackets indicate that you may omit the controller ID, to rescan all controllers, and the noscan parameter, so that the operation will be reported to the operating system.

Shell Object Commands

Shell object commands are either applicable to all the controllers in the system (such as show, rescan, flush, commit), or redirect the focused object.

Syntax

```
focus Object
show [attribute [modifier]]
    ver
    alarms [reverse]
    diag
    rebuild
    verify
    selftest
rescan
flush
commit
```

focus *Object*

The focus command is active in interactive mode only and is provided to reduce typing.

The focus command will set the specified object in focus and change the prompt to reflect this. This allows you to enter a command that applies to the focus, instead of having to type the entire object name each time.

For example, where normally you might type:

```
//hostname/c0/u0 show
```

if you set the focus to `//hostname/c0/u0`, the prompt changes to reflect that, and you only have to type `show`. The concept is similar to being in a particular location in a filesystem and requesting a listing of the current directory.

object can have the following forms:

`//hostname/cx/ux` specifies the fully qualified URI (Universal Resource Identifier) of an object on host `hostname`, controller `cx`, unit `ux`.

`//hostname` specifies the root of host `hostname`.

`..` specifies one level up (the parent object).

`/` specifies the root at the current focused hostname.

./obj specifies the next level of the object.

`/c0/bbu` specifies a relative path with respect to the current focused hostname.

For example:

```
//localhost> focus //elvis.amcc.com
//elvis.amcc.com>
```

```
//elvis.amcc.com> focus /c0/u0
//elvis.amcc.com/c0/u0>

//elvis.amcc.com/c0/u0> focus ..
//elvis.amcc.com/c0>

//elvis.amcc.com/c0/> focus u0
//elvis.amcc.com/c0/u0>

//elvis.amcc.com/c0> focus /
//elvis.amcc.com>
```

The focus command is available by default. You can disable focus by setting TW_CLI_INPUT_STYLE to *old*. (See "Environment Variables" on page 72.)

show

This command shows a general summary of all detected controllers.

Note that the device drivers for the appropriate operating system should be loaded for the list to show all controllers. The intention is to provide a global view of the environment.

Typical output of the Show command looks like:

```
//localhost> show

Ctl   Model      Ports   Drives  Units   NotOpt   RRate   VRate   BBU
------------------------------------------------------------------------
c0    7500-12     12      8       3       1        2       -       -
c1    9506S-12    12      6       1       0        3       5       TESTING
```

Indicating that Controller 0 is a 7500 model with 12 Ports, with 8 Drives detected (attached), total of 3 Units, with one unit in a NotOpt (Not Optimal) state, RRate (Rebuild Rate) of 2, VRate (Verify Rate) of '-' (Not Applicable), BBU of '-' (Not Applicable). Not Optimal refers to any state except OK and VERIFYING. Other states include VERIFY-PAUSED, INITIALIZING, INIT-PAUSED, REBUILDING, REBUILD-PAUSED, DEGRADED, MIGRATING, MIGRATE-PAUSED, RECOVERY, INOPERABLE, and UNKNOWN.

RRate also applies to initializing, migrating, and recovery background tasks.

show ver

This command will show the CLI and API version.

Example:

```
//localhost> show ver
CLI Version = 2.00.00.0xx
API Version = 2.00.00.0xx
```

where xx is the actual version. See the Release Notes for details.

show alarms [reverse]

This command shows the alarms or AEN messages of all controllers in the system. The default is to display the most recent message first. The reverse attribute displays the most recent message last.

show diag

This command shows the diagnostic information of all controllers in the system.

show rebuild

This command displays all rebuild schedules for the 9000 controllers in the system.

The rebuild rate is also applicable for initializing, migrating, and recovery background tasks.

Example:

```
//localhost> show rebuild

Rebuild Schedule for Controller /c0
=======================================================
Slot   Day       Hour      Duration     Status
-------------------------------------------------------
1      Sun       12:00am   24 hr(s)     disabled
2      Mon       12:00am   24 hr(s)     disabled
3      Tue       12:00am   24 hr(s)     disabled
4      Wed       12:00am   24 hr(s)     disabled
5      Thu       12:00am   24 hr(s)     disabled
6      Fri       12:00am   24 hr(s)     disabled
7      Sat       12:00am   24 hr(s)     disabled
```

For additional information about rebuild schedules, see “/cx add rebuild=ddd:hh:duration” on page 42, and see the discussion of background tasks and schedules in *3ware 9000 Series Serial ATA RAID Controller User Guide*.

show verify

This command displays all verify schedules for the 9000 controllers in the system.

Example:

```
//localhost> show verify
```

```
Verify Schedule for Controller /c0
=========================================================
Slot   Day       Hour        Duration    Status
---------------------------------------------------------
1      Sat       11:00pm      4 hr(s)    enabled
2      -         -                       enabled
3      -         -                       enabled
4      -         -                       enabled
5      -         -                       enabled
6      -         -                       enabled
7      -         -                       enabled
```

For additional information about verify schedules, see “/cx add verify=ddd:hh:duration” on page 43, and see the discussion of background tasks and schedules in *3ware 9000 Series Serial ATA RAID Controller User Guide*.

show selftest

This command displays all selftest schedules for the 9000 controllers in the system.

Example:

```
//localhost> show selftest

Selftest Schedule for Controller /c0
=========================================================
Slot   Day       Hour        UDMA        SMART
---------------------------------------------------------
1      Sun       12:00am     enabled     enabled
2      Mon       12:00am     enabled     enabled
3      Tue       12:00am     enabled     enabled
4      Wed       12:00am     enabled     enabled
5      Thu       12:00am     enabled     enabled
6      Fri       12:00am     enabled     enabled
7      Sat       12:00am     enabled     enabled
```

For additional information about selftest schedules, see “/cx add selftest=ddd:hh” on page 44, and see the discussion of background tasks and schedules in *3ware 9000 Series Serial ATA RAID Controller User Guide*.

flush

This command sends a flush command to all 3ware controllers in the system. For more information, see “/cx flush” on page 37.

rescan

This command sends a rescan command to all 3ware controllers in the system. For more information, see “/cx rescan [noscan]” on page 36.

commit

This command sends a commit command to all 3ware controllers in the system. For more information, see "/cx commit" on page 37.

Controller Object Commands

Controller object commands provide information and perform actions related to a specific controller, such as `/c0`. For example, you use controller object commands for such tasks as seeing alarms specific to a controller, creating schedules during which background tasks are run, and setting policies for the controller. You also use the controller object command `/cx add type` to create RAID arrays.

Syntax

```
/cx show
/cx show attribute [attribute ...]     where attributes are:
   driver|model|firmware|memory|bios|monitor|serial
   |pcb|pchip|achip|numports|numunits|numdrives|
   unitstatus|drivestatus|allunitstatus|exportjbod|
   ondegrade|spinup|autocarve|stagger
/cx show all
/cx show diag
/cx show alarms [reverse]
/cx show rebuild  (9000 only)
/cx show verify  (9000 only)
/cx show selftest  (9000 only)

/cx add type=<RaidType> disk=<p:-p..> [stripe=<Stripe>]
   [noscan] [nocache] [group=<3|4|5|6>] [autoverify] [ignor-
   eECC] RaidType = { raid0, raid1, raid5, raid10, raid50,
   single, spare, jbod (7000/8000 only)} [name=string (9000
   only)]
/cx add rebuild=ddd:hh:duration     (9000 only)
/cx add verify=ddd:hh:duration      (9000 only)
/cx add selftest=ddd:hh             (9000 only)

/cx del rebuild=slot_id             (9000 only)
/cx del verify=slot_id              (9000 only)
/cx del selftest=slot_id            (9000 only)

/cx set exportjbod=on|off           (9000 only)
/cx set ondegrade=cacheoff|follow   (9000 only)
/cx set spinup=nn                   (9000 only)
/cx set stagger=nn                  (9000 only)
/cx set autocarve=on|off          (9000 only)

/cx set rebuild=enable|disable|<1..5>
```

```
                  (enable|disable for 9000 only)
/cx set verify=enable|disable|<1..5>
                  (enable|disable for 9000 only)
/cx set selftest=enable|disable [task=UDMA|SMART] 9000 only)
/cx flush
/cx commit                          (Windows only)

/cx start mediascan             (7000/8000 only)

/cx stop mediascan              (7000/8000 only)

/cx rescan [noscan]
   NOTE: Does not import non-JBOD on 7000/8000 models.
```

/cx show

This command shows summary information on the specified controller */cx*. This information is organized into a report containing two to three parts:

- A **Unit** summary listing all present units
- A **Port** summary section listing of all ports and disks attached to them.
- A **BBU** summary section listing, if a BBU unit is installed on the controller.

The **Unit** summary section lists all present unit and specifies their unit number, unit type (such RAID 5), status, size (usable capacity) in gigabytes or terabytes, number of blocks, and unit status (OK, RECOVERY, INOPERABLE, UNKNOWN, DEGRADED, INITIALIZING, INIT-PAUSED, VERIFYING, VERIFY-PAUSED, REBUILDING, REBUILD-PAUSED, MIGRATING, MIGRATE-PAUSED).

%Cmpl reports the percent completion of REBUILDING, VERIFYING, INITIALIZING, or MIGRATING units.

The **Port** summary section lists all present ports and specifies the port number, disk status, unit affiliation, size (in gigabytes) and blocks (512 bytes), and the serial number assigned by the disk vendor.

The **BBU** summary lists details about the BBU, if one is installed. It includes a few important attributes such as hours left (in which the current BBU can backup the controller cache in the event of power loss), temperature, voltage, readiness, and so forth.

Additional attributes about controllers, units, ports and disks can be obtained by querying for them explicitly. For details, see the other show sub-commands below.

Typical output looks like:

```
//localhost> /c0 show
Unit   UnitType       Status         %Cmpl   Stripe  Size(GB) Cache  AVerify  IgnECC
------------------------------------------------------------------------------------
u0     RAID-1         OK             -       -       149.05   ON     OFF      OFF
u1     RAID-5         OK             -       64k     298.22   ON     OFF      OFF
u2     SPARE          OK             -       -       149.05   ON     OFF      -


Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
p0     OK               u0     149.05 GB   312581808     3JS0TF14
p1     OK               u0     149.05 GB   312581808     3JS0TETZ
p2     OK               u1     149.05 GB   312581808     3JS0VG85
p3     OK               u1     149.05 GB   312581808     3JS0VGCY
p4     OK               u1     149.05 GB   312581808     3JS0VGGQ
p5     OK               u2     149.05 GB   312581808     3JS0VH1P
p6     OK               -      149.05 GB   312581808     3JS0TF0P
p7     OK               -      149.05 GB   312581808     3JS0VF43
p8     OK               -      149.05 GB   312581808     3JS0VG8D
p9     NOT-PRESENT      -      -           -             -
p10    NOT-PRESENT      -      -           -             -
p11    NOT-PRESENT      -      -           -             -
Name  OnlineState  BBUReady  Status    Volt     Temp     Hours  LastCapTest
---------------------------------------------------------------------------
bbu   On           Yes       OK        OK       OK       241    22-Jun-2004
```

/cx show *attribute* [*attribute* ...]

This command shows the current setting of the specified attributes on the specified controller. One or many attributes can be specified. Specifying an invalid attribute will terminate the loop. Possible attributes are: driver, model, firmware, memory, bios, monitor, serial, pcb, pchip, achip, numports, numunits, numdrives, unitstatus, drivestatus, allunitstatus, exportjbod, ondegrade, spinup, autocarve, and stagger.

Example: To see the driver and firmware installed on controller 0, enter the following:

```
//localhost> /c0 show driver firmware
/c0 Driver Version = 2.x
/c0 Firmware Version  = FE9X 2.x

Where x=actual version
```

/cx show driver

This command reports the device driver version associated with controller /cx.

Example:

```
//localhost> /c0 show driver
/c0 Driver Version = 2.x
```

/cx show model

This command reports the controller model of controller /cx.

Example:

```
//localhost> /c0 show model
/c0 Model = 9500-x
```

/cx show firmware

This command reports the firmware version of controller /cx.

Example:

```
//localhost> /c0 show firmware
/c0 Firmware Version = FGXX 2.x
```

/cx show bios

This command reports the BIOS version of controller /cx.

Example:

```
//localhost> /c0 show bios
/c0 BIOS Version = BG9X 2.x
```

/cx show monitor

This command reports the monitor (firmware boot-loader) version of controller /cx.

Example:

```
//localhost> /c0 show monitor
/c0 Monitor Version = BLDR 2.x
```

/cx show serial

This command reports the serial number of the specified controller /cx.

Example:

```
//localhost> /c0 show serial
/c0 Serial Number = F12705A3240009
```

/cx show pcb

This command reports the PCB (Printed Circuit Board) version of the specified controller /cx.

Example:

```
//localhost> /c0 show pcb
/c0 PCB Version = RevX
```

/cx show pchip

This command reports the PCHIP (PCI Interface Chip) version of the specified controller /cx.

Example:

```
//localhost> /c0 show pchip
/c0 PCHIP Version = 1.x
```

/cx show achip

This command reports the ACHIP (ATA Interface Chip) version of the specified controller /cx.

Example:

```
//localhost> /c0 show achip
/c0 ACHIP Version = 3.x
```

/cx show numports

This command reports the port capacity (number of physical ports) of the specified controller /cx.

Example:

```
//localhost> /c0 show numports
/c0 Number of Ports = 12
```

/cx show numunits

This command reports the number of units currently managed by the specified controller /cx. This report does not include off-line units (or exported units).

Example:

```
//localhost> /c0 show numunits
/c0 Number of Units = 1
```

/cx show numdrives

This command reports the number of drives currently managed by the specified controller /cx. This report does not include (logically) removed or exported drives. Also note that physically removed disk(s) will not be detected unless I/O is performed against the disk. See “/cx/px show smart” on page 62 for a workaround.

Example:

```
//localhost> /c0 show numdrives
/c0 Number of Drives = 5
```

/cx show exportjbod

This feature only applies to 9000 series controllers.

This command reports the current JBOD Export Policy: “on”, “off” or “Not Supported.”

Example:

```
//localhost> /c0 show exportjbod
/c0 JBOD Export Policy = Not Supported.
```

//localhost> /c1 show exportjbod

```
/c1 JBOD Export Policy = on
```

/cx show spinup

This feature only applies to 9000 series controllers.

This command reports the number of concurrent disks that will spin up when the system is powered on, after waiting for the number of seconds specified with the `set stagger` command.

Example:

```
//localhost> /c0 show spinup
/c0 Disk Spinup Policy = 1
```

/cx show stagger

This feature only applies to 9000 series controllers.

This command reports the time delay between each group of spinups at the power on.

Example:

```
//localhost> /c0 show stagger
/c0 Spinup Stagger Time Policy (sec) = 2
```

/cx show ondegrade

This feature only applies to 9000 series controllers.

This command reports the cache policy for degraded units. If the ondegrade policy is “Follow Unit Policy,” a unit cache policy stays the same when the unit becomes degraded. If the ondegrade policy is off, a unit cache policy will be forced to “off” when the unit becomes degraded.

Example:

```
//localhost> /c0 show ondegrade
/c0 Cache on Degraded Policy = Follow Unit Policy
```

/cx show autocarve

This feature only applies to 9000 series controllers.

This command reports the Over 2TB Auto-Carve policy. If the policy is on, all newly created or migrated units which contain over 2TB of storage will be automatically carved into multiples of 2TB volumes plus one remainder volume. If the policy is off, all newly created or migrated units will be in single volume.

For operating systems that support disk drives over 2 TB of storage per unit, there is no need to set the policy to "on" unless you want the operating system to have multiple smaller volumes.

Example:

```
//localhost> /c0 show autocarve
/c0 Over 2TB Auto-Carving Policy = on
```

/cx show memory

This command presents the size of the memory installed on the controller.

Note: The 9500S ships with 128 MBytes of cache, yet only 112MB shows as memory installed. The other 16 MB is reserved for use by the controller.

Example:

```
//localhost> /c0 show memory
/c0 Memory Installed = 112MB
```

/cx show unitstatus

This command presents a list of units currently managed by the specified controller /cx, and shows their types, capacity, and status.

Possible statuses include: OK, VERIFYING, VERIFY-PAUSED, INITIALIZING, INIT-PAUSED, REBUILDING, REBUILD-PAUSED, DEGRADED, MIGRATING, MIGRATE-PAUSED, RECOVERY, INOPERABLE, and UNKNOWN.

Example:

```
//localhost> /c0 show unitstatus

Unit  UnitType  Status         %Cmpl  Stripe  Size(GB)  Cache  AVerify  IgnECC
------------------------------------------------------------------------------
u0    RAID-5    OK             -      64K     223.485   OFF    OFF      ON
u1    JBOD      OK             -      -       76.3352   OFF    OFF      -
u2    RAID-0    OK             -      64K     148.99    ON     ON       -
```

/cx show allunitstatus

This command presents a count of total and NotOptimal units managed by the specified controller /cx. For more about the meaning of NotOptimal, on See “Shell Object Commands” on page 21.

Example:

```
//localhost> /c0 show allunitstatus
/c0 Total Units = 2
/c0 NotOptimal Units = 0
```

/cx show drivestatus

This command reports a list of drives and their port assignment, status, the unit with which they are associated, their size in gigabytes and blocks, and the serial number assigned by the drive manufacturer.

Example:

```
//localhost> /c0 show drivestatus

Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
-
p0     OK               u0     149.05 GB   312581808     3JS0TF14
p1     OK               u0     149.05 GB   312581808     3JS0TETZ
p2     OK               u1     149.05 GB   312581808     3JS0VG85
p3     OK               u1     149.05 GB   312581808     3JS0VGCY
p4     OK               u1     149.05 GB   312581808     3JS0VGGQ
p5     OK               u2     149.05 GB   312581808     3JS0VH1P
p6     OK               -      149.05 GB   312581808     3JS0TF0P
p7     OK               -      149.05 GB   312581808     3JS0VF43
p8     OK               -      149.05 GB   312581808     3JS0VG8D
p9     NOT-PRESENT      -      -           -             -
p10    NOT-PRESENT      -      -           -             -
p11    NOT-PRESENT      -      -           -             -
```

/cx show all

This command shows the current setting of all of the following attributes on the specified controller: driver, model, firmware, memory, bios, monitor, serial, pcb, pchip, achip, numports, numunits, numdrives, unitstatus, drivestatus, allunitstatus, exportjbod, ondegrade, spinup, autocarve, and stagger.

Example: (where *x* represents the actual version number)

```
//localhost> /c0 show all
/c0 Driver Version = 2.x
/c0 Model = 9500S-12
/c0 Memory Installed  = 112MB
/c0 Firmware Version = FE9X 2.x
/c0 Bios Version = BE9X 2.x
/c0 Monitor Version = BL9X 2.x
/c0 Serial Number = xxxxx
/c0 PCB Version = Rev 0xx
/c0 PCHIP Version = 1.xx
/c0 ACHIP Version = 3.xx
/c0 Number of Ports = 12
/c0 Number of Units = 2
/c0 Number of Drives = 12
/c0 Total Units = 2
/c0 NotOptimal Units = 0

/c0 JBOD Export Policy = off
/c0 Disk Spinup Policy = 7
/c0 Spinup Stagger Time Policy (sec) = 4
/c0 Cache on Degrade Policy = Follow Unit Policy
/c0 Over 2TB Multi-Volume Policy = off

Unit  UnitType  Status         %Cmpl  Stripe  Size(GB)  Cache  AVerify  IgnECC
------------------------------------------------------------------------------
u0    RAID-5    OK             -      256K    148.99    ON     ON       ON
u1    RAID-5    OK             -      256K    595.961   ON     OFF      ON

Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
p0     OK               u0     74.53 GB    156301488     3JV3MV1C
p1     OK               u0     74.53 GB    156301488     3JV3MK6B
p2     OK               u0     74.53 GB    156301488     3JV3LW52
p3     OK               u1     74.53 GB    156301488     3JV49S77
p4     OK               u1     74.53 GB    156301488     3JV3MVTA
p5     OK               u1     74.53 GB    156301488     5JV980Z0
p6     OK               u1     74.53 GB    156301488     5JV9820G
p7     OK               u1     111.79 GB   234441648     WD-WMAEL10275
p8     OK               u1     111.79 GB   234441648     WD-WMAEL10274
p9     OK               u1     111.79 GB   234441648     WD-WMAEL10281
p10    OK               u1     111.79 GB   234441648     WD-WMAEL10273
p11    OK               u1     111.79 GB   234441648     WD-WMAEL10274

Name  OnlineState  BBUReady  Status    Volt     Temp     Hours  LastCapTest
---------------------------------------------------------------------------
bbu   On           Yes       OK        OK       OK       165    06-Nov-2004
```

/cx add type=<*RaidType*> disk=<*p:-p*> [stripe=Stripe] [noscan] [group=<*3/4/5/6*>] [nocache] [autoverify] [ignoreECC] [name=*string*]

This command allows you to create a new unit on the specified controller. You specify *type*, *disks*, and optional *stripe* size. By default the host operating system will be informed of the new block device and write cache will be enabled. In case of RAID 50, you can also specify the layout of the unit by specifying the number of disks per disk group with the *group* attribute.

/cx is the controller name, for example /c0, /c1, and so forth.

type=RaidType specifies the type of RAID unit to be created. Possible unit types include `raid0`, `raid1`, `raid5`, `raid10`, `raid50`, `single`, `spare`, and `JBOD`.

Example: `type=raid50`

When a new unit is created, it is automatically assigned a unique serial number. In addition, users can assign the unit a name.

Note: The unit's serial number cannot be changed.

The following table shows supported types and controller models.

Table 5: Supported RAID Types

Model	R0	R1	R5	R10	R50	Single	JBOD	Spare
7K/8K	Yes	Yes	Yes	Yes	No	No	Yes	Yes
9K	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

disk=*p:-p* consists of a list of ports (disks) to be used in the construction of the specified unit type. One or more ports can be specified. Multiple ports can be specified using a colon (:) or a dash (-) as port index separators. A dash indicates a range and can be mixed with colons. For example `disk=0:1:2-5:9:12` indicates port 0, 1, 2 through 5 (inclusive), 9 and 12.

stripe=*Stripe* consists of the stripe size to be used. The following table illustrates the supported and applicable stripes on unit types and controller models. Stripe size units are in K (kilobytes). If no stripe size is specified, 64K is used by default, if applicable. If you need to change the stripe size after the unit is created, you can do so my migrating the unit.

Table 6: Supported Stripe Sizes (KB)

Model	R0	R1	R5	R10	JBOD	Spare	R50	Single
7K/8K	64	N/A	64	64	N/A	N/A	N/S	N/S
	128			128				
	256			256				
	512			512				
	1024			1024				
9K	16	N/A	16	16		N/A	16	N/A
	64		64	64			64	
	256		256	256			256	

group=*3*|*4*|*5*|*6* indicates the number of disks per group for a RAID 50 type. (This attribute can only be used when type=raid50.) Recall that a RAID 50 is a multi-tier array. At the bottom-most layer, N number of disks per group are used to form the RAID 5 layer. These RAID 5 arrays are then integrated into a RAID 0. This attribute allows you to specify the number of disks in the RAID 5 level. Valid values are 3, 4, 5 and 6.

Note that a sufficient number of disks are required for a given pattern or disk group. For example, given 6 disks, specifying 3 will create two RAID 5 arrays. With 12 disks, specifying 3 will create four RAID 5 arrays under the RAID 0 level. With only 6 disks a grouping of 6 is not allowed, as you would basically be creating a RAID 5.

The default RAID 50 grouping varies, based on number of disks. For 6 and 9 disks, default grouping is 3. For 8 disks, the default grouping is 4. For 10 disks, the default grouping is 5, and for 12 disks, the default grouping is 4. In the case of 12, the disks could be grouped into groups of 3, 4, or 6 drives. A grouping of 4 is set by default as it provides the best of net capacity and performance.

noscan attribute instructs CLI not to notify the OS of the creation of the new unit. By default CLI will inform the OS. One application of this feature is to prevent the OS from creating block special devices such as /dev/sdb and /dev/sdc as some implementations might create naming fragmentation and a moving target.

nocache attribute instructs CLI to disable the write cache on the newly created unit. Enabling write cache increases write performance at the cost of potential data loss in case of sudden power loss (unless a BBU or UPS is installed). By default the cache is enabled. To avoid the possibility of data loss in the event of a sudden power loss, it is recommended not to set nocache unless there is a BBU (battery backup unit) or UPS (uninterruptable power supply) installed.

autoverify attribute enables the autoverify attribute on the unit that is to be created. For more details on this feature, see “/cx/ux set autoverify=on|off” on page 55. This feature is not supported on model 7000/8000. On model 9000, the JBOD autoverify attribute is not persistent (does not survive reboots).

ignoreECC attribute enables the ignoreECC/OverwriteECC attribute on the unit that is to be created. For more details on this feature, see “/cx/ux set ignoreECC=on|off” on page 55. The following table illustrates the supported Model-Unit Types. This table only applies to setting this feature at unit creation time. IgnoreECC only applies to redundant units. For the 7/8000

series, this setting is only applicable during rebuild; it is not applicable during creation.

Table 7: Supported Model-Unit Types for ignoreECC

Model	R-0	R-1	R-5	R-10	R-50	Single	JBOD	Spare
7K/8K	No	No	No	No	No	No	No	No
9K	No	Yes	Yes	Yes	Yes	No	No	No

name=string attribute allows you to name the new unit. The string can be up to 21 characters and cannot contain spaces. In order to use reserved characters (‘<‘, ‘>’, ‘!’, ‘&’, etc.) put double quotes (" ") around the name string. The name can be changed after the unit has been created. For more information, see *“/cx/ux set name=string” on page 56* and “/cx/ux show name” on page 51.

/cx rescan [noscan]

This command instructs the controller to rescan all ports and reconstitute all units. The controller will update its list of ports (attached disks), and attempts to read every DCB (Disk Configuration Block) in order to re-assemble its view and awareness of logical units. Any newly found unit(s) or drive(s) will be listed.

`noscan` is used to not inform the OS of the unit discovery. The default is to inform the OS.

Note: If you are adding new drives, add them physically before issuing the rescan commands. Hot swap carriers are required unless you first power-down the system to prevent system hangs and electrical damage.

Example:

```
//localhost> /c1 rescan
Rescanning controller /c1 for units and drives ...Done
Found following unit(s): [/c1/u3]
Found following drive(s): [/c1/p7, /c1/p8]
```

Note: Rescanning does not import non-JBOD on 7000/8000 models.

/cx commit

This command only applies to the Windows operating system if a faster shutdown method is needed when running certain database applications. Linux and FreeBSD file systems do not require this command since they have their own ways of notifying the controller to do clean up for shut down.

/cx flush

This command forces the controller to write all cached data to disk for the specified controller.

/cx show alarms [reverse]

Asynchronous events (also referred to as AENs or alarms) are originated by firmware and captured by their respective device drivers. These events reflect warning, debugging, and/or informative messages for the end user. These events are kept in a finite queue inside the kernel, awaiting extraction by user space programs such as CLI and/or 3DM.

The /cx show alarms command displays all available alarms on a given controller. The default is to display the most recent alarm or AEN message first. The user can also use the [reverse] attribute to display the most recent alarm or AEN message last.

Alarms generated on 7000/8000 controllers do not have dates, so you will see a '-' in the Date column. This means that it is not applicable. In addition, alarm messages on 7000/8000 controllers contain the severity in the message text, so the Severity column also shows a '-'.

Typical output looks like:

```
tw_cli> /c1 show alarms reverse
Ctl Date                      Severity Message
--------------------------------------------------------------------------------
c1  [Fri Nov 28 04:26:31 2003] ERROR   (0x04:0x0002): Degraded unit detected:unit=0, port=2
c1  [Fri Nov 28 06:13:54 2003] INFO    (0x04:0x000B): Rebuild started: unit=0
c1  [Fri Nov 28 06:30:35 2003] INFO    (0x04:0x003B): Background rebuild paused:unit=0
c1  [Fri Nov 28 06:33:00 2003] ERROR   (0x04:0x0002): Degraded unit detected:unit=0, port=0
c1  [Fri Nov 28 06:33:04 2003] ERROR   (0x04:0x0002): Degraded unit detected:unit=0, port=4
c1  [Fri Nov 28 06:33:46 2003] INFO    (0x04:0x000B): Rebuild started: unit=0
c1  [Fri Nov 28 06:37:58 2003] INFO    (0x04:0x000B): Rebuild started: unit=0
c1  [Fri Nov 28 07:51:34 2003] INFO    (0x04:0x0005): Background rebuild done:unit=0
c1  [Fri Nov 28 07:59:43 2003] INFO    (0x04:0x0005): Background rebuild done:unit=0
c1  [Mon Dec  1 02:26:12 2003] ERROR   (0x04:0x0002): Degraded unit detected:unit=0, port=3
```

/cx show diag

This command extracts controller diagnostics suitable for technical support usage. Note that some characters might not be printable or rendered correctly (human readable). It is recommended to save the output from this command to a file, where it can be communicated to technical support or further studied with Linux utilities such as od(1).

In order to redirect the output you must run the following command from a command line, not from within the tw_cli shell.

```
tw_cli /c0 show diag > diag.txt
```

/cx show rebuild

9000 series controllers support background tasks and allow you to schedule a regular time when they occur.

Rebuild is one of the supported background tasks. Migrate and initialize are other background tasks that follow the same schedule as rebuild. Other background tasks for which there are separate schedules are verify and selftest. For each background task, up to 7 time periods can be registered, known as slots 1 through 7. Each task schedule can be managed by a set of commands including **add**, **del**, **show** and **set** a task. Background task schedules have a slot id, start-day-time, duration, status attributes.

For details about setting up a schedule for background rebuild tasks, see “Setting Up a Rebuild Schedule” on page 42.

B<rebuild> activity attempts to (re)synchronize all members of redundant units such as RAID-1, RAID-10, RAID-5 and RAID-50. Rebuild can be started manually or automatically if a spare has been defined. Scheduled rebuilds will take place during the scheduled time slot, if enabled the schedules are enabled. For in depth information about rebuild and other background tasks, see “About Background Tasks” in the *3ware 9000 Series Serial ATA RAID Controller User Guide*.

The `show rebuild` command displays the current rebuild background task schedule as illustrated below.

```
//localhost> /c1 show rebuild
Rebuild Schedule for Controller /c1
=======================================================
Slot    Day     Hour            Duration        Status
-------------------------------------------------------
1       Mon     2:00pm          10 hr(s)        disabled
2       Thu     7:00pm          18 hr(s)        disabled
3       -       -               -               disabled
4       -       -               -               disabled
5       -       -               -               disabled
6       Mon     1:00am           4 hr(s)        disabled
7       Sun     12:00am          1 hr(s)        disabled
```

A status of “disabled” indicates that the task schedule is disabled. In this case, the controller will not use the defined schedule timeslots. If the rebuild command is entered manually, rebuilding will start within 10 to 15 minutes. It will begin automatically if a rebuild is needed and a proper spare drive is set up.

If the rebuild schedule is enabled while a rebuild process is underway, the rebuild will pause until a scheduled time slot.

Example:

If a unit is in the initialization state at noon on Wednesday and the rebuild schedule shown above is in use (with schedules disabled), you would see the following status using the show command:

```
//localhost>/c1 show
Unit  UnitType  Status      %Cmpl  Stripe  Size(GB)  Cache  AVerify  IgnECC
------------------------------------------------------------------------------
u0    RAID-5    INITIALIZING   0      64K     521.466   ON     OFF      OFF

Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
p0     NOT-PRESENT      -      -           -             -
p1     OK               u0     76.33 GB    160086528     Y2NXL7FE
p2     NOT-PRESENT      -      -           -             -
p3     OK               u0     76.33 GB    160086528     Y2NXLB9E
p4     NOT-PRESENT      -      -           -             -
p5     OK               u0     76.33 GB    160086528     Y2NXQPZE
p6     NOT-PRESENT      -      -           -             -
p7     OK               u0     76.33 GB    160086528     Y2NXM4VE
p8     OK               u0     74.53 GB    156301488     3JV3WTSE
p9     OK               u0     74.53 GB    156301488     3JV3WRHC
p10    OK               u0     74.53 GB    156301488     3JV3WQLQ
p11    OK               u0     74.53 GB    156301488     3JV3WQLZ

Name  OnlineState  BBUReady  Status    Volt     Temp     Hours  LastCapTest
---------------------------------------------------------------------------
bbu   On           Yes       OK        OK       OK       0      xx-xxx-xxxx
```

If you then enable the rebuild schedules, the unit initialization will be paused until the next scheduled time slot, as reflected in the examples below:

```
//localhost> /c1 set rebuild=enable
Enabling scheduled rebuilds on controller /c1 ...Done.

//localhost> /c1 show rebuild

Rebuild Schedule for Controller /c1
=======================================================
Slot    Day     Hour            Duration        Status
-------------------------------------------------------
1       Mon     2:00pm          10 hr(s)        enabled
2       Thu     7:00pm          18 hr(s)        enabled
3       -       -               -               -
4       -       -               -               -
5       -       -               -               -
6       Mon     1:00am           4 hr(s)        enabled
7       Sun     12:00am          1 hr(s)        enabled
```

```
//localhost> /c1 show

Unit  UnitType  Status     %Cmpl  Stripe  Size(GB)  Cache  AVerify  IgnECC
------------------------------------------------------------------------
u0    RAID-5    INIT-PAUSED  0      64K     521.466   ON     OFF      OFF

Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
p0     NOT-PRESENT      -      -           -             -
p1     OK               u0     76.33 GB    160086528     Y2NXL7FE
p2     NOT-PRESENT      -      -           -             -
p3     OK               u0     76.33 GB    160086528     Y2NXLB9E
p4     NOT-PRESENT      -      -           -             -
p5     OK               u0     76.33 GB    160086528     Y2NXQPZE
p6     NOT-PRESENT      -      -           -             -
p7     OK               u0     76.33 GB    160086528     Y2NXM4VE
p8     OK               u0     74.53 GB    156301488     3JV3WTSE
p9     OK               u0     74.53 GB    156301488     3JV3WRHC
p10    OK               u0     74.53 GB    156301488     3JV3WQLQ
p11    OK               u0     74.53 GB    156301488     3JV3WQLZ

Name  OnlineState  BBUReady  Status    Volt     Temp     Hours  LastCapTest
---------------------------------------------------------------------------
bbu   On           Yes       OK        OK       OK       0      xx-xxx-xxxx
```

/cx show verify

9000 series controllers support background tasks and allow you to schedule a regular time when they occur.

Verify is one of the supported background tasks. Rebuild and selfttest are other background tasks for which there are separate schedules. Migrate and initialize are additional background tasks that follow the same schedule as rebuild. For each background task, up to 7 time periods can be registered, known as slots 1 through 7. Each task schedule can be managed by a set of commands including **add**, **del**, **show** and **set** a task. Background task schedules have a slot id, start-day-time, duration, status attributes.

For details about setting up a schedule for background verify tasks, see “Setting Up a Verify Schedule” on page 43.

Verify activity verifies all units based on their unit type. Verifying RAID-1 involves checking that both drives contain the exact data. On RAID-5, the parity information is used to verify data integrity. RAID-10 and 50 are composite types and follow their respective array types. On 9000 series, non-redundant units such as RAID-0, JBOD, single, and spare, are also verified (by reading and reporting un-readable sectors). If any parity mismatches are found, the array will be automatically background initialized. (For information about the initialization process, see the *3ware 9000 Series Serial ATA RAID Controller User Guide*.)

The `show verify` command displays the current verify background task schedule as illustrated below.

```
//localhost> /c1 show verify
Verify Schedule for Controller /c1
=========================================================
Slot    Day     Hour            Duration        Status
---------------------------------------------------------
1       Mon     2:00am           4 hr(s)        disabled
2       -       -               -               disabled
3       Tue     12:00am         24 hr(s)        disabled
4       Wed     12:00am         24 hr(s)        disabled
5       Thu     12:00am         24 hr(s)        disabled
6       Fri     12:00am         24 hr(s)        disabled
7       Sat     12:00am         24 hr(s)        disabled
```

A status of “disabled” indicates that the controller will not use the defined schedule timeslots and will start verifying immediately (within 10 to 15 minutes), if the verify command is entered manually, or it will begin automatically if the autoverify option is set. Rebuilds, migrations, and initializations will take priority over verifies.

/cx show selftest

9000 series controllers support background tasks and allow you to schedule a regular time when they occur.

Selftest is one of the supported background tasks. Rebuild and verify are other background tasks for which there are separate schedules. Migrate and initialize are additional background tasks that follow the same schedule as rebuild. For each background task, up to 7 time periods can be registered, known as slots 1 through 7. Each task schedule can be managed by a set of commands including **add**, **del**, **show** and **set** a task. Background task schedules have a slot id, start-day-time, duration, status attributes.

For details about setting up a schedule for background selftest tasks, see “Setting Up a Selftest Schedule” on page 44.

Selftest activity provides two types of selftests; UDMA (Ultra Direct Memory Access) and SMART (Self Monitoring Analysis and Reporting). Both self tests are checked once each day by default.

UDMA self test entails checking the current ATA bus speed (between controller and attached disk), which could have been throttled down during previous operations and increase the speed for best performance (usually one level higher). Possible speeds include 33, 66, 100 and 133 Mhz (at this writing). Note that UDMA selftest is not applicable (or required) with SATA drives, but is left enabled by default.

SMART activity instructs the controller to check certain SMART supported thresholds by the disk vendor. An AEN is logged to the alarms page if a drive reports a SMART failure.

The `show selftest` command displays the current selftest background task schedule as illustrated below. Selftests do not have a time duration since they are completed momentarily.

```
//localhost> /c1 show selftest

Selftest Schedule for Controller /c1
=====================================================
Slot    Day     Hour            UDMA            SMART
-----------------------------------------------------
1       Sun     12:00am         enabled         enabled
2       Mon     12:00am         enabled         enabled
3       Tue     12:00am         enabled         enabled
4       Wed     12:00am         enabled         enabled
5       Thu     12:00am         enabled         enabled
6       Fri     12:00am         enabled         enabled
7       Sat     12:00am         enabled         enabled
```

/cx add rebuild=*ddd:hh:duration*

This command adds a new background rebuild task to be executed on the day *ddd* (where ddd is Sun, Mon, Tue, Wed, Thu, Fri, and Sat), at the hour *hh* (range 0 .. 23), for a duration of *duration* (range 1 .. 24) hours. A maximum of seven rebuild tasks can be scheduled. This command will fail if no (empty) task slot is available.

For example:

```
//localhost> /c1 add rebuild=Sun:16:3
```

adds a rebuild background task schedule to be executed on Sundays at 16 hours (4:00 PM) for a duration of 3 hours.

Setting Up a Rebuild Schedule

Setting up a rebuild schedule requires several steps, and several different CLI commands in addition to */cx add rebuild*.

To set up the rebuild schedule you want to use, follow this process:

1 Use the */cx show rebuild* command to display the current schedule for rebuild tasks. (For details, see page 38.)

2 If any of the scheduled tasks do not match your desired schedule, use the */cx del rebuild* command to remove them. (For details, see page 44.)

3 Use the */cx add rebuild* command to create the rebuild schedule slots you want (described above.)

4 Use the */cx set rebuild=enable* command to enable the schedule (this enables all rebuild schedule slots). (For details, see page 45.)

Warning: If all time slots are removed, be sure to also disable the schedule. Otherwise the applicable background task will never occur.

/cx add verify=*ddd:hh:duration*

This command adds a new background verify task to be executed on the day *ddd* (where ddd is Sun, Mon, Tue, Wed, Thu, Fri, and Sat), at hour *hh* (range 0 .. 23), for a duration of *duration* (range 1 .. 24) hours. A maximum of seven verify tasks can be scheduled. This command will fail if no (empty) task slot is available.

For example:

```
//localhost> /c1 add verify=Sun:16:3
```

adds a verify background task schedule to be executed on Sundays at 16 hours (4:00 PM) for a duration of 3 hours.

Setting Up a Verify Schedule

Setting up a verify schedule requires several steps, and several different CLI commands in addition to */cx add verify.*

To set up the verify schedule you want to use, follow this process:

1 Use the */cx show verify* command to display the current schedule for verify tasks. (For details, see page 40.)

2 If any of the scheduled tasks do not match your desired schedule, use the */cx del verify* command to remove them. (For details, see page 45.)

3 Use the */cx add verify* command to create the verify schedule slots you want (described above.)

4 Use the */cx set verify=enable* command to enable the schedule (this enables all rebuild schedule slots). (For details, see page 46.)

5 Use the /*cx/ux set autoverify=on* command to turn on autoverify for each unit you want to follow the schedule. (For details, see page 55.)

Note: If you do not enable autoverify for units or start a verification manually, your verify schedule will not run, even if it is enabled with the */cx set verify=enable* command.

Warning: If all time slots are removed, be sure to also disable the schedule. Otherwise the applicable background task will never occur.

/cx add selftest=*ddd:hh*

This command adds a new background selftest task to be executed on the day *ddd* (where ddd is Sun, Mon, Tue, Wed, Thu, Fri, and Sat), at hour *hh* (range 0 .. 23). Notice that selftest runs to completion and as such no duration is provided. A maximum of seven selftest tasks can be scheduled. This command will fail if no (empty) task slot is available.

For example:

```
//localhost> /c1 add selftest=Sun:16
```

adds a selftest background task schedule to be executed on Sundays at 16 hours (4:00 PM).

Setting Up a Selftest Schedule

Setting up a selftest schedule requires several steps, and several different CLI commands in addition to */cx add selftest*.

To set up the selftest schedule you want to use, follow this process:

1 Use the */cx show selftest* command to display the current schedule for selftest tasks. (For details, see page 41.)

2 If any of the scheduled tasks do not match your desired schedule, use the */cx del selftest* command to remove them. (For details, see page 45.)

3 Use the */cx add selftest* command to create the selftest schedule slots you want (described above.)

4 Use the */cx set selftest=enable* command to enable the schedule (this enables all selftest schedule slots). (For details, see page 46.)

/cx del rebuild=*slot_id*

This command removes the rebuild background task in slot *slot_id*.

For example:

```
//localhost> /c1 del rebuild=2
```

removes the rebuild background task in slot 2.

Warning: If all time slots are removed, be sure to also disable the schedule. Otherwise the rebuild background task will never occur.

/cx del verify=*slot_id*

This command removes the verify background task in slot *slot_id*.

For example:

```
//localhost> /c1 del verify=3
```

removes the rebuild background task in slot 3.

Warning: If all time slots are removed, be sure to also disable the schedule. Otherwise the verify background task will never occur.

/cx del selftest=*slot_id*

This command removes (or unregisters) the selftest background task in slot *slot_id*.

For example:

```
//localhost> /c1 del selftest=3
```

Will remove selftest background task in slot 3.

Warning: If all time slots are removed, be sure to also disable the schedule. Otherwise the selftest background task will never occur.

/cx set rebuild=enable|disable/1..5

This command enables or disables all rebuild background tasks on controller */cx* and sets the priority of rebuild versus I/O operations. When enabled, only defined scheduled tasks will be followed (or used). Any previous on-demand background tasks will be ignored.

The priority of rebuild versus I/O operations is specified with *1..5*, where 1 is more resources and 5 the least. Setting the value to 1 gives maximum processing time to rebuilds rather than I/O. Setting the value to 5 gives maximum processing time to I/O rather than rebuilds.

Enabling and disabling rebuild is only for 9000 models, however the rebuild rate (*1..5*) applies to all controllers.

7000- and 8000-series controllers have only one setting for Task Rate; it applies to both rebuild and verify rates. This rate is not persistent following a reboot for 7000- and 8000-series controllers.

/cx set verify=enable|disable|1..5

This command enables or disables all verify background tasks on controller */cx* and (when enabled) sets the priority of verification versus I/O operations. When enabled, only defined scheduled tasks will be followed (or used). Any previous on-demand background tasks will be ignored.

The priority of verify versus I/O operations is specified with *1..5*, where 1 is more resources and 5 the least. Setting this value to 1 implies fastest verify, and 5 implies fastest I/O.

Enabling and disabling verify is only for 9000 models, however the verify rate (*1..5*) applies to all controllers.

Note: When enabling the verify schedule you must also remember to enable the autoverify setting for the arrays to be verified. For more information see "/cx/ux set autoverify=on|off" on page 55.

/cx set selftest=enable|disable [task=UDMA|SMART]

This command enables or disables all selftest tasks or a particular selftest_task (UDMA or SMART).

Enabling and disabling selftest is only for 9000 models. 7/8000 models have the same internal schedule, but it is not viewable or changeable.

For example:

```
//localhost> /c0 selftest=enable task=UDMA
```

enables UDMA selftest on controller c0.

/cx set exportjbod=*on/off*

This command allows you to set the JBOD Export Policy to *on* or *off*. If the JBOD export policy is *off*, CLI will not be able to create JBODs. During reboot, firmware will not export JBOD units to the operating system.

The JBOD Export Policy is only supported on 9000-series controllers. Previous models did not have such a policy enforcement feature.

A JBOD is an unconfigured disk attached to your 3ware RAID controller. AMCC recommends that you use Single Disk as a replacement for JBOD, to take advantage of features such as RAID level migration.

/cx set ondegrade=cacheoff|follow

This command is only for 9000 series controllers.

This command allows you to set a controller-based cache policy. If the policy is set to *cacheoff* and a unit degrades, the firmware will disable the write-cache on the degraded unit, regardless of what the unit-based cache policy is.

If the policy is set to *follow* and a unit degrades, firmware will follow whatever cache policy has been set for that unit. (For details about the unit-based policy, see "/cx/ux set cache=on|off [quiet]" on page 55.)

/cx set spinup=*nn*

This command is only for 9000 series controllers.

This command allows you to set a controller-based Disk Spinup Policy. The value must be a positive integer between 1 and the number of disks/ports supported on the controller (4, 8, or 12).

This policy is used to stagger spinups of disks at boot time in order to spread the power consumption on the power supply. For example, given a spinup policy of 2, the controller will spin up two disks at a time, pause, and then spin up another 2 disks. The amount of time to pause can be specified with the Spinup Stagger Time Policy (*/cx set stagger*).

Not all drives support staggered spinup. If you enable staggered spinup and have drives that do not support it, the setting will be ignored.

/cx set stagger=*nn*

This command is only for 9000 series controllers.

This command allows you to set a controller-based Disk Spinup Stagger Time Policy. The value must be a positive integer between 0 to 60 seconds. This policy, in conjunction with Disk Spinup Policy, specifies how the controller should spin up disks at boot time.

/cx set autocarve=*on*/*off*

This command is only for 9000 series controllers.

This command allows you to set the auto-carve policy to *on* or *off*. When the auto-carve policy is set to *on*, any unit over 2 TB is created as one or more 2 TB volumes and a remaining volume. Each volume can then be treated as an individual disk with its own file system. This feature is used to make units larger than 2 TBs accessible to operating systems limited to 2 TB filesystem.

For example, a 3 TB unit would be configured into one 2 TB volume and one 1 TB volume. A 5 TB unit would be configured into two 2 TB volumes and one 1 TB volume.

When auto-carve policy is set to *off*, all new units are created as a single large volume. If the operating system can only recognize up to 2 TBs, space over 2 TB will not be available.

/cx start mediascan

This command applies only to 7000/8000 controllers. For 9000 series controllers, use the verify command.

This command provides media scrubbing for validating the functionality of a disk, including bad block detection, remapping, and so forth. The command starts a media scan operation on the specified controller */cx*.

/cx stop mediascan

This command applies only to 7000/8000 controllers.

This commands stops a media scan operation on the specified controller */cx*. (Media scans are started using */cx start mediascan*.)

Unit Object Commands

Unit Object commands provide information and perform actions related to a specific unit, such as `/c0/u1` (unit 1 on controller 0). For example, you use logical disk object commands for such tasks as seeing the rebuild, verify, or initialize status of a unit, starting, stopping, and resuming rebuilds and verifies, and setting policies for the unit.

Syntax

```
/cx/ux show
/cx/ux show attribute [attribute ...] where attributes are:
  status|rebuildstatus|verifystatus|initializestatus
   |name|serial|volumes
/cx/ux show all

/cx/ux start rebuild disk=<p:-p...> [ignoreECC]
/cx/ux start verify

/cx/ux pause rebuild   (7000/8000 only)

/cx/ux resume rebuild  (7000/8000 only)

/cx/ux stop verify

/cx/ux flush

/cx/ux del [noscan] [quiet]

/cx/ux set autoverify=on|off
/cx/ux set cache=on|off [quiet]
/cx/ux set ignoreECC=on|off

/cx/ux set name=string

/cx/ux migrate type=RaidType [disk=p:-p] [exclude=p:-p]
  [group=3|4|5|6] [stripe=Stripe] [noscan] [nocache]
  [autoverify]         (9000 only0
/cx/ux export [noscan] [quiet]
```

/cx/ux show

This command shows summary information about the specified unit */cx/ux*. If the unit consists of sub-units as with the case of RAID-1, RAID-5, RAID-10, RAID-50, then each sub-unit is further presented.

One application of this command is to see which sub-unit of a degraded unit has caused the unit to degrade and which disk within that sub-unit is the

source of degradation. Another application is to see the source and destination units during a migration.

Example:

```
//localhost> /c0/u0 show
Unit     UnitType  Status        %Cmpl  Port  Stripe  Size(GB)  Blocks
------------------------------------------------------------------------
u0       RAID-50   OK            -      -     64K     596.05    1249921024
u0-0     RAID-5    OK            -      -     64K     -         -
u0-0-0   DISK      OK            -      p0    -       149.10    312481280
u0-0-1   DISK      OK            -      p2    -       149.10    312481280
u0-0-2   DISK      OK            -      p3    -       149.10    312481280
u0-1     RAID-5    OK            -      -     64K     -         -
u0-1-0   DISK      OK            -      p4    -       149.10    312481280
u0-1-1   DISK      OK            -      p5    -       149.10    312481280
u0-1-2   DISK      OK            -      p6    -       149.10    312481280.
```

/cx/ux show *attribute* [*attribute* ...]

This command shows the current setting of the specified attributes. One or many attributes can be requested. Specifying an invalid attribute will terminate the loop. Possible attributes are: status, rebuildstatus, verifystatus, initializestatus.

/cx/ux show status

This command presents the status of the specified unit.

Possible statuses include: OK, VERIFYING, VERIFY-PAUSED, INITIALIZING, INIT-PAUSED, REBUILDING, REBUILD-PAUSED, DEGRADED, MIGRATING, MIGRATE-PAUSED, RECOVERY, INOPERABLE, and UNKNOWN.

Examples:

```
//localhost> /c0/u0 show status
/c0/u5 status = OK
```

/cx/ux show rebuildstatus

This command presents the rebuildstatus (if any) of the specified unit.

Examples:

```
//localhost> /c0/u5 show rebuildstatus
/c0/u5 is not rebuilding, its current state is OK
```

If the unit is in the process of migrating, the command will return the following:

```
//localhost> /c0/u5 show rebuildstatus
/c0/u5 is not rebuilding, its current state is MIGRATING
```

/cx/ux show verifystatus

This command presents the verifystatus (if any) of the specified unit.

Examples:

```
//localhost> /c0/u5 show verifystatus
/c0/u5 is not verifying, its current state is OK
```

/cx/ux show initializestatus

This command presents the initializestatus (if any) of the specified unit.

Examples:

```
//localhost> /c0/u5 show initializestatus
/c0/u5 is not initializing, its current state is OK
```

/cx/ux show name

This feature only applies to 9000 series controllers.

This command presents the name (if any) of the specified unit.

Examples:

```
//localhost> /c0/u5 show name
/c0/u5 name = Joe
```

/cx/ux show serial

This feature only applies to 9000 series controllers.

This command presents the unique serial number of the specified unit.

Examples:

```
//localhost> /c0/u5 show serial
/c0/u5 Serial Number = 12345678901234567890
```

/cx/ux show volumes

This feature only applies to 9000 series controllers.

This command presents the number of volumes in the specified unit. The number of volumes will normally be “1” unless the drive capacity exceeds 2TB and auto-carving is enabled.

/cx/ux show all

This command shows the current setting of all above attributes.

If the auto-carve policy was on at the time the unit was created and the unit is over 2TB, the number of multiple volumes will be displayed.

Example:

```
//localhost> /c0/u1 show all
/c0/u1 status = OK
/c0/u1 is not rebuilding, its current state is OK
/c0/u1 is not verifying, its current state is OK
/c0/u1 is not initializing, its current state is OK
/c0/u1 volume(s) = 2
/c0/u1 name = 1234567
/c0/u1 serial number = C6CPR7JMF98DA8001DF0

Unit     UnitType  Status        %Cmpl  Port  Stripe  Size(GB)  Blocks
-------------------------------------------------------------------------
u1       RAID-0    OK            -      -     64K     3578.40   7499550720
u1-0     DISK      OK            -      p0    -       298.20    624962560
u1-1     DISK      OK            -      p1    -       298.20    624962560
u1-2     DISK      OK            -      p2    -       298.20    624962560
u1-3     DISK      OK            -      p3    -       298.20    624962560
u1-4     DISK      OK            -      p4    -       298.20    624962560
u1-5     DISK      OK            -      p5    -       298.20    624962560
u1-6     DISK      OK            -      p6    -       298.20    624962560
u1-7     DISK      OK            -      p7    -       298.20    624962560
u1-8     DISK      OK            -      p8    -       298.20    624962560
u1-9     DISK      OK            -      p9    -       298.20    624962560
u1-10    DISK      OK            -      p10   -       298.20    624962560
u1-11    DISK      OK            -      p11   -       298.20    624962560
```

/cx/ux export [noscan] [quiet]

This command allows you to export (or remove) a unit. Exporting a unit instructs the firmware to remove the specified unit from its poll of managed units, but retains the DCB (Disk Configuration Block) metadata. As such the unit can later be re-imported.

noscan is used to not inform the OS of this change. The default is to inform the OS.

quiet is used for non-interactive mode. No confirmation is given and the command is executed immediately. This is useful for scripting purposes.

Example of interactive mode:

//localhost> /c0/u0 export

```
Are you sure you want to export or remove the unit /c0/u0? Enter
Y to continue.
```

Note: After the unit is removed through the CLI, the unit can be physically removed. Hot swap carriers are required to do this while the system is online. Otherwise you must power down the system to prevent system hangs and damage.

/cx/ux del [noscan] [quiet]

This command allows you to delete a unit. Deleting a unit not only removes the specified unit from the controller's list of managed units, but also destroys the DCB (Disk Configuration Block) metadata. After deleting a unit, ports (or disks) associated with the unit will be part of the free pool of managed disks.

Warning: This is a destructive command and should be used with care. All data on the specified unit will be lost after executing this command.

noscan is used to not inform the OS of this change. The default is to inform the OS.

quiet is used for non-interactive mode. No confirmation is given and the command is executed immediately. This is useful for scripting purposes.

Example of interactive mode:

//localhost> /c0/u0 del

```
Are you sure you want to delete the unit /c0/u0? Enter Y to
continue.
```

/cx/ux start rebuild disk=p<*p:-p...*> [ignoreECC]

This command allows you to rebuild a degraded unit using the specified *disk=p*. Rebuild only applies to redundant arrays such as RAID 1, RAID 5, RAID 10, and RAID 50.

During rebuild, bad sectors on the source disk will cause the rebuild to fail. You can allow the operation to continue by using *ignoreECC*.

The rebuild process is a background task and will change the state of a unit to REBUILDING. Various show commands also show the percent completion as rebuilding progresses.

Note that the disk used to rebuild a unit (specified with *disk=p*) must be a SPARE or a unconfigured disk. You must first remove the degraded drive(s) before starting the rebuild. Refer to the command “/cx/px export [noscan] [quiet]” on page 63 for details. Also refer to the command “/cx rescan [noscan]” on page 36 to add new drives or to retry the original drive.

If you are rebuilding a RAID 50 or RAID 10 unit, multiple drives can be specified if more than one sub-array is degraded.

When you issue this command, the specified rebuild will begin if schedules are disabled; otherwise it will pause until the next scheduled rebuild. A file system check is recommended following rebuild when using the ignoreECC option.

/cx/ux start verify

This command starts a background verification process on the specified unit */cx/ux*. The following table shows the supported matrix as a function of the controller model and logical unit type.

N/A (Not Applicable) refers to cases where the given logical unit type is not supported on that controller model.

Table 8: Supported RAID (Logical Unit) Types for Verification

Model	R0	R1	R5	R10	R50	Single	JBOD	Spare
7K/8K	No	Yes	Yes	Yes	N/A	N/A	No	No
9K	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

When you issue this command, the specified verify will begin if schedules are disabled; otherwise it will pause until the next scheduled verify. Verify will also pause if a rebuild or initialization is currently in progress.

/cx/ux pause rebuild

This command allows you to pause the rebuild operation on the specified unit */cx/ux*. This feature is only supported on the 7000/8000 series controllers. 9000 series controllers have an on-board scheduler where rebuild operations can be scheduled to take place at specified start and stop times. The *pause rebuild* function is provided to enable 7000/8000 users to achieve similar functionality with use of Linux-provided schedulers such as cron(8) or at(1), or user-supplied programs.

/cx/ux resume rebuild

This command allows you to resume the rebuild operation on the specified unit */cx/ux*. This feature is intended only for 7000/8000 series controllers. 9000 series controllers have an on-board scheduler where rebuild operations can be scheduled to take place at specified start and stop times. The *rebuild resume* function is provided to enable 7000/8000 users to achieve similar functionality with use of Linux-provided schedulers such as cron(8) or at(1), or user supplied programs.

/cx/ux stop verify

This command stops a background verification process on the specified unit */cx/ux*. Table 8 on page 54 shows the supported matrix as a function of the controller model and logical unit type.

/cx/ux flush

This command allows you to flush the write cache on the specified unit */ux* associated with controller */cx*. Note that this command does not apply to spare unit types.

/cx/ux *set* autoverify=*on/off*

This feature only applies to 9000 series controllers.

This command allows you to turn on and off the autoverify operation on a specified unit */cx/ux* during allocated schedule windows. You can use the *show verify* command to display the existing schedule windows.

Auto-verify allows the controller to run the verify function, if or when deemed necessary, during the schedule window. If no schedule is set up and auto-verify is enabled, then the controller can run the verify function any time it is deemed necessary. This can include pausing the process and restarting it before the verify finishes. For additional information, see "Setting Up a Verify Schedule" on page 43.

/cx/ux *set* cache=*on/off* [quiet]

This command allows you to turn on or off the write cache for a specified unit */cx/ux*. This feature is supported on both 7000/8000 and 9000 models.

Write cache includes the disk drive cache and controller cache. Note that for some configuration types, there is only disk drive cache and no controller cache (for example, JBOD).

The following table shows the supported RAID types for caching as a function of controller model and logical unit type. N/A (Not Applicable) refers to cases where the given logical unit type is not supported on a particular controller model.

Table 9: Supported RAID Types for Caching

Model	R0	R1	R5	R10	R50	Single	JBOD	Spare
7K/8K	Yes	Yes	Yes	Yes	N/A	N/A	Yes	No
9K	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

The *quiet* attribute turns off interactive mode.

/cx/ux set ignoreECC=on|off

This feature only applies to 9000 series controllers.

This command allows you to set the ignoreECC policy for a given unit.

When ignoreECC policy is set to *off*, if a rebuild process encounters bad sectors on the source disk, the rebuild will fail. When ignoreECC is set to *on*, such errors are ignored, and the rebuild will continue. When you use ignoreECC, a file system check is recommended following the rebuild, to insure data integrity.

/cx/ux set name=string

This command allows you to name the unit with an arbitrary name. You can use this name in conjunction with the unit serial number to cross-reference with the unit. The system does not check to ensure uniqueness of names, so be careful to assign different names to each unit.

Note: The unit's serial number is automatically assigned when the unit is created and is not changeable.

/cx/ux migrate type=*RaidType* [disk=p:-*p*] [exclude=p:-*p*] [group=*3*/*4*/*5*/*6*] [stripe=*Stripe*] [noscan] [nocache] [autoverify]

This feature only applies to 9000 series controllers.

This command allows you to change the existing configuration of a unit with *type=RaidType*. You can make three types of changes:

- Increase the capacity
- Change the RAID level (with the same or increased capacity)
- Change the stripe size

The unit that results from the migration is subject to the same rules and policies that apply when creating a new unit with the /cx/add command. For example, a valid number of disks and parameters must be specified.

The destination unit must use all source disks and potentially augment the number of disks in the disk=*p:-p* disk list, if the capacity is being expanded. Or, if the capacity of the unit is being reduced, the destination unit must use source disks with the exclude=*p:-p* disk list to exclude particular disk(s). Unspecified parameters are inherited from the source unit. Both source name and serial number will be carried over to the destination unit.

A special case of this command is when the source unit has a type of RAID1 and destination unit has a type of single. In this case, the migrate command splits both drives into two identical single disks. The disk namename will be duplicated on the destination units, but the source unit serial number will not be carried over to the new unit. The new destination unit will have its own serial number.

type=RaidType specifies the RAID type of the destination unit. Possible unit types include **raid0**, **raid1**, **raid5**, **raid10**, **raid50**, or **single**.

For example, “type=raid5” indicates the destination unit is RAID-5. The **type=single** is a special case of the migrate command. It splits the source unit RAID-1 or TWINSTOR into multiple Single units.

Note: You can only migrate a unit to a RAID level that has the same or more capacity as the existing one. A four-drive RAID 5 unit can migrate to a four-drive RAID 0, but a four-drive RAID 0 unit cannot migrate to a four-drive RAID 5, without adding another drive, due to the need for additional storage capacity for parity bits.

The following table illustrates valid migration paths:

Table 10: Valid Migration Paths

	Destination							
Source	**R0**	**R1**	**R5**	**R10**	**R50**	**Single**	**JBOD**	**Spare**
R0	Yes	No	Yes	Yes	Yes	No	No	No
R1	Yes	No	Yes	Yes	Yes	Yes	No	No
R5	Yes	No	Yes	Yes	Yes	No	No	No
R10	Yes	No	Yes	Yes	Yes	No	No	No
R50	Yes	No	Yes	Yes	Yes	No	No	No
Single	Yes	Yes	Yes	Yes	Yes	No	No	No
JBOD	No	No	No	No	No	No	No	No
Spare	No	No	No	No	No	No	No	No

disk=p:-p.. consists of a list of ports (disks) to be used in addition to the source disks in the construction of the destination unit. One or more ports can be specified. Multiple ports can be specified using a colon (:) or a dash (-) as port index separators. A dash indicates a range and can be mixed with colons. For example **disk=0:1:2-5:9:12** indicates port 0, 1, 2 through 5 (inclusive), 9 and 12.

Table 11: Supported Stripe Sizes

Model	R0	R1	R5	R10	JBOD	Spare	R50	Single
9K	16	N/A	16	16	N/A	N/A	16	N/A
	64		64	64			64	
	256		256	256			256	

group=3|4|5|6 indicates the number of disks per group for a RAID 50 type. (This attribute can only be used when type=raid50.) Recall that a RAID 50 is a multi-tier array. At the bottom-most layer, N number of disks per group are used to form the RAID 5 layer. These RAID 5 arrays are then integrated into a RAID 0. This attribute allows you to specify the number of disks in the RAID 5 level. Valid values are 3, 4, 5 and 6. For example **group=3** indicates 3 disks of RAID 5 at the bottom layer of RAID 50.

Note that a sufficient number of disks are required for a given pattern or disk group. For example, given 6 disks, specifying 3 will create two RAID 5 arrays. With 12 disks, specifying 3 will create four RAID 5 arrays under the RAID 0 level. With only 6 disks a grouping of 6 is not allowed, as you would basically be creating a RAID 5.

The default RAID 50 grouping varies, based on number of disks. For 6 and 9 disks, default grouping is 3. For 8 disks, the default grouping is 4. For 10 disks, the default grouping is 5, and for 12 disks, the default grouping is 4. In the case of 12, the disks could be grouped into groups of 3, 4, or 6 drives. A grouping of 4 is set by default as it provides the best of net capacity and performance.

Note that RAID-10 always has **group=2**, so an attribute specifying it's group is not necessary.

stripe=Stripe consists of the stripe size to be used. The following table illustrates the supported and applicable stripes on unit types and controller models. Stripe size units are in KB (kilobytes).

Table 12: Supported Stripe Sizes

Model	R0	R1	R5	R10	JBOD	Spare	R50	Single
9K	16	N/A	16	16	N/A	N/A	16	N/A
	64		64	64			64	
	256		256	256			256	

noscan attribute instructs CLI not to notify the OS of the creation of the new unit. By default CLI will inform the OS. One application of this feature is to prevent the OS from creating block special devices such as /dev/sdb and /dev/sdc as some implementations might create naming fragmentation and a moving target.

nocache attribute instructs CLI to disable the write cache on the migrated unit. Enabling write cache increases write performance at the cost of potential data loss in case of sudden power loss (unless a BBU or UPS is installed). By default the cache is enabled. To avoid the possibility of data loss in the event of a sudden power loss, it is recommended not to set nocache unless there is a BBU (battery backup unit) or UPS (uninterruptable power supply) installed..

autoverify attribute enables the autoverify attribute on the unit that is to be migrated. For more details on this feature, see “/cx/ux set autoverify=on|off” on page 55.

Migration Process

In all cases of migration, the background migration process must be completed before the newly sized unit is available for use. You can continue using the original unit during this time. Once the migration is finished, a reboot will be required if you are booted from the unit. For secondary storage, depending on your operating system, you may need to first unmount the unit, then then use CLI commands to ‘export’ and ‘rescan’ the unit so that the operating system can see the new capacity, and then remount the unit. For details see “/cx/ux export [noscan] [quiet]” on page 52 and “/cx rescan [noscan]” on page 36.

You may also need to resize the file system, and either resize the existing partition or add a new partition. For instructions, consult the documentation for your operating system.

Warning: It is important that you allow migration to complete before adding or removing drives from the unit. Making physical changes to the unit during migration may cause the migration process to stop, and can jeopardize the safety of your data.

Example of splitting a mirror

```
//localhost> /c1/u3 migrate type=single
```

Indicates that u3 is a TWINSTOR or RAID-1 and the Migrate command splits u3 to u3 and ux with a RAID type of Single.

Example of drive reduction

```
//localhost> /c1/u1 migrate type=raid5 exclude=3
```

In this example, u1 is a 4 disk RAID10 unit. The command indicates that the user wants to migrate it to a RAID5 unit **without** disk 3. **Note:** Before and after the migration, the unit capacity remains the same in this case.

Example of capacity expansion

```
//localhost> /c0/u0 migrate type=raid10 disk=3-4 stripe=16
```

Indicates that the destination unit has a RAID type of raid10 and has the disk 3 and 4 in addition to all the disks in the existing unit u0.

Example of migrate output

The following is an example of how migrating units will be displayed. In this example, the report indicates that */c0/u0* is a migrating unit with 67%

completion. The report also indicate that Source Unit *su0* is of type RAID-1 and Destination Unit *du0* is of type RAID-10.

```
  //localhost> /c0/u0 migrate type=raid10 disk=2-3 stripe=16
Sending migration message to /c0/u0 ... Done.

3ware CLI> /c0 show

Unit  UnitType  Status         %Cmpl  Stripe  Size(GB)  Cache  AVerify  IgnECC
------------------------------------------------------------------------------
u0    Migrator  MIGRATING      0      -       74.4951   ON     OFF      OFF

Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
p0     OK               u0     74.53 GB    156301488     3JV2Q1VA
p1     OK               u0     232.83 GB   488281250     WD-WMAEH15764
p2     OK               u0     232.83 GB   488281250     WD-WMAEH17004
p3     OK               u0     232.83 GB   488281250     WD-WMAEH17000
p4     NOT-PRESENT      -      -           -             -
p5     NOT-PRESENT      -      -           -             -
p6     NOT-PRESENT      -      -           -             -
p7     NOT-PRESENT      -      -           -             -
p8     NOT-PRESENT      -      -           -             -
p9     NOT-PRESENT      -      -           -             -
p10    NOT-PRESENT      -      -           -             -
p11    NOT-PRESENT      -      -           -             -

3ware CLI> /c0/u0 show

Unit     UnitType  Status         %Cmpl  Port  Stripe  Size(GB)  Blocks
---------------------------------------------------------------------
u0       Migrator  MIGRATING      0      -     -       -         -

su0      RAID-1    OK             -      -     -       74.4951   156227584
su0-0    DISK      OK             -      p0    -       74.4951   156227584
su0-1    DISK      OK             -      p1    -       74.4951   156227584

du0      RAID-10   OK             -      -     16K     148.99    312455168
du0-0    RAID-1    OK             -      -     -       -         -
du0-0-0  DISK      OK             -      p0    -       74.4951   156227584
du0-0-1  DISK      OK             -      p1    -       74.4951   156227584
du0-1    RAID-1    OK             -      -     -       -         -
du0-1-0  DISK      OK             -      p2    -       74.4951   156227584
du0-1-1  DISK      OK             -      p3    -       74.4951   156227584
```

Port Object Commands

Port Object Messages are commands that provide information and perform actions related to a specific disk, attached to a port, such as `/c0/p0`. You use port object commands for such tasks as seeing the status, model, or serial number of the drive.

Syntax

```
/cx/px show
/cx/px show attribute [attribute ...]  where attributes are:
   status|model|serial|capacity|smart|firmware.
/cx/px show all

/cx/px export [quiet]
```

/cx/px *show*

This command shows summary information about the specified disk attached to port */cx/px*. Typical information looks like:

Example:

```
//localhost> /c1/p5 show

Port   Status     Unit   Size        Blocks        Serial
---------------------------------------------------------
p5     OK         u0     149.05 GB   312581808     3JS0L9QW
```

The above report indicates that port 5 of controller 1 is attached to one l disk with status OK participating in unit 0.

/cx/px show *attribute* [*attribute* ...]

This command shows the current setting of the given attributes on the specified port or disk. One or many attributes can be requested. Specifying an invalid attribute will terminate the loop. Possible attributes are: status, model, serial, firmware, capacity, and smart.

/cx/px show status

This command displays the status of the disk attached to the specified port.

Example:

```
 //localhost> /c0/p5 show status
 /c0/p5 Status = OK
```

/cx/px show model

This command displays the model of the disk attached to the specified port.

Example:

```
//localhost> /c0/p5 show model
/c0/p5 Model = WDC WD1600BB-00DAA0
```

/cx/px show serial

This command displays the serial number of the disk attached to the specified port.

Example:

```
//localhost> /c0/p5 show serial
/c0/p5 Serial = WD-WMACK140649
```

/cx/px show firmware

This command displays the firmware version of the disk attached to the specified port.

Example:

```
//localhost> /c0/p5 show firmware
/c0/p5 Firmware Version = 65.13G65
```

/cx/px show capacity

This command presents the capacity of the disk attached to the specified port in two formats—GB and blocks. Note that of this version, the GB format is computed based on division by 1000 (not 1024).

Example:

```
//localhost> /c0/p5 show capacity
149.05 GB (312581808)
```

/cx/px show smart

This command extracts SMART (Self Monitoring Analysis and Reporting) data from the specified disk. Because the data is extracted live from the disk, this command can be used to get the most recent data about the presence or absence of a disk.

The SMART data is displayed in hexadecimal form.

Example:

```
//localhost> /c0/p5 show smart
10 00 01 0B 00 C8 C8 00 00 00 00 00 00 00 03 07
00 9A 96 BC 14 00 00 00 00 00 04 32 00 64 64 7A
00 00 00 00 00 00 05 33 00 C8 C8 00 00 00 00 00
...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2C
```

Note: The SMART data is not decoded. If the disk attached to the specified port is not present or if there are cabling problems reaching the disk, CLI will return an error. This can be one way of detecting whether or not a disk is present.

/cx/px show all

This command shows the current setting for all port-related attributes: status, model, serial, firmware, capacity, and smart.

Example:

```
//localhost> /c1/p5 show all
/c1/p5 Status = OK
/c1/p5 Model = ST3160023AS
/c1/p5 Firmware Version = 3.14
/c1/p5 Serial = 3JS0L9QW
/c1/p5 Capacity = 149.05 GB (312581808)

0A 00 01 0F 00 3D 33 25 8C BA 03 00 00 00 03 03
00 61 60 00 00 00 00 00 00 00 04 32 00 64 64 00
00 00 00 00 00 00 05 33 00 64 64 00 00 00 00 00
00 00 07 0F 00 4E 3E 05 13 D8 03 00 00 00 09 32
...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 8D
```

/cx/px export [noscan] [quiet]

This command allows you to export (or remove) a port (or drive) */cx/px*. Exporting a port instructs the firmware to remove the specified port from its pool of managed ports, but does not retain the DCB (Disk Configuration Block) metadata on the attached disk. You can import (or re-introduce) the port by rescanning the controller.

noscan is used to not inform the OS of this change. The default is to inform the OS.

quiet is for non-interactive mode.

Warning: Use caution when using this command as this operation will degrade any redundant units. This command will fail if you attempt to remove a drive from a non-redundant unit. After the drive is removed in CLI it can be removed physically, without powering down the system if a hot swap carrier is available. System hangs and damage can occur if a hot swap carrier is not used.

BBU Object Commands

BBU (Battery Backup Unit) Object Commands are commands that provide information and perform actions related to a specific BBU installed on a specific controller, such as `/c0/bbu`. This object is only available on 9000 series controllers on which a BBU is actually installed.

Syntax

```
/cx/bbu show                              (9000 only)
/cx/bbu show attribute [attribute ...] where attributes are:
   status|batinst|lasttest|volt|temp|cap|serial|fw|
    bootloader|pcb
/cx/bbu show all                          (9000 only)

/cx/bbu test [quiet]                 (9000 only)
  Warning: May take up to 24 hours to complete. Write cache
  will be disabled during the test.

/cx/bbu enable                          (9000 only)

/cx/bbu disable                         (9000 only)
```

/cx/bbu *show*

This command presents a summary report on the specified BBU object.

Example:

```
//localhost> /c0/bbu show
Name  OnlineState BBUReady  Status    Volt     Temp     Hours  LastCapTest
---------------------------------------------------------------------------
bbu   ON          No        Testing   OK       OK       72     01-Jul-2004
```

The command output indicates that the battery capacity was last measured on 01-Jul-2004. The battery is estimated to last for 72 hours from the last tested date. In this example, the BBU unit is currently testing the battery. Both voltage and temperature are normal. The BBU is not ready to backup the write cache on the controller (due to the testing). (For complete information about the BBU, see *3ware 9000 Series Serial ATA RAID Controller User Guide*.

Note: If the BBU is either not present or disabled, the following will be displayed after the //localhost> /c0/bbu show command:

```
Error:  (CLI:053) Battery Backup Unit is not present.
```

/cx/bbu show *attribute* [*attribute* ...]

This command shows the current setting of the given attribute(s) on the BBU board. One or many attributes can be specified. Specifying an invalid attribute will terminate the loop. Possible attributes are: ready, status, batinst, lasttest, volt, temp, cap, serial, fw, pcb, bootloader.

/cx/bbu show *status*

This command shows the status of the BBU. Possible values are:

Testing. A battery test is currently in progress. This test may take up to 24 hours to complete. During the test, the BBU is not capable of backup operation and the write cache of the RAID controller is also disabled. If the test is completed with no error and the BBU status changes to WeakBat or OK, the write cache will be re-enabled. If a Fault, Failed or Error occurs during the test, the write cache remains in the disabled state until the problem is fixed.

Charging. The BBU is currently charging the battery. Charging is started automatically by the BBU whenever necessary. During charging, the BBU is not capable of backup operation and the write cache is disabled. Once the test is completed with no error and the BBU status changes to OK, the write cache will be re-enabled. If a FAULT or ERROR occurs during the test, the write cache remains in the disabled state until the problem is fixed.

Fault. A battery fault is detected. The BBU is not capable of backup operation and the write cache is disabled. Replace the battery and/or the BBU board as soon as possible so that the write cache will be enabled again.

Error. A BBU error is detected. The BBU is not capable of backup operation and the write cache is disabled. Replace the battery and/or the Battery Backup Unit as soon as possible so that the write cache will be enabled again.

Failed. The battery failed a test. In this state, the BBU is not capable of backup operation and the write cache is disabled. We recommend you replace the battery and/or the Battery Backup Unit as soon as possible so that the write cache will be enabled again.

WeakBat. The BBU is functioning normally and is online and capable of backing up the write cache. However, the battery is weak and should be replaced.

OK. The BBU is ready, online and capable of backing up the write cache.

- (dash) A battery is not present or a Battery Backup Unit is not installed

/cx/bbu show batinst

This command shows the date when the current battery was installed.

/cx/bbu show lasttest

This command shows the date the battery capacity was last measured. If the battery capacity test has never been run, then 'xx-xxx-xxxx' will be displayed.

Note: The estimated BBU capacity hours displayed is based on the measurement taken during the last test. If you have not run the BBU test command for some time, this number can be misleading. For information about running a test, see "/cx/bbu test [quiet]" on page 67.

/cx/bbu show volt

This command shows the voltage status of the battery. The status can be OK, HIGH, LOW, TOO-HIGH, and TOO-LOW. The HIGH and LOW are in warning range. TOO-HIGH and TOO-LOW are out of the operating range and indicate that it is time to replace the battery. (Contact AMCC to obtain a replacement battery.)

/cx/bbu show temp

This command shows the temperature status of the battery. The status can be OK, HIGH, LOW, TOO-HIGH, and TOO-LOW. The HIGH and LOW are in warning range. TOO-HIGH and TOO-LOW are out of the operating range and indicate that it may be time to replace the battery. (Contact AMCC to obtain a replacement battery.)

/cx/bbu show cap

This command shows the battery capacity in hours.

A value of '0 hours' will be displayed if the battery capacity test has never been run.

Note: The estimated BBU capacity hours displayed is based on the measurement taken during the last test. If you have not run the BBU test command for some time, this number can be misleading. You can use the command `/cx/bbu show lasttest` to check the date of the last test. For information about running a test, see "/cx/bbu test [quiet]" on page 67.

/cx/bbu show serial

This command shows the BBU serial number.

/cx/bbu show fw

This command shows the BBU firmware version number.

/cx/bbu show pcb

This command shows the PCB revision number on the BBU.

/cx/bbu show bootloader

This command shows the BBU's boot loader version.

/cx/bbu show all

This command shows the current settings of all BBU-related attribute: ready, status, batinst, lasttest, volt, temp, cap, serial, fw, pcb, bootloader.

For example:

```
//localhost> /c1/bbu show all

/c1/bbu Firmware Version          = BBU: 1.04.00.007
/c1/bbu Serial Number             = Engineering Sample.
/c1/bbu BBU Ready                 = Yes
/c1/bbu BBU Status                = OK
/c1/bbu Battery Voltage           = OK
/c1/bbu Battery Temperature       = OK
/c1/bbu Estimated Backup Capacity = 241 Hours
  /c1/bbu Last Capacity Test        = 22-Jun-2004
  /c1/bbu Battery Installation Date = 20-Jun-2004
  /c1/bbu Bootloader Version        = BBU 0.02.00.002
  /c1/bbu PCB Revision              = 65

//localhost>
```

/cx/bbu test [quiet]

This command starts the battery capacity test. The test may take up to 24 hours to complete. During the test, the BBU is not capable of backup operation and the write cache of all units attached to that controller is disabled. Once the test is completed with no error and the BBU status returns to OK, the write cache will be re-enabled.

Note: Once started, the test can not be terminated before it completes. Write cache cannot be enabled until the test completes.

AEN (Asynchronous Event Notification) messages are also generated by controllers to notify the user of the command status.

Check for AENs with the alarms command `/cx show alarms [reverse]`. Using the “reverse” attribute displays the most recent AEN message at the bottom of the list. (For a list of all AENs, see the *3ware 9000 Series Serial ATA RAID Controller User Guide*.)

/cx/bbu enable

This command enables BBU detection on the controller. If the BBU is Ready, the controller will utilize BBU functionality in the event of a power failure.

/cx/bbu disable

This command disables BBU detection on the controller. When disabled, the controller ignores the existence of the BBU and will show no BBU is installed even if a BBU is physically attached.

Help Commands

The Help commands provides brief on-line help.

You can get overview help by typing Help at the top-level prompt. This displays a brief definition of commands in both the new syntax and the legacy syntax. (For an example, see the discussion of the command “help” on page 70.)

You can also get help with specific commands, by entering *help* before an object name, or by typing a question mark (?) at the point in a command where you are uncertain what the attributes are.

Help with specific commands

If you enter the help command at the top level, you are considered to be in the Shell Object, and the help command will provide help on the Shell commands focus, show, flush, rescan, and commit. Using the help command on objects (such as /cx, /cx/ux, /cx/px, and /cx/bbu), displays all possible sub-commands associated with the object.

For example: help on the controller object /cx, will display all the sub-commands associated with the controller /cx, like this:

```
//localhost> help /cx
 /cx show
 /cx show Attribute [Attribute ...]          where Attribute is:
          driver|model|firmware|memory|bios|monitor|serial|pcb|pchip|achip
          numports|numunits|numdrives|unitstatus|drivestatus|allunitstatus
          exportjbod|ondegrade|spinup|stagger|autocarve
 /cx show all               where all means Attributes and configurations.

 /cx show diag
 /cx show alarms [reverse]
 /cx show rebuild                                              (9000 only)
 /cx show verify                                               (9000 only)
 /cx show selftest                                             (9000 only)
```

```
/cx add type=<RaidType> disk=<p:-p..> [stripe=<Stripe>] [noscan] [nocache]
            [group=<3|4|5|6>] [autoverify] [ignoreECC]
            [name=string (9000 only)] RaidType = { raid0, raid1, raid5,
            raid10, raid50, single, spare, JBOD (7000/8000 only)}
 /cx add rebuild=ddd:hh:duration                              (9000 only)
 /cx add verify=ddd:hh:duration                               (9000 only)
 /cx add selftest=ddd:hh                                      (9000 only)

 /cx del rebuild=slot_id                                      (9000 only)
 /cx del verify=slot_id                                       (9000 only)
 /cx del selftest=slot_id                                     (9000 only)

 /cx set exportjbod=on|off                                    (9000 only)
 /cx set ondegrade=cacheoff|follow                            (9000 only)
 /cx set spinup=nn                                            (9000 only)
 /cx set stagger=nn                                           (9000 only)
 /cx set autocarve=on|off                                     (9000 only)
 /cx set rebuild=enable|disable|<1..5>  (enable|disable for 9000 only)
 /cx set verify=enable|disable|<1..5>   (enable|disable for 9000 only)
 /cx set selftest=enable|disable [task=UDMA|SMART]         (9000 only)

 /cx flush
 /cx commit                    (Windows only) (Also known as shutdown)
 /cx start mediascan                                  (7000/8000 only)
 /cx stop mediascan                                   (7000/8000 only)
 /cx rescan [noscan] NOTE: Does not import non-JBOD on 7000/8000 models.

//localhost>
```

Help with attributes

As you work with specific objects or commands, you can also use *?* to get help.

For example: If you enter the command `/c0 show` and then need help on what specific attribute syntax is possible, you can use *?* to get help as following:

```
//localhost> /c0 show ?

 /cx show
 /cx show Attribute [Attribute ...]           where Attribute is:
          driver|model|firmware|bios|monitor|serial|pcb|pchip|achip
          numports|numunits|numdrives|unitstatus|drivestatus|allunitstatus
          exportjbod|ondegrade|spinup|stagger|autocarve
 /cx show all               where all means Attributes and configurations.
 /cx show diag
 /cx show alarms [reverse]
 /cx show rebuild                                             (9000 only)
 /cx show verify                                              (9000 only)
 /cx show selftest                                            (9000 only)

//localhost>
```

help

This help command provide a table of contents, providing help with the overall navigation of the CLI commands. Typical output looks like the following.

(Details about help commands for the new command syntax are included on the next couple of pages. For details about help commands for the legacy syntax, see “help” on page 105.)

```
//localhost> help
Copyright(c) 2004 Applied Micro Circuits Corporation(AMCC). All rights
reserved.
Copyright(c) 2002, 2003, 2004 3ware, Inc.  All rights reserved.

AMCC/3ware CLI (version 2.x)


Commands  Description
------------------------------------------------------------------
info      Displays information about controller(s), unit(s) and port(s).
maint     Performs maintenance operations on controller(s), unit(s) and
ports.
alarms    Displays current AENs.
set       Displays or modifies controller and unit settings.
sched     Schedules bachground tasks on controller(s)             (9000
only)
quit      Exits the CLI.
           ---- New Command Syntax ----
focus     Changes from one object to another. For Interactive Mode Only!
show      Displays information about controller(s), unit(s) and port(s).
flush     Flush write cache data to units in the system.
rescan    Rescan all empty ports for new unit(s) and disk(s).
commit    Commit dirty DCB to storage on controller(s).        (Windows
only)
/cx       Controller specific commands.
/cx/ux    Unit specific commands.
/cx/px    Port specific commands.
/cx/bbu   BBU specific commands.                                  (9000
only)

Type help <command> to get more details about a particular command.
For more detail information see tw_cli's documentation.
```

help *show*

This command provides specific show-related help, illustrating various ways to use the show command. It provides reports on Controllers, Units and Drives. See the section “Shell Object Commands” on page 21 for more information.

help *flush*

This command provides specific flush-related help, illustrating various ways to use the flush command. See the section “Shell Object Commands” on page 21 for more information.

help *rescan*

This command provides specific rescan related help, illustrating various ways to use the rescan command. See the section “Shell Object Commands” on page 21 for more information.

help *commit*

This command provides specific commit related help, illustrating various ways to use the commit command. See the section “Shell Object Commands” on page 21 for more information.

help *focus*

This command provides specific focus related help, illustrating various ways to use the focus command. See the section “Shell Object Commands” on page 21 for more information.

help /cx

This command provides specific controller */cx* related help, illustrating various commands associated with the controller */cx*. See the section “Controller Object Commands” on page 25 for more information.

help /cx/ux

This command provides specific unit */cx/ux* related help, illustrating various commands to use on a unit */cx/ux*. See the section “Unit Object Commands” on page 49 for more information.

help /cx/px

This command provides specific */cx/px* related help, illustrating various ways to use the */cx/px* command. See the section “Port Object Commands” on page 61 for more information.

help /cx/bbu

This command provides specific */cx/bbu* related help, illustrating various ways to use the */cx/bbu* command. See the section “BBU Object Commands” on page 64 for more information.

Environment Variables

There are two environment variables:

- TW_CLI_INPUT_STYLE (described below)
- TW_CLI_STYLE is a reporting style variable for the legacy syntax. For details, see "Screen Reporting Style" on page 57.

By default, TW_CLI_INPUT_STYLE variable is set to *new.* If you want to disable the focus command, set the TW_CLI_INPUT_STYLE variable to *old.*

- For Redhat and SuSE (bash, ksh, or sh), enter
  `export TW_CLI_INPUT_STYLE=OLD`
- For Linux csh (C-shell), enter:
  `setenv TW_CLI_INPUT_STYLE OLD`
- For Windows, enter
  `set TW_CLI_INPUT_STYLE=OLD`

To keep the new CLI Input Style following a reboot or when a new window or shell is opened you must edit the environment variables in both Windows and Linux. Refer to your operating system's administration guide for more details.

Return Code

While informative messages are written to standard output, error messages are written to standard error. On success, 0 is returned. On failure, 1 is returned.

To view the return code for Linux, at the shell command prompt type:

```
echo $?
```

The screen prints either a 0 or a 1, depending on whether the command was successful or not.

For example, if you had a 3ware controller with an ID of 0, you could type this command:

```
tw_cli /c0 show
(c0 information displayed here)
echo $?
0
```

If you type:

```
tw_cli /c7 show
error: (CLI003) specified controller does not exist.
echo $?
1
```

This example fails (returns 1) because there is no controller 7.

To view the return code for Windows, in a command window type

```
tw_cli /c0 show
(c0 info displayed here)
if errorlevel 0 echo 0
0

tw_cli /c7 show
error....
if errorlevel 1 echo 1
1
```

This example fails (returns 1) because there is no controller 7.

Legacy CLI Syntax Reference

This chapter provides detailed information about using the legacy syntax for the 3ware CLI, using the commands Info, Maint, Sched, Alarms, Set, and Help. Support for the legacy syntax is available only for a limited time.

Note: Information contained in this document that describes usage only for the 3ware 9000 series products does not work with 3ware 7000 or 8000 series controllers.

Info. Information commands provide all information and settings about the 3ware controllers, including array types, array status, array settings, detail controller information, and detail drive information. For details, see “Info Commands” on page 77.

Maint. Maintenance commands perform all maintenance operations on the drives and arrays connect to the 3ware controller. Typical operations include: create array, delete array, rebuild array, verify array, and remove array from the controller. For details, see “Maint Commands” on page 88.

Warning: Operations under the maint command can destroy data, so care should be taken before using this command; CLI does not prompt before the operation is committed.

Sched. Scheduling commands only affect the 9000 controller. Schedule commands allow you to schedule different time slots for background tasks such as rebuild, verify, and selftest. For details, see “Sched Commands” on page 96.

In order to use CLI scheduling commands with 7000 and 8000 controllers, you must use them in conjunction with a time-driven scheduling component under Windows, Linux, or FreeBSD. For example, under Linux, you can use

the cron daemon scheduling utility with the CLI commands for rebuild, verify, and mediascan. (For more information about the specific CLI commands, see “Maint Commands” on page 88 and refer to your Linux documentation or manpages.)

In addition, 3ware also includes on the 3ware CD a utility named tw_sched(1), which is a wrapper around tw_cli(1). Used in conjunction with a time-driven scheduler such as crond(1d), it provides background task scheduling features such as rebuild, verify, and mediascan. For details about tw_sched(1), see the manpages for it.

Scheduling for the 7000 and 8000 series models can also be done using 3DM 1.x. (Note that 3DM 2 only provides scheduling only for 9000 series models.)

Alarms. The Alarms command allows you to display Asynchronous Event Notification (AEN) events that have been generated by controllers. AEN events have different levels of severity. They can be extracted and archived for overall trend analysis. For details, see “Alarms Commands” on page 101.

CLI does not store alarms command in a log file, so when you reboot, alarms in the previous session will be lost. To preserve the alarms through reboot, you can either extract the alarms from CLI and store them in a file, or install 3DM 2, which does log alarm messages. For more information, see the *3ware 9000 Series Serial ATA RAID Controller User Guide*.

Set. Setting commands can be used to modify and change controller and array settings. Settings that can be changed include: rebuild rate, verify rate, and turning on or off cache, autoverify, or overwriteECC. For details, see “Set Commands” on page 103.

Help. Help commands. Options in this category allow you to display help information on the other commands and options. For details, see “Help Commands” on page 105.

Throughout this chapter the examples reflect the interactive method of executing 3ware CLI.

Conventions

The following conventions are used through this guide:

- In text, `monospace font` is used for code and for things you type.
- In commands, an italic font indicates items that you must specify, such as a controller ID, or a unit ID.
- In commands, brackets around an item indicates that it is optional.
- In commands, ellipses (. . .) indicate that more than one parameter can be included.
- In commands, a brace (|) indicates an 'or' situation where the user has a choice between more than one option, but only one can be specified.

For example, in the maint command to rescan all ports and reconstitute all units, the syntax appears as `maint rescan [cid ...] [noscan]`. The italic *cid* indicates that you need to supply a controller ID. The ellipses indicate that you can specify more than one controller ID, separated by spaces. The brackets indicate that you may omit the controller ID, to rescan all controllers, and the noscan parameter, so that the operation will be reported to the operating system.

Screen Reporting Style

In the previous version of the CLI, 3ware has changed the default reporting style to a tabular reporting style for screen displays. Using this format, information is easier to read and analyze. The new style also accommodates automation, by providing consistent columns with or without values so that it can be easily parsed.

The original, non-tabular style is still available. To use the old style, set the TW_CLI_STYLE to OLD as shown below, depending on your operating system.

- For Redhat and SuSE (bash, ksh, or sh), enter
  `export TW_CLI_STYLE=OLD`
- For Linux csh (C-shell), enter:
  `setenv TW_CLI_STYLE OLD`
- For Windows, enter
  `set TW_CLI_STYLE=OLD`

To keep the new CLI output style following a reboot or when a new window or shell is opened you must edit the environment variables in both Windows and Linux.

To use the new style, enter
`TW_CLI_STYLE=" "`

or
`TW_CLI_STYLE="NEW"`

The examples in this document use the new style of reporting.

Info Commands

The *info* commands provide information about the 3ware controller, the attached drives, and configured RAID arrays or units. The *info* commands are for querying purposes only.

Info commands are read-only operations showing various values of controllers, units, and drives.

Syntax

```
info
info   c<c>   [driver|model|firmware|bios|monitor|serial|pcb|pchip|achip
     numports|numunits|numdrives|unitstatus|drivestatus|allunitstatus
     exportjbod|ondegrade|spinup|stagger]
  info c<c> u<u> [status|rebuildstatus|verifystatus|initializestatus]
  info c<c> p<p> [status|model|serial|capacity|smart]
  info c<c> diag
```

Parameters

cid - the controller id

uid - the unit id

pid - the port id

option - specifies the kind of information you want to see.

info

Provides information on all detected controllers. The appropriate device driver must be loaded for the list to show all controllers. The intention is to provide a global view of the environment.

Typical output looks like the following:

```
//localhost> info
Ctl   Model     Ports   Drives  Units  NotOpt   RRate   VRate
------------------------------------------------------------
c0    7500-12   12      5       1      1        2       -
c1    8506-12   12      6       1      0        3       5
```

The output indicates that controller 0 is a 7000 series with 12 ports, 5 drives, and a total of 1 unit in a not optimal state. Not optimal refers to any state except OK and VERIFYING. Other states include INITIALIZATING, REBUILDING, DEGRADED, MIGRATING, and INOPERABLE. The example shows that the controller's rebuild rate (RRate) is set to 2 and the verify rate (VRate) is not applicable (-).

Additional attributes about individual controllers, units, ports and disks can be obtained by querying for them explicitly, using, for example, `info cid` or `info cid uid`. See the other *info* sub-commands below.

info *cid*

Provides overall summary information on controller *cid*. The report consists of two parts; a unit summary listing all present units, and a port summary section listing all present disks and their attached ports.

The unit summary section lists all present units specifying their unit number, unit type (such as RAID 5), status, size (usable capacity) in gigabytes or terabytes, number of blocks, and unit status such as OK, VERIFYING, INITIALIZING, etc. %Compl reports percent completion of REBUILDING

or VERIFYING units. It shows the Stripe size, if applicable, and whether Cache is on or off. It also shows whether AutoVerify and OvrECC are on or off (9000 only). OvrECC is short for Overwrite ECC, or Force Continue on source error. The function is explained in “Set Commands” on page 103.

The port summary section lists all present ports specifying the port number, disk status, unit affiliation, size (GB), blocks (of 512 bytes), and the serial number assigned by the disk vendor.

Additional attributes about units, ports and disks can be obtained by querying for them explicitly. See other info sub-commands below.

Typical output looks like:

```
//localhost> info c0
Unit      UnitType   Status    %Cmpl   Stripe  Size      Cache   AVerify
OvrECC
------------------------------------------------------------------
u0        RAID-1     OK        -       -       149.05    ON      OFFOFF
u1        RAID-5     OK        -       64k     298.22    ON      OFFOFF
u2        SPARE      OK        -       -       149.05    ON      OFF-

Port      Status     Unit      Size    Blocks  Serial
---------------------------------------------------------------
p0        OK         u0        149.05 GB3125818083JS0TF14
p1        OK         u0        149.05 GB3125818083JS0TETZ
p2        OK         u1        149.05 GB3125818083JS0VG85
p3        OK         u1        149.05 GB3125818083JS0VGCY
p4        OK         u1        149.05 GB3125818083JS0VGGQ
p5        OK         u2        149.05 GB3125818083JS0VH1P
p6        OK         -         149.05 GB3125818083JS0TF0P
p7        OK         -         149.05 GB3125818083JS0VF43
p8        OK         -         149.05 GB3125818083JS0VG8D
p9        NOT-PRESENT-         -       -       -
p10       NOT-PRESENT-         -       -       -
p11       NOT-PRESENT-         -       -       -
```

3ware CLI calculates one megabyte as 1024 x 1024, the same calculation that Windows and Linux use. 3DM 2 uses 1024 x 1024, so when using 3DM 2, the capacity listed will match the capacity stated by the CLI. Previous versions of 3DM (v1.x) calculate one megabyte as 1000 x 1000, which is the calculation disk drive vendors use.

info *cid driver*

This command reports the device driver version associated with controller *cid*.

Example:

```
//localhost> info c0 driver
/c0 Driver Version = 1.02.00.036
```

info *cid model*

This command reports the controller model of controller *cid*.

Example:

```
//localhost> info c0 model
/c0 Model = 7506-12
```

info *cid firmware*

This command reports the firmware version of controller *cid*.

Example:

```
//localhost> info c0 firmware
/c0 Firmware Version = FGXX 2.01.00.025
```

info *cid bios*

This command reports the BIOS version of controller *cid*.

Example:

```
//localhost> info c0 bios
/c0 BIOS Version = BG9X 2.01.00.026
```

info *cid monitor*

This command reports the monitor (firmware boot-loader) version of controller *cid*.

Example:

```
//localhost> info c0 monitor
/c0 Monitor Version = BLDR 1.00.00.008
```

info *cid serial*

This command reports the serial number of the specified controller *cid*.

Example:

```
//localhost>//localhost> info c0 serial
/c0 Serial Number = F12705A3240009
```

info *cid pcb*

This command reports the PCB (printed circuit board) revision of the specified controller *cid*.

Example:

```
//localhost> info c0 pcb
/c0 PCB Version = Rev3
```

info *cid pchip*

This command reports the PCHIP (PCI Interface Chip) version of the specified controller *cid*.

Example:

```
//localhost> info c0 pchip
/c0 PCHIP Version = 1.30-33
```

info *cid achip*

This command reports the ACHIP (ATA Interface Chip) version of the specified controller *cid*.

Example:

```
//localhost> info c0 achip
/c0 ACHIP Version = 3.20
```

info *cid numports*

This command reports the number of ports of the specified controller *cid*.

Example:

```
//localhost> info c0 numports
/c0 Number of Ports = 12
```

info *cid numunits*

This command reports the number of units currently managed by the specified controller *cid*. This report does not include units that have been removed (placed off-line) with the maint remove command.

Example:

```
//localhost> info c0 numunits
/c0 Number of Units = 1
```

info *cid numdrives*

This command reports the number of drives currently managed by the specified controller *cid*. This report does not include units that have been removed (placed off-line) with the maint remove command.

Also note that a physically removed disk is not detected unless I/O is performed against the disk. See “info cid pid smart” on page 86 for a workaround.

Example:

```
//localhost> info c0 numdrives
/c0 Number of Drives = 5
```

info *cid unitstatus*

This command presents status of units managed by the specified controller *cid*. It provides a list of units, their types, current status, percent complete if rebuilding or verifying, size in GB, and the number of blocks of 512 bytes.

Example:

```
//localhost> info c0 unitstatus
Unit  UnitType Status %Cmpl  Stripe Size(GB)Cache AVerify OvrECC
-------------------------------------------------------------
u0    RAID-5 VERIFYING   79    16K  819.446   ON    OFF    ON
```

info *cid allunitstatus*

This command presents a count of Total and NotOptimal units managed by the specified controller *cid*. See “Info Commands” on page 77 for more information on NotOptimal.

Example:

```
//localhost> info c0 allunitstatus
Total Units = 2
NotOptimal Units = 0
```

info *cid drivestatus*

This command presents a list of port assignments, status, unit affiliation, size in GB, the number of blocks of 512 bytes, and the disk's serial number.

Example:

```
//localhost> info c0 drivestatus
Port     Status         Unit Size        Blocks      Serial
---------------------------------------------------------------
p0       OK             u0   149.05 GB   312581808   3JS0TF14
p1       OK             u0   149.05 GB   312581808   3JS0TETZ
p2       OK             u1   149.05 GB   312581808   3JS0VG85
p3       OK             u1   149.05 GB   312581808   3JS0VGCY
p4       OK             u1   149.05 GB   312581808   3JS0VGGQ
p5       OK             u2   149.05 GB   312581808   3JS0VH1P
p6       OK             -    149.05 GB   312581808   3JS0TF0P
p7       OK             -    149.05 GB   312581808   3JS0VF43
p8       OK             -    149.05 GB   312581808   3JS0VG8D
p9       NOT-PRESENT    -    -           -           -
p10      NOT-PRESENT    -    -           -           -
p11      NOT-PRESENT    -    -           -           -
```

info *cid exportjbod*

This command shows whether the Export JBOD policy is enabled or disabled.

Example:

```
//localhost> info c0 exportjbod
/c0 JBOD Export Policy = off
```

info *cid ondegrade*

This command shows whether the write cache will be disabled if a unit degrades.

Example:

```
//localhost> info c0 ondegrade
/c0 Cache on Degrade Policy = Follow Unit Policy
```

info *cid spinup*

This command shows whether staggered spinup is enabled.

Example:

```
//localhost> info c0 spinup
/c0 Disk Spinup Policy = 1
```

Note that "1" indicates enabled. When disabled, "255" is shown.

info *cid stagger*

This command shows the delay between drive groups that spin up at one time on this controller

Example:

```
//localhost> info c0 stagger
/c0 Spinup Stagger Time Policy (sec) = 2
```

info *cid uid*

This command presents detailed information on the specified unit. If the unit consists of sub-units as is the case in RAID 1, RAID 5, RAID 10, and RAID 50 arrays (applicable for 9000 controllers), then details about each sub-unit are also presented. One application of this command is to see which sub-unit of a degraded unit has caused the unit to degrade and which disk within that sub-unit is the source of degradation.

Example:

```
//localhost> info c0 u0

 Unit   UnitType  Status  %Cmpl  Port  Stripe  Size(GB)  Blocks
-------------------------------------------------------------
u0     RAID-5    VERIFYING  0     -     16K   819.446 718503424
u0-0   DISK      OK         -     p0    -     74.4951 156227584
u0-1   DISK      OK         -     p1    -     74.4951 156227584
u0-2   DISK      OK         -     p2    -     74.4951 156227584
u0-3   DISK      OK         -     p3    -     74.4951 156227584
u0-4   DISK      OK         -     p4    -     74.4951 156227584
u0-5   DISK      OK         -     p5    -     74.4951 156227584
u0-6   DISK      OK         -     p6    -     74.4951 156227584
u0-7   DISK      OK         -     p7    -     74.4951 156227584
u0-8   DISK      OK         -     p8    -     74.4951 156227584
u0-9   DISK      OK         -     p9    -     74.4951 156227584
u0-10  DISK      OK         -     p10   -     74.4951 156227584
u0-11  DISK      OK         -     p11   -     74.4951 156227584
```

info *cid uid status*

This command presents the status of the specified unit.

Example:

```
//localhost> info c0 u5 status
/c0/u5 status=DEGRADED
```

info *cid uid rebuildstatus*

This command presents the rebuild status (if any) of the specified unit.

Example:

```
//localhost> info c0 u5 rebuildstatus
/c0/u5 is not rebuilding.
```

Or, when the unit is rebuilding:

```
//localhost> info c0 u5 rebuildstatus
/c0/u5 is rebuilding with Percent Completion = %14
```

info *cid uid verifystatus*

This command presents the verify status (if any) of the specified unit.

Example:

```
//localhost> info c0 u5 verifystatus
/c0/u5 is not verifying.
```

info *cid uid initializestatus*

This command presents the initialize status (if any) of the specified unit.

Example:

```
//localhost> info c0 u5 initializestatus
/c0/u5 is not initializing.
```

info *cid pid*

This command presents various information on the specified disk attached to port *pid*. Typical information looks like:

Example:

```
//localhost> info p5
Port                                    Status        Unit Size
Blocks                                  Serial
---------------------------------------------------------------
p5                                      OK            u2 149.05
GB                                      312581808     3JS0VH1P
```

This report indicates that port 5 of controller 0 is attached to disk serial number 3JS0VH1P, with status OK participating in unit 5.

info *cid pid status*

This command presents the status of the specified port.

Example:

```
//localhost> info c0 p5 status
/c0/p5 Status = OK
```

info *cid pid model*

This command presents the model of the specified port.

Example:

```
//localhost> info c0 p5 model
/c0/p5 Model = WDC WD1600BB-00DAA0
```

info *cid pid serial*

This command presents the serial number of the specified port.

Example:

```
//localhost> info c0 p5 serial
/c0/p5 Serial = WD-WMACK1406498
```

info *cid pid capacity*

This command presents the capacity, both in human readable form (such as GB) and block count of the specified port. Note that capacity is computed based on division by 1024 (not 1000 as is popular with hard disk vendors). For additional information, see the explanation at “info cid” on page 78.

Example:

```
//localhost> info c0 p5 capacity
/c0/p5 Capacity = 149.05 GB (312581808)
```

info *cid pid smart*

This command extracts SMART (Self Monitoring Analysis and Reporting) data from the specified disk. The data is extracted live from the disk; therefore, this command is used to get the most recent data about the presence or absence of a disk. The SMART data is displayed in hexadecimal form. Since SMART data is extracted live from this disk, it places a burden on the I/O bandwidth.

Example:

```
//localhost> info c0 p5 smart
10 00 01 0B 00 C8 C8 00 00 00 00 00 00 00 03 07
00 9A 96 BC 14 00 00 00 00 00 04 32 00 64 64 7A
00 00 00 00 00 00 05 33 00 C8 C8 00 00 00 00 00
...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2C
```

info *cid diag*

This command extracts the internal log of diagnostic events of controller *cid*, controller diagnostics suitable for technical support usage. Note that some characters might not be printable or rendered correctly (human readable). It is recommended to save this output to a file, where it can be communicated to tech support.

Example:

```
 //localhost> info c0 diag

 --Port[ 5]-
 DIT status: DRV_PRESENT (0xFF)
  Model #: Maxtor 6Y080M0
  Serial #: Y3LLQWPE
  Drv FW #: YAR51EW0
  Capacity: 160086528 (0x098ABA00)
  Features: SMART: 1, Security: 1, 48-bit addr: 0, Acoustic: 1,
 Feat. Ext: TimeLimited R/W: 0, WDMA FUA: 0, Stream: 0
  Acoustic: 0xFE, def=0xC0 (0xFE=best performance)
  Security: Status=0x7 (ENABLED, LOCKED)
  SATA NCQ: 0
 Udma Mode: 0x5 (UDMA-100)
Pwr Cycles: 14
….
SELF TEST: all tests completed.
UA::SpareUnit(0) checking for spare needed
UA::SpareUnit done.
Auto Clean: 0
DcbMgr::UpdateStatus: UNIT  0 (time   7704019)
DcbMgr::WriteSegment(map=0xFFE, segID=0x8, events=22,
error=0x0)
DcbMgr::UpdateStatus:         (finish 7704024)
SETTINGS: Saving USER parameters ... done. (count=1, bytes=9)
SETTINGS: Saving USER parameters ... done. (count=1, bytes=9)
SETTINGS: Saving USER parameters ... done. (count=1, bytes=9)
SETTINGS: Saving USER parameters ... done. (count=1, bytes=9)
Send AEN (code, time): 0x31, 0x7a2d6b
Synchronize host/controller time
(EC:0x31, SK=0x00, ASC=0x00, ASCQ=0x00, SEV=04, Type=0x71)
```

Maint Commands

The *maint* command lets you perform maintenance operations on the controller, its units, and drives. It is recommended that you use the *info* command first to verify the controller information before using the *maint* command to make any changes to it.

Sub-commands under this category allow you to create and modify objects and their attributes such as creating and deleting logical units, rebuilding, etc. These commands are read/write operations and should be used with care.

Use of the keyword “maint” is now optional. For example, “`maint rescan c0`” is the same as “`rescan c0`”.

Syntax

```
[maint]        (Note: maint keyword is now optional)
   rescan [c<c> ...] [noscan] |NOTE: Does not import non-
     JBOD on 7000/8000 models.
  remove c<c> u<u> [noscan]
  remove c<c> p<p>
  deleteunit c<c> u<u> [noscan]
   createunit c<c> r<RaidType> p<p:-p..> [k<stripe>] [nos-
  can]  [nocache]  [g<3|4|5|6>]  [autoverify]  [ignoreECC]
  RaidType = { raid0, raid1, raid5, raid10, raid50, single,
  spare }
  rebuild c<c> u<u> p<p:-p...> [ignoreECC]
  rebuild c<c> u<u> pause  (7000/8000 only)
  rebuild c<c> u<u> resume  (7000/8000 only)
  flush c<c> [u<u>]
  verify c<c> u<u> [stop]
 mediascan c<c> start|stop (7000/8000 only)
 commit c<c> (** Windows only **)
```

[maint] *rescan* [cid ...] [noscan]

This command instructs the controller to rescan all ports, and reconstitute all units. The controller updates its list of ports (attached disks), and visits every Disk Configuration Block (DCB) in order to re-assemble its view and awareness of logical units.

If no controller is specified, all controllers are rescanned. One or several controllers can be specified.

By default, the OS is informed of changes resulting from rescan. You can alter this behavior using the noscan option.

Rescan imports JBOD units only when attached to either a 7000 or 8000 controller, unless you reboot. All other RAID types can be imported when attached to the 9000 series.

Warning!

Adding any drive requires use of an approved hot swap carrier. If you do not have such a carrier you must first power down your system. Failure to do so may cause the system to hang or become corrupted. It may even damage your system.

Example:

```
//localhost> maint rescan
Rescanning controller /c0 for units and drives ...Done.
Rescanning controller /c1 for units and drives ...Done.
```

If you use the noscan option:

```
//localhost>maint rescan c0 noscan
```

Using the noscan option allows a system administrator to export units to the OS a later time rather than having the CLI do it for them.

[maint] remove *cid uid* [noscan]

This command allows you to remove (or export) a unit. Exporting a unit instructs the firmware to remove the specified unit from its poll of managed units, but retains the Disk Configuration Block (DCB) metadata.You can import (re-introduce) the unit via rescan. By default the OS is informed of this change. You can alter this behavior using the noscan option.

Warning!

You must first unmount the array before issuing the maint remove command. Failure to do so may cause the system to hang or become corrupted.

Warning!

Physically removing any drive requires use of an approved hot swap carrier. If you do not have such a carrier you must first power down your system. Failure to do so may cause the system to hang or become corrupted. It may even damage your system.

[maint] remove *cid pid* [noscan]

This command allows you to remove (or export) a port (or drive). Exporting a port instructs the firmware to remove the specified port from its poll of managed ports, but retains the Disk Configuration Block (DCB) metadata on the attached disk. You can import (re-introduce) the port via rescan. By default the OS is informed of this change. If you use the noscan option the OS is not notified of the drive removal.

Warning!

Removing any drive requires use of an approved hot swap carrier. If you do not have such a carrier you must first power down your system. Failure to do so may cause the system to hang or become corrupted. It may even damage your system.

Alert!

Removing a drive causes a redundant array to degrade. Drives cannot be removed if they are part of a degraded or non-redundant array, with the exception of Single and JBOD drives.

Warning! Single and JBOD Drives

You must first unmount any Single or JBOD drive before issuing the remove command. Failure to do so may cause the system to hang or become corrupted.

[maint] deleteunit *cid uid* [noscan]

This command allows you to delete a unit. Deleting a unit not only removes the specified unit from the controller's list of managed units, but also destroys the DCB (Disk Configuration Block) metadata. Ports (or disks) associated with this unit will now be part of the free poll of managed disks. This is a destructive command and should be used with care. By default the OS is informed of this change. You can alter this behavior using the noscan option.

Warning! Back up data

Back up any critical data prior to deleting a unit. Failure to do so will result in lost data.

[maint] createunit *cid rRAIDType pid_list* [kStripe] [noscan] [Dsk_Grp] [nocache] [autoverify] [ignoreECC]

This command allows you to create a unit on the specified controller *cid*, of type *rRAIDType*, optional stripe size of *kStripe*, using one or many disks specified by *pid_list*. By default the host operating system is informed of the new block device and write cache is enabled. In case of RAID 50, you can also specify the layout of the unit by specifying the number of disks per disk group with the *gDsk_Grp* option.

cid is the controller name as in c0, c1, etc.

rRAIDType is the RAID or Logical Unit type as in RAID 0, RAID 1, RAID 5, RAID 10, RAID 50, single, spare, and JBOD. The following table illustrates supported types and controller models.

Table 13: Supported RAID Types

Model	R-0	R-1	R-5	R-10	R-50	Single	JBOD	Spare
7K/8K	Yes	Yes	Yes	Yes	N/A	N/A	Yes	Yes
9K	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

pid_list is a list of ports (disks) to be used in the construction of the specified unit. One or more ports can be specified. Multiple ports can be specified using a colon (:) to separate port indexes and a dash (-) to include a range of port indexes. A dash indicates a range and can be mixed with colons. For example p0:1:2-5:9:12 indicates port 0, 1, 2 through 5 (inclusive), 9 and 12.

kstripe indicates the stripe size to be used. The following table illustrates the supported and applicability of stripes on unit types and controller models. Stripe size units are in K (kilobytes).

Table 14: Supported Stripe Sizes

Model	R0	R1	R5	R10	JBOD	Spare	R50	Single
7K/8K	64	N/A	64	64	N/A	N/A	N/S	N/S
	128			128				
	256			256				
	512			512				
	1024			1024				
9K	16	N/A	16	16	N/A	N/A	16	N/A
	64		64	64			64	
	256		256	256			256	

gdsk_grp indicates the number of disks per group for a RAID 50 type. A RAID 50 is a multi-tier array. At the most bottom layer, N number of disks per group are used to form the RAID 5 layer. These RAID 5 arrays are then integrated into a RAID 0. This option allows you to specify the number of disks in the RAID 5 level. Valid values are 3, 4, 5 and 6.

Note that a sufficient number of disks are required for a given pattern or disk group. For example, given 6 disks, specifying 3 creates two RAID 5 arrays. However given 12 disks, specifying 3 creates four RAID 5 arrays under the RAID 0 level. Given 6 disks, specifying 6 is not allowed as you would basically be creating a RAID 5.

The default RAID 50 grouping **(gdsk_grp)** varies based on number of disks. For 6 and 9 disks, the default grouping is 3. For 8 disks the default grouping is 4. For 10 disks the default grouping is 5 and for 12 disks, the default grouping is 4. In the case of 12, disks could be grouped into groups of 3, 4, or 6 drives. A grouping of 4 is set by default as it provides the best of net capacity and performance.

noscan switch instructs CLI not to notify OS of the creation of the new unit. By default CLI informs the OS. One application of this feature is to avoid OS creating block special devices such as /dev/sdb, /dev/sdc as some implementations might create naming fragmentation and creating a moving target.

nocache switch instructs CLI not to enable the write cache. Enabling write cache increases performance at the cost of high-availability.

autoverify switch enables the autoverify attribute on the unit that is to be created. This feature is not supported on model 7000/8000. Autoverify is used in conjunction with the scheduling option. If autoverify is enabled, the array is verified repeatedly during a scheduled verify window. If autoverify is disabled, verify is not initiated by the controller and must be started manually.

ignoreECC switch enables the ignoreECC/OverwriteECC attribute on the unit that is to be created. The following table illustrates the supported Model-UnitType. This table only applies to setting this feature at Unit Creation time. Generally ignoreECC applies to redundant units.

Table 15: Supported Model-Unit Types

Model	R-0	R-1	R-5	R-10	R-50	Single	JBOD	Spare
7K/8K	No	No	No	No	No	No	No	No
9K	No	Yes	Yes	Yes	No	No	Yes	No

Examples:

To create a 12-member RAID-0 array with 128K stripe size on controller 0:

```
CLI> maint createunit c0 rraid0 k256 p0-11
```

To create a hot spare using a drive on port-2 controller-0 for automatic rebuilds:

```
CLI> maint createunit c0 rspare p2
```

Alert!

When creating a hot spare, be sure to select a drive with an equal or larger size than the smallest drive in your redundant array. Otherwise it can't be used in a rebuild.

[maint] rebuild *cid uid pid_list* [ignoreECC]

This command allows you to rebuild a DEGRADED unit by using the specified port. Rebuild applies only to redundant arrays such as RAID 1, RAID 5, RAID 10 and RAID 50.

During rebuild, bad sectors on the source disk cause the rebuild to fail. Using the ignoreECC option (equivalent to checking the 'Force continue on source errors' box in 3DM) allows the rebuild to continue when source errors occur, but a file system check is recommended once the rebuild is complete.

Note: The ignoreECC option is not required for the 9000 series, if the variable is already assigned when you create or set later. Refer to the "set overwriteECC cid uid on|off" on page 104 for more info.

The rebuild process is a background process and changes the state of a unit to REBUILDING. Various *info* commands also show a percent completion as rebuilding progresses.

Ports that are to be used to rebuild a unit must be a Spare type or an unconfigured drive. You must first use `remove` to remove the failed drive, then use `rescan` to add the new drive before you can use `rebuild`.

[maint] rebuild *cid uid* pause

This command allows you to pause the rebuild operation on the specified unit. This feature is intended for model 7000 and 8000 only. Model 9000 has an on-board scheduler where rebuild operations can be scheduled to take place at specified start and stop times. Rebuild pause and resume function is provided to enable 7000/8000 users to achieve similar functionality with the use of OS-provided schedulers.

See also "Sched Commands" on page 96.

[maint] rebuild *cid uid* resume

This command allows you to resume the rebuild operation on the specified unit. See "[maint] rebuild cid uid pause" on page 93 for more details.

[maint] flush *cid* [*uid* ...]

This command allows you to flush the write cache on the specified unit or all units associated with controller *cid*. This command does not apply to Spare unit types.

[maint] verify *cid uid* [stop]

This command starts or stops a background verification process on the specified unit. To start, omit “stop”.

The following table shows the supported RAID types for verification as a function of controller model and logical unit type. N/A (Not Applicable) refers to cases where the given logical unit type is not supported on a particular controller model.

Table 16: Supported RAID (Logical Unit) Types for Verification

Model	R-0	R-1	R-5	R-10	R-50	Single	JBOD	Spare
7K/8K	No	Yes	Yes	Yes	N/A	N/A	No	No
9K	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

[maint] mediascan *cid* start|stop

This command applies to 7000/8000 controllers. It provides media scrubbing for validating the operationality of a disk, including bad block detection and remapping. The start or stop operations start or stop media scan operation on the specified controller. For the 9000 series, the verify function includes the features of the media scan command.

[maint] commit *cid*

This command only applies to the Windows operating system.

This command instructs the controller to commit its dirty DCBs to persistent storage (disks). While the controller is processing I/O requests against underlying disks, an in-transaction bit is set. If a failure (such as power failure) is experienced, subsequent reads from the disk inform the controller that an un-clean shutdown took place. This command allows the end user to complete all pending I/Os on disks and clear the in-transaction bit.

Typical application of this feature is when an application is using a given unit in raw mode (such as databases) and the user would like to shut down the host (including UPS post failure automations). This command expedites the process by instructing the controller to finish pending requests and clear the DCBs in-transaction flag as the disk is going down.

Note that block devices (cooked devices) do not require this command. Clients of block devices (such as File system) send such requests via ASPI SRB_SHUTDOWN_REQUEST.

Sched Commands

Sched commands are applicable for 9000-series controllers to schedule background tasks to occur at a later time or day, when peak performance is not required. When the schedule is disabled, by default, background task occur almost immediately. Background tasks include rebuild, verify, and selftest activities. For each activity, up to 7 tasks can be registered, known as slots 0 through 6. Each task can be managed with these commands including adding, removing, enabling and disabling a task. Background tasks have slot id, category (rebuild, verify, selftest), start time, and duration attributes.

rebuild activity is the process of rebuilding one or more DEGRADED units to one or more specified ports. Rebuild applies only to redundant arrays such as RAID 1, RAID 5, RAID 10 and RAID 50. Initialization activity of new arrays is also included with this background task.

verify activity visits every unit of a given controller and attempts to verify all members of redundant units. On the 9000 series, non-redundant units, including spares, are also verified by doing a background scrub which reads each sector. Verifying RAID 1 involves checking that both drives contain the exact data. On RAID 5, the parity information is used for error correction. RAID 10 and 50 are composite types and follow their respective array types.

selftest activity provides two types of selftests: Ultra Direct Memory Access (UDMA) and Self Monitoring Analysis and Reporting (SMART). UDMA selftest checks the current ATA bus speed (between the controller and an attached disk) which could have been throttled down during previous operations and increases the speed for best performance (usually one level higher). Possible speeds include 33, 66, 100 and 133 MB/s.

SMART activity instructs the controller to check certain SMART-supported thresholds by the disk vendor. The UDMA selftest is not required for serial ATA drives.

Syntax

```
sched rebuild c<c>
sched rebuild c<c> add d<d> h<h> t<t>
sched rebuild c<c> remove <n>
sched rebuild c<c> enable|disable
sched verify c<c>
sched verify c<c> add d<d> h<h> t<t>
sched verify c<c> remove <n>
sched verify c<c> enable|disable
sched selftest c<c>
sched selftest c<c> add d<d> h<h>
sched selftest c<c> remove <n>
sched selftest c<c> enable|disable s<s>
```

sched rebuild *cid*

This command displays the current rebuild background tasks as illustrated below.

```
//localhost> tw_cli sched rebuild c1
Rebuild Schedule for controller /c1
=======================================================
Slot    Day     Hour            Duration        Status
-------------------------------------------------------
0       Mon     2:00pm          10 hr(s)        disabled
1       Thu     7:00pm          18 hr(s)        disabled
2       -       -               -               -
3       -       -               -               -
4       -       -               -               -
5       Mon     1:00am           4 hr(s)        disabled
6       Sun     12:00am          1 hr(s)        disabled
```

sched rebuild *cid* add *day hour duration*

This command adds a new background rebuild task to be executed on day (range 0 .. 6, where Sunday is zeroth day of the week), at hour (range 0 .. 23), for a duration of duration (range 1 .. 24) hours. This command will fail if no (empty) slot is available.

Note: The new schedule is added to the first available slot. Events do not need to be added in sequential order

For example:

```
//localhost> tw_cli sched rebuild c1 add d0 h16 t3
```

Adds a rebuild background task to be executed on Sundays at 4:00 PM for a duration of 3 hours.

sched rebuild *cid* remove slot_id

This command removes (or unregisters) the rebuild background task in slot slot_id.

Warning: If all timeslots are removed, be sure to also disable the schedule. Otherwise the applicable background task will never occur.

For example:

```
//localhost> tw_cli sched rebuild c1 remove 2
```

Removes the rebuild background task in slot 2.

sched rebuild *cid* enable

This command enables ALL rebuild background tasks on controller *cid*.

sched rebuild *cid* disable

This command disables ALL rebuild background tasks on controller *cid*.

sched verify *cid*

This command displays the current verify background task as illustrated below.

```
//localhost> tw_cli sched verify c1
Verify Schedule for controller /c1
=======================================================
Slot    Day     Hour            Duration        Status
-------------------------------------------------------
0       Mon     2:00am          4 hr(s)         disabled
1       -       -               -               -
2       Tue     12:00am         24 hr(s)        disabled
3       Wed     12:00am         24 hr(s)        disabled
4       Thu     12:00am         24 hr(s)        disabled
5       Fri     12:00am         24 hr(s)        disabled
6       Sat     12:00am         24 hr(s)        disabled
```

sched verify *cid* add day hour duration

This command adds a new background verify task to be executed on day (range 0 .. 6, where Sunday is zeroth day of the week), at hour (range 0 .. 23), for a duration of duration (range 1 .. 24) hours. This command will fail if no (empty) slot is available.

For exmaple:

```
//localhost> tw_cli sched verify c1 add d0 h16 t3
```

Adds a verify background task to be executed on Sundays at 4:00 PM for a duration of 3 hours.

Note: The new schedule is added to the first available slot. Events do not need to be added in sequential order.

sched verify *cid* remove *slot_id*

This command removes (or unregisters) the verify background task in slot *slot_id*.

For example:

```
//localhost> tw_cli sched verify c1 remove 3
```

Removes the verify background task in slot 3.

Warning: If all timeslots are removed, be sure to also disable the schedule. Otherwise the applicable background task will never occur.

sched verify *cid* enable

This command enables all verify background tasks on controller *cid*.

Note: When enabling the verify schedule you must also remember to also enable the autoverify setting for the arrays to be verified. For details, see "set autoverify cid uid on|off" on page 104.

sched verify *cid* disable

This command disables all verify background tasks on controller *cid*.

sched selftest *cid*

This command displays the current selftest background task as illustrated below.

```
//localhost> tw_cli sched selftest c1
Selftest Schedule for controller /c1
=======================================================
Slot    Day     Hour            UDMA            SMART
-------------------------------------------------------
0       Sun     12:00am         enabled         enabled
1       Mon     12:00am         enabled         enabled
2       Tue     12:00am         enabled         enabled
3       Wed     12:00am         enabled         enabled
4       Thu     12:00am         enabled         enabled
5       Fri     12:00am         enabled         enabled
6       Sat     12:00am         enabled         enabled
```

sched selftest *cid* add *day hour*

This command adds a new background selftest task to be executed on day (range 0 .. 6, where Sunday is zeroth day of the week), at hour (range 0 .. 23). Notice that selftest runs to completion and as such no duration is provided. This command fails if no (empty) slot is available.

For example:

```
$ tw_cli sched selftest c1 add d0 h16
```

Adds a selftest background task to be executed on Sundays at 4:00 PM.

Note: The new schedule is added to the first available slot. Events do not need to be added in sequential order. Also the selftests are completed almost

sched selftest *cid* remove *slot_id*

This command removes (or unregisters) the selftest background task in slot *slot_id.*

For example:

```
//localhost> tw_cli sched selftest c1 remove 3
```

Removes rebuild selftest task in slot 3.

Warning: If all timeslots are removed, be sure to also disable the schedule. Otherwise the applicable background task will never occur

sched selftest *cid* enable *selftest_task_id*

This command enables a particular selftest_task (UDMA or SMART). Selftest_task_id s0 is interpreted as UDMA; Selftest_task_id s1 is interpreted as SMART.

Note: When enabling the verify schedule you must also remember to also enable the autoverify setting for the arrays to be verified.

For example:

```
//localhost> tw_cli sched selftest c1 enable s0
```

Enables UDMA selftest on controller c0.

sched selftest *cid* disable *selftest_task_id*

This command disables a particular selftest task (UDMA or SMART). For the selftest_task_id, s0 is interpreted as UDMA, s1 is interpreted as SMART.

For example:

```
//localhost> tw_cli sched selftest c1 disable s1
```

Disables SMART selftest on controller c1.

Alarms Commands

The *alarms* command provides a log of alarms, also called Asynchronous Event Notifications (AENs), that have occurred on the disk arrays. An alarm occurs when the ATA RAID controller requires attention, such as when a disk array becomes degraded and is no longer fault tolerant. SMART notifications appear in this display. Alarm messages are categorized by the following levels of severity:

- Errors
- Warnings
- Information

When the *alarms* command is executed, only AENs that have been logged since the last time the command was executed are displayed. For Linux, AENs are also saved in a text file at `/var/log/messages`.

Windows users can see the AEN messages in the Windows System Event Logs that can be seen in the Event Viewer.

Asynchronous events are originated by firmware and captured by their respective device drivers. These events are kept in a finite queue inside the kernel, awaiting extraction by user programs such as CLI or 3DM 2. These events reflect warning, debugging, or informative messages for the end user.

Alarms generated on 7000/8000 models do not have dates, so a dash (-) meaning *not-applicable* appears in the Date column. Also on 7000/8000 models, the alarm message does not contain the severity, hence the Severity column displays a dash (-) as well.

Syntax

```
alarms
alarms c<c> [c<c> ...]
```

Warning!

3ware does not recommend installing both 3DM and CLI. Conflicts may occur. If both are installed, alarms will be captured only by 3DM.

Warning!

3DM and CLI handle alarms differently. When using CLI with the 7/8000 series, save the alarm data immediately after viewing it. Once the alarms are viewed using CLI, they cannot be viewed again.

With the 9000 series, the alarms can be viewed multiple times with the CLI, but will be lost following a reboot. At this time there is no CLI option to clear the alarms log.

alarms [*cid* ...]

This command displays all available alarms on a single controller, multiple controllers, or on all controllers. Invoked without *cid*, displays alarms associated with all detected controllers.

Note: A listing of AEN codes can be found in the "Troubleshooting: Problems and Solutions" section of *3ware 9000 Series Serial ATA RAID Controller User Guide*.

Typical output looks like:

```
tw_cli> alarms

Ctl  Date                  Severity  Message
---------------------------------------------------
c0   -              -         ERROR: Unit degraded: Unit #0
c1 [Fri Nov 28 04:26:31 2003] ERROR (0x04:0x0002): Degraded unit detected: unit=0, port=2
c1 [Fri Nov 28 06:13:54 2003] INFO  (0x04:0x000B): Rebuild started: unit=0
c1 [Fri Nov 28 06:30:35 2003] INFO  (0x04:0x003B): Background rebuild paused: unit=0
c1 [Fri Nov 28 06:33:00 2003] ERROR (0x04:0x0002): Degraded unit detected: unit=0, port=0
c1 [Fri Nov 28 06:33:04 2003] ERROR (0x04:0x0002): Degraded unit detected: unit=0, port=4
c1 [Fri Nov 28 06:33:46 2003] INFO  (0x04:0x000B): Rebuild started: unit=0
c1 [Fri Nov 28 06:37:58 2003] INFO  (0x04:0x000B): Rebuild started: unit=0
c1 [Fri Nov 28 07:51:34 2003] INFO  (0x04:0x0005): Background rebuild done: unit=0
c1 [Fri Nov 28 07:59:43 2003] INFO  (0x04:0x0005): Background rebuild done: unit=0
c1 [Mon Dec  1 02:26:12 2003] ERROR (0x04:0x0002): Degraded unit detected: unit=0, port=3
```

Set Commands

These commands allow you to set certain controller and unit specific parameters as described below. The set command can be used to set its rebuild rate, and enable or disable cache. For information about viewing information about the controller and units, see the “Info Commands” on page 77.

Syntax

```
set rebuild c<c> <1..5>
set cache c<c> u<u> on|off
set verify  c<c> <1..5>                (Note: 9000 only)
set autoverify c<c> u<u> on|off        (Note: 9000 only)
set overwriteECC c<c> u<u> on|off      (Note: 9000 only)
```

Parameters

- *rebuild* - sets the rebuild rate (per controller basis) and also sets the initialize rate.
- *cache* - enables or disables caching on a per array or unit basis for RAID 1, 5, and 10 arrays.
- *verify* - sets the verify rate (per controller basis)
- *autoverify* - enables or disables the automatic verification (per unit basis)
- *overwriteECC* - enables or disables the ignoreECC function dring rebuild (per unit basis)

Note: A value of 1 indicates slowest I/O and fastest rebuild rate. A value of 5 indicates fastest I/O and slowest rebuild. Interim values scale linearly (e.g., a value of 3 indicates a rebuild rate half as fast as a rebuild of 1).

set rebuild *cid 1..5*

This command allows you to set the priority of rebuild in relation to I/O operations. Setting this value to 1 implies that rebuilds should consume more resources (cpu time, I/O bandwidth) to complete its task. Conversely, setting this value to 5 implies that I/O has higher priority and rebuild. This command applies to 7000, 8000, and 9000 models. For the 7/8000 series, the rebuild rate also applies to the verify and background scrub tasks.

set verify *cid 1..5*

This command allows you to set the priority of verification in relation to I/O operations. Setting this value to 1 implies fastest verify, and 5 implies fastest I/O. Note that This feature only applies to 9000 series controllers.

set cache *cid uid on|off*

This command allows you to turn on or off the write cache on a specified unit. This feature is supported on both 7000/8000 and 9000 models.

The following table shows the supported RAID types for caching as a function of controller model and logical unit type. N/A (Not Applicable) refers to cases where the given logical unit type is not supported on a particular controller model.

Table 17: Supported RAID Types for Caching

Model	R-0	R-1	R-5	R-10	R-50	Single	JBOD	Spare
7K/8K	Yes	Yes	Yes	Yes	N/A	N/A	Yes	No
9K	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

set autoverify *cid uid on|off*

This command allows you to turn on or off the verify operation on a specified unit during times specified by sched verify commands. The sched verify command allows you to specify times for the verify operation, it does not associate the operation with a unit. This command allows you to associate a unit with the verify operation. This feature only applies to 9000 series controllers.

set overwriteECC *cid uid on|off*

Setting overwriteECC to on means ignoreEcc. This command allows you to set the ignoreECC policy for a given unit. This policy is then automatically enforced during a rebuild of the specified unit. This option can also be specified when rebuilding the array manually as discussed in the maintenance section. If this setting is already enabled you do not need to specify it again when rebuilding an array. This setting only applies to 9000 models.

Help Commands

This command set provides brief on-line help.

help

This command provides a table of contents, providing brief descriptions of the help sub-commands. Typical output looks like:

```
 Copyright (c) 2003 3ware, Inc.  All rights reserved.
 List of Commands
 ----------------------------------------
 info   - displays information about the controller
 alarms - displays or deletes the list of AENs
 set    - displays or modifies controller settings
 maint  - performs maintenance operations on a controller
 sched  - Sets the schedule for a controller (9000 controllers
only)
 quit   - exits the CLI
 Type help <command> to get more details about a particular
command.
```

help info

This command provides specific *info* related help, illustrating various ways to use the info command. Info provides reports on 3ware controllers, units and drives.

help alarms

This command provides specific *alarms* related help, illustrating various ways to use the alarms command.

help set

This command provides specific *set* related help, illustrating various ways to use the set command.

help maint

This command provides specific *maint* related help, illustrating various ways to use the maint command.

help sched

This command provides specific *sched* related help, illustrating various ways to use the sched command. Applies to the 9000 version only.

help quit

This command provides information about the CLI *quit* command. For example:

```
//localhost> help quit
This command quits the CLI

  quit

    Synonyms: q exit
```

Return Code

While informative messages are written to standard output, error messages are written to standard error. On success, 0 is returned. On failure, 1 is returned.

To view the return code, at the shell command prompt type:

```
echo $?
```

The screen prints either a 0 or a 1, depending on whether the command was successful or not.

For example, if you had a 3ware controller with an ID of 0, you could type this command:

```
tw_cli info c0
(c0 information displayed here)
echo $?
0
```

If you type:

```
tw_cli info c7
error: (CLI003) specified controller does not exist.
echo $?
1
```

This example fails (returns 1) because there is no controller 7.