NAME

`tw_cli(8)` - AMCC/3ware Storage Controller Management Command Line Interface.

SYNOPSIS

```
tw_cli              Interactive Mode
tw_cli -f file      Process from a file
tw_cli command      Process single command (batch mode)
```

DESCRIPTION

tw_cli(8) is a Command Line Interface Storage Management Software for AMCC/3ware ATA RAID Controller(s). It provides controller, logical unit and drive management. tw_cli can be used in both interactive and batch mode, providing higher-level API (Application Programming Interface) functionalities.

As a way of synchronizing terminologies, logical units (a.k.a. units) 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. A controller has one or many ports (typically 4, 8, 12). Each port is connected to zero or one disk drive.

CLI also supports *comments*. Command lines beginning with # denotes start of comment. This feature is mostly useful with batch processing via *-f script* flag.

CLI prompt indicates the current object in focus, expressed in URI (Universal Resource Identifier) syntax consisting of a hostname (*//hostname*), and an object path (*/path/path/object*) such as *//elvis/c0/u0*. User can set the focus to a particular object by *focus URI*.

In this version of the CLI release, the tw_cli supports a set of primary command syntax (so called new command syntax) and a set of legacy command syntax (so called old command syntax or original command syntax). Note: we reserve the right to discontinue the legacy command syntax in the future releases.

Please also note that some of the commands listed in this document are qualified with restrictions of controller type/model support. For example, ``9000 series'' or ``9KSX/SE only'' may be next to a command. The following is a summary of the qualifications.

Commands with:

```
  No qualifications     Could be used across all controller platforms.  This includes
                        the 7000 and 8000 series controllers.
  9000 series           Could be used in all controllers in the 9000 series.  This
                        excludes the 7000 and 8000 series controllers, and includes
                        the 9550SX, 9590SE and 9650SE controllers.
  9KSX/SE only          Could be used in all controllers of the 9xxxSX or SE only.
```

The above implies an ordering of the command set hierarchy, where ``No qualification'' is the superset of ``9000 series'' that in turn is the superset of ``9KSX/SE only''.

For the Mac OS-X system, while still true, the command qualifications is not meaningful as all commmands are supported, provided the controller model is 9590SE or 9650SE (or above).

Primary Command Syntax

The primary command syntax will replace the legacy command syntax in the future releases. This new and improved command format follows a general grammar in the form:

```
Object Message Attributes
```

Objects are either shell commands or specify a certain controller, logical unit, port (drive), or battery backup unit (bbu). Messages are commands sent to the requested objects and will be a read operation such as show or write operation such as set, delete, add, stop, start, or remove. Attributes specify what values to read or write. Attributes are either *Boolean Attributes* or *Named Attributes*. Value of a Boolean attribute is deduced by presence or not. Value of named attributes are expressed in a ``key = value" format.

Shell Object Messages

Shell Object Messages are commands (a.k.a. methods/messages) that are sent to the Command Interpreter (a.k.a. Shell/CLI) itself.

show

This command shows a general summary of all detected controllers. Note that the appropriate kernel device drivers should be loaded for the list to show all controllers. The intention is to provide a global view of the environment.

Typical output 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
```

The output indicates 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, a 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 INITIALIZING, INIT-PAUSED, REBUILDING, REBUILD-PAUSED, DEGRADED, MIGRATING, MIGRATE-PAUSED, RECOVERY, INOPERABLE, and UNKNOWN.

For a system with an enclosure unit and support that includes the EPCT (Enclosure Port Configuration Table), applicable firmware and software, and the appropriate controllers (9550SX, 9590SE, or 9650SE), a global view of the environment also includes summary information about detected enclosures.

For example, for a typical system with an enclosure:

```
//localhost> show

Ctl    Model          Ports   Drives  Units   NotOpt   RRate   VRate   BBU
----------------------------------------------------------------------------
c0     Geroni133/Ap   4       2       1       0        4       4       -

Encl    Slots   Drives  Fans   TSUnits   Ctls
---------------------------------------------
e0      4       2       1      1         1
```

The enclosure summary information shows the name of the enclosure, the number of drive slots, the number of drives, the number of fans, the number of temperature sensors, and the number of controllers that are associated with the enclosure.

show ***ver***

This command will show the CLI and API version.

For example:

```
//localhost> show ver

CLI Version = 2.00.03.018
API Version = 2.01.00.004
CLI Compatible Range = [2.00.00.001 to 2.00.03.018]

//localhost>
```

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 offers the display order to be 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 of all the 9000 controllers in the system.

show ***verify***

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

show ***selftest***

This command displays all self test schedules of all the 9000 controllers in the system.

update ***fw=filename_with_path*** **[*force*]**

This command iterates through all the controllers in the system and downloads the specified firmware image to the architectural compatible controllers. Please refer to command */cx update fw=filename_with_path [force]* for detail.

focus ***Object***

This command will set the specified object in focus. This command is active in interactive mode only and is provided to reduce typing. Recall that messages (or commands) are sent to objects such as

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

Instead, if the focus is set to *//hostname/c0/u0*, the prompt is changed automatically to reflect this and the user would only have to type *show*. The concept is similar to being in a particular location in a file system and requesting a listing of the current directory.

object can have the following forms:

//hostname/cx/ux specifies the fully qualified URI of an object on host **hostname**, controller **cx**, unit **ux**.

//hostname specifies root of host **hostname**. The hostname is the name of the system where your 3ware RAID controllers are. With current releases, the hostname here should be always your system's name.

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

/ specifies the root at the current focused host.

./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>
```

Note that *focus* is available as default. You can also set *TW_CLI_INPUT_STYLE=OLD* in the following to disable the feature.

```
 If Bash,    then "export TW_CLI_INPUT_STYLE=OLD"
 If csh,     then "setenv TW_CLI_INPUT_STYLE OLD"
 If Windows, then "set    TW_CLI_INPUT_STYLE=OLD"
```

Controller Object Messages

Controller Object Messages are commands (a.k.a. methods/messages) that are sent to an instance of a controller such as */c0*.

***/cx* show**

This command shows summary information on the specified controller */cx*. This report consists of two to three parts; a **Unit** summary listing all present units, a **Port** summary section listing of all present disks and their attached ports, and a **BBU** summary section listing if a BBU unit is installed on the controller.

The **Unit** summary section lists all present units specifying their Unit Number, Unit type (such RAID 5), the unit status (such as OK, VERIFYING, INITIALIZING, etc.), the **%RCompl** which reports percent completion, the **%V/I/M** which reports the percent completion of Verify, Initialize, or Migrating status of the unit, the stripe size, the usable capacity in Giga (or Tera) Bytes, the write cache state setting, and the autoverify setting.

Note: If a ``*'' appears at the end of the status, it means one of its sub-unit is in WARNING state which a recoverable error might be found in the corresponding port.

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

The **BBU** summary section lists 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, and its readiness, etc.

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

Typical output looks like:

```
 //localhost> /c2 show

 Unit  UnitType  Status         %RCmpl  %V/I/M  Stripe  Size(GB)  Cache  AVrfy
 ------------------------------------------------------------------------------
 u0    RAID-5    OK             -       -       64K     596.004   ON     OFF
 u1    RAID-0    OK             -       -       64K     298.002   ON     OFF
 u2    SPARE     OK             -       -       -       149.042   -      OFF
 u3    RAID-1    OK             -       -       -       149.001   ON     OFF

 Port   Status           Unit   Size        Blocks        Serial
 ---------------------------------------------------------------
 p0     OK               u0     149.05 GB   312581808     WD-WCANM1771318
 p1     OK               u0     149.05 GB   312581808     WD-WCANM1757592
 p2     OK               u0     149.05 GB   312581808     WD-WCANM1782201
 p3     OK               u0     149.05 GB   312581808     WD-WCANM1753998
 p4     OK               u2     149.05 GB   312581808     WD-WCANM1766952
 p5     OK               u3     149.05 GB   312581808     WD-WCANM1882472
 p6     OK               u0     149.05 GB   312581808     WD-WCANM1883862
 p7     OK               u3     149.05 GB   312581808     WD-WCANM1778008
 p8     OK               -      149.05 GB   312581808     WD-WCANM1770998
 p9     NOT-PRESENT      -      -           -             -
 p10    OK               u1     149.05 GB   312581808     WD-WCANM1869003
 p11    OK               u1     149.05 GB   312581808     WD-WCANM1762464

 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 given *attribute(s)*. One or many attributes can be requested. An invalid attribute will terminate the loop. Possible attributes are: achip, allunitstatus, autocarve(9000 series), autorebuild(9KSX/SE only), bios, carvesize(9000 series), driver, drivestatus, exportjbod, firmware, memory, model, monitor, numdrives, numports, numunits, ctlbus(9KSX/SE only), ondegrade(9000S only), pcb, pchip, serial, spinup, stagger, and unitstatus.

/cx* show *driver

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

Example:

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

_/cx_ show _model_

This command reports the controller model of controller */cx*.

Example:

```
//localhost> /c0 show model
/c0 Model = 7500-12
```

_/cx_ show _firmware_

This command reports the firmware version of controller */cx*.

Example:

```
//localhost> /c0 show firmware
/c0 Firmware Version = FE9X 3.03.06.X03
```

_/cx_ show _bios_

This command reports the BIOS version of controller */cx*.

Example:

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

_/cx_ show _monitor_

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

Example:

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

_/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) revision of the specified controller */cx*.

Example:

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

_/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 = 1.30-33
```

_/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 = 3.20
```

_/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 removed 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/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** for a workaround.

Example:

```
//localhost> /c0 show numdrives

/c0 Number of Drives = 5
```

_/cx_ show _exportjbod_ (9000 series)

This command presents 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* (9000 series)**

This command presents the number of concurrent disks spin up at the power on.

Example:

```
//localhost> /c0 show spinup

/c0 Disk Spinup Policy = 1
```

***/cx* show *ondegrade* (9000S only)**

This command presents 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 force to be off when the unit becomes degraded.

Example:

```
//localhost> /c0 show ondegrade

/c0 Cache on Degraded Policy = Follow Unit Policy
```

***/cx* show *stagger* (9000 series)**

This command presents 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 *autocarve* (9000 series)**

This command shows the Auto-Carving policy. If the policy is on, all newly created or migrated units larger than carvesize will be automatically carved into multiples of carvesize volumes and 1 remainder volume. Each volume can be treated as an individual disk with its own file system. The default carvesize is 2 TB.

This feature is useful for operating systems limited to 2 TB filesystems. For 64-bit OS users, there is no need to set the policy to be ``on'' unless users want to have multiple smaller volumes to the OS. For 32-bit OS users, it is recommended to keep the policy on unless users know their OS supports more than 2 TB disk devices.

When autocarve policy is off, all the new unit creation consists of one single volume.

Example:

```
//localhost> /c0 show autocarve

/c0 Auto-Carving Policy = on
```

***/cx* show *carvesize* (9000 series)**

This command shows the carvesize that Auto-Carving policy needs. The carve size is between 1024 to 2048 GB. Default carvesize is 2048 GB (i.e. 2 TB). See ``*/cx* **show** *autocarve*'' command above for details.

Example:

```
//localhost> /c0 show carvesize

/c0 Auto-Carving Size = 2000 GB
```

/cx* show *memory

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

Example:

```
//localhost> /c0 show memory

/c0 Memory Installed = 112MB
```

***/cx* show *ctlbus* (9KSX/SE only)**

This command presents the controller host bus type, bus speed and bus width.

Example:

```
//localhost> /c0 show ctlbus

/c0 Controller Bus Type = PCIX
/c0 Controller Bus Width = 64 bits
/c0 Controller Bus Speed = 133 Mhz
```

***/cx* show *autorebuild* (9KSX/SE only)**

This command shows the Auto-Rebuild policy. If the policy is enabled, the firmware will choose following drives in order to find a candidate for rebuild operation on a degraded unit.

1. Smallest usable capacity spare.

2. Smallest usable unconfigured drive.

3. Smallest usable capacity failed drive.

If the policy is disabled, spare drives are the only candidates for an automatic rebuild operation.

Example:

```
//localhost> /c0 show autorebuild

/c0 Auto-Rebuild Policy = on
```

/cx* show *unitstatus

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

Example:

```
//localhost> /c2 show unitstatus

Unit  UnitType  Status         %RCmpl  %V/I/M  Stripe  Size(GB)  Cache  AVrfy
------------------------------------------------------------------------------
u0    RAID-5    OK             -       -       64K     596.004   ON     OFF
u1    RAID-0    OK             -       -       64K     298.002   ON     OFF
u2    SPARE     OK             -       -       -       149.042   -      OFF
u3    RAID-1    OK             -       -       -       149.001   ON     OFF
```

/cx* show *allunitstatus

This command presents a count of Total and Not Optimal units managed by the specified controller */cx*. See Shell Object Messages for more on Not Optimal definition.

Example:

```
//localhost> /c0 show allunitstatus

/c0 Total Optimal Units = 2
/c0 Not Optimal Units = 0
```

/cx* show *drivestatus

This command presents a list of drives, port assignment, vendor signature, size, status, and unit membership/affiliation.

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 attributes.

***/cx* add type=<RaidType> disk=<p:-p> [stripe=Stripe] [noscan] [group=<3|4|5|6|7|8>] [nocache] [autoverify] [noqpolicy] [ignoreECC] [name=string] [storsave=<protect|balance|perform>]**

This command allows you to add a new unit or create a unit on the specified controller */cx*, of type *RaidType*, optional stripe size of *Stripe*, using one or many disks specified by *disk=p:-p*. By default the host operating system will be 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 *group=3|4|5|6|7|8* attribute.

Upon the success of the new unit creation, a unique serial number is also assigned to the new unit. Please refer to commands */cx/ux show serial* for checking.

Please Note: 1) The default of the unit creation sets write cache to ``on'' for performance reasons. However, if there is no BBU available for the controller, a warning is sent to standard error. 2) The default drive queuing policy is enabled, unless it is specifically set to disable queuing by specifing *noqpolicy*. 3) The *noqpolicy* attribute is not applicable to the ``spare'' unit. Specifying the noqpolicy attribute returns an error.

Since this command is by far the richest command, it deserves more details.

/cx is the controller name as in /c0, /c1, etc.

type=RaidType consists of logical unit type as in **raid0**, **raid1**, **raid5**, **raid10**, **raid50**, **single**, **spare**, **JBOD (7000/8000 only)**, and **raid6 (9650SE only)**.

For example type=raid50.

The following table illustrates supported types and controller models.

Model	Raid0	Raid1	Raid5	Raid10	JBOD	Spare	Raid50	Single	Raid6
7K/8K	Y	Y	Y	Y	Y	Y	N	N	N
9K	Y	Y	Y	Y	N	Y	Y	Y	N
9650SE	Y	Y	Y	Y	N	Y	Y	Y	Y

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 ``**:**'' or ``**-**'' as port index separators. A dash indicates a range and can be mixed with ``:''. For example **disk=0:1:2-5:9:12** indicates port 0, 1, 2 thru 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 of units are in K (kilo bytes).

Model	Raid0	Raid1	Raid5	Raid10	JBOD	Spare	Raid50	Single	Raid6
7K/8K	64 128 256 512 1024	N/A	64	64 128 256 512 1024	N/A	N/A	N/S	N/S	N/S
9K	16 64 256	N/A	16 64 256	16 64 256	N/A	N/A	16 64 256	N/A	
9650SE	16 64 256	N/A	16 64 256	16 64 256	N/A	N/A	16 64 256	N/A	64

group=3|4|5|6|7|8 consists of the number of disks per group for a Raid 50 type. Note, this attribute can only be used when type=raid50.

Recall that 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 attribute allows you to specify the number of disks in the RAID-5 level. Valid values are 3, 4, 5, 6, 7 and 8.

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. However given 12 disks, specifying 3 will create four RAID-5 under the RAID-0 level. Given 6 disks and grouping of 6 is not allowed, as you'll basically be creating a RAID-5.

The default group varies based on number of disks. For 6 & 9 disks, default is group=3. For 8 disks, default is group=4. For 10 or 15 disks, default is group=5. For 12 or 16 disks, default is group=4. For 14 disks, default is group=7. Case of 12 disks could be grouped with group=3, group=4, or group=6. Group=4 was set by default as it provides best net capacity and performance. Case of 15 disks could be grouped with group=3 or group=5. And case of 16 disks could be grouped with group=4 and group=8.

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

nocache attribute instructs CLI disable the write cache on the newly created unit. Enabling write cache increases performance at the cost of high-availability. No cache is recommended when no BBU or UPS is installed.

autoverify attribute enables the autoverify attribute on the unit that is to be created. For more details on this feature, refer to cx/ux set Commands section of this document. This feature is not supported on model 7000/8000. On model 9000, JBOD autoverify attribute is not persistent (does not survive reboots).

noqpolicy attribute instructs CLI to disable the qpolicy (drive queuing) on the newly created unit. The default qpolicy is *on* (i.e., noqpolicy is not specified). For the spare unit, drive queueing is not meaningful and the qpolicy cannot be set. During unit creation, specifying *noqpolicy* for spare returns an error.

ignoreECC attribute enables the ignoreECC/OverwriteECC attribute on the unit that is to be created. For more details on this feature, refer to cx/ux set Commands section of this document. The following table illustrates the supported Model-Unit Type. This table only applies to setting this feature at unit creation time. Generally ignoreECC applies to redundant units.

Model	Raid0	Raid1	Raid5	Raid10	JBOD	Spare	Raid50	Single
7K/8K	N	N	N	N	N	N	N	N
9K	N	Y	Y	Y	N	N	Y	N

name=string attribute allows user to name the new unit. The maximum characters allowed for the string are 21. No space is allowed within the string. If user likes to use some special characters which the OS command shell reserves such as '<', '>', '!', and '&', etc in the name string, the user has to use quote ``'' around the name string in order to bypass the command shell. User can change the name of the unit any time after the unit creation. This is a feature for 9000 or above series of controllers. Please refer to commands */cx/ux set name=sting* for changing the name and */cx/ux show name* for checking.

storsave=protect|balance|perform attribute allows user to set the storsave policy of the new unit. It is a 9KSX/SE only feature. Please refer to command */cx/ux set storsave=protect|balance|perform* for detail.

***/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 visits 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. Default is to inform the OS.

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: Does not import non-JBOD on 7000/8000 models.

***/cx* commit**

This command instructs the controller to commit its dirty DCBs to persistent storage (ie disks). While 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 read from the disks, will 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 user would like to shutdown the host (Including UPS post failure automations). This command can then expedite the process by instructing the controller to finish pending requests, clear DCB's in-transaction flag as we are going down.

Note that block devices (cooked devices) do not require this and clients of block devices (such as file systems) will send its own shutdown request to the devices.

This command only applies to Windows operating system.

***/cx* flush**

This command allows you to flush the write cache on all units associated with the */cx* controller

***/cx* update *fw=filename_with_path* [*force*]**

This command allows the download of the specified firmware image to the corresponding controller. This command is for 9000 series controllers only.

fw=filename_with_path attribute allows user to specify the firmware image file name along with its path. The new image specified by this **filename_with_path** will be checked for compatibility with the current controller, current driver and current application versions. Subsequently a recommendation is given to the user followed by a prompt to continue. Once the user decides to proceed, the image will be downloaded to the controller. However, a reboot is required for the new image to take effect.

Example:

```
//localhost> /c2 update fw=/tmp/prom0006.img

Warning: We strongly recommend backing up your data before updating
the firmware. Updating the firmware can render the device driver
and/or management tools incompatible. It is recommended to have
a copy of current firmware image for rollbacks.

Examining compatibility data from firmware image and /c2 ... Done.

New-Firmware        Current-Firmware    Current-Driver    Current-API
----------------------------------------------------------------------
FE9X 3.05.00.005    FE9X 3.05.00.005    2.26.04.007       2.01.00.008

Current firmware version is the same as the new firmware.
Recommendation: No need to update.

Given the above recommendation...
Do you want to continue ? Y|N [N]: y
Downloading the firmware from file /tmp/prom0006.img ... Done.
The new image will take effect after reboot.
```

force attribute is optional. With it the compatibility checks are bypassed.

***/cx* show *alarms* [*reverse*]**

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 space programs such as CLI and/or 3DMPlus. These events reflect warning, debugging and/or informative messages for end user.

Alarms generated on 7000/8000 models do not have dates, as such you'll see a '-' (read not-applicable) in ``Date'' column. Also on 7000/8000 models, the alarm message, contain the severity as well, hence the ``Severity'' column is showing a '-' as well.

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

Typical output looks like:

```
//localhost> /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 this output to a file, where it can be communicated to tech support or further studied with Linux utilities like od(1).

Example:

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

/cx* show *rebuild

Model 9000 series support background tasks such as rebuild, verify, or self test activities. For each activity, up to 7 tasks can be registered, known as slots 1 through 7. Each task activity can be managed by a set of commands including **add**, **del**, **show** and **set**. Background tasks have a slot id, start day, hour, duration, and status attributes.

Rebuild activity attempts to (re)synchronize all members of redundant units such as RAID-1, RAID-10, RAID-5 and RAID-50. Rebuilds can be started manually or automatically if a spare has been defined. Scheduled rebuilds will take place during the scheduled window, if enabled.

This command displays the current rebuild background task schedule as illustrated below.

```
$ tw_cli /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       -       -               -               -
4       -       -               -               -
5       -       -               -               -
6       Mon     1:00am           4 hr(s)        disabled
7       Sun     12:00am          1 hr(s)        disabled
```

Status ``disabled'' indicates that the controller will not use the tabled schedules.

Note: The rebuild schedules are also applicable to initialization and migration processes.

For example:

If a unit is in the initialization state at noon on Wed, the tabled schedule above is show in the following:

```
$ tw_cli /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       -       -               -               -
4       -       -               -               -
5       -       -               -               -
6       Mon     1:00am           4 hr(s)        disabled
7       Sun     12:00am          1 hr(s)        disabled

$ tw_cli /c1 show

Unit  UnitType  Status         %RCmpl  %V/I/M  Stripe  Size(GB)  Cache  AVrfy
------------------------------------------------------------------------------
u0    RAID-5    INITIALIZING   0       -       64K     521.466   ON     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
```

Then if the user enables the tabled schedules, the unit initialization will be paused until next scheduled slot comes.

```
$ tw_cli /c1 set rebuild=enable
Enabling scheduled rebuilds on controller /c1 ...Done.

$ tw_cli /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

$ tw_cli /c1 show

Unit  UnitType  Status         %RCmpl  %V/I/M  Stripe  Size(GB)  Cache  AVrfy
------------------------------------------------------------------------------
u0    RAID-5    INIT-PAUSED    0       -       64K     521.466   ON     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***

Model 9000 series support background tasks such as rebuild, verify, or self test activities. For each activity, up to 7 tasks can be registered, known as slots 1 through 7. Each activity can be managed by a set of commands including **add**, **del**, **show** and **set** a task. Background tasks have a slot id, start day, hour, duration, and status attributes.

Verify activity attempts to verify 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 the 9000 series, non-redundant units such as RAID-0, JBOD, single, and spare, are also verified (by reading and reporting un-readable sectors).

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

```
$ tw_cli /c1 show verify

Verify Schedule for Controller /c1
========================================================
Slot    Day     Hour            Duration        Status
--------------------------------------------------------
1       Mon     2:00am           4 hr(s)        disabled
```

```
2       -       -               -               -
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
```

Status ``disabled'' indicates that the controller will not use the tabled schedules.

/cx* show *selftest

Model 9000 series support background tasks such as rebuild, verify, or self test activities. For each activity, up to 7 tasks can be registered, known as slots 1 through 7. Each activity can be managed by a set of commands including **add**, **del**, **show** and **set** a task. Background tasks have a slot id, start-day-time, duration, status attributes.

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 parallel ATA bus speed (between controller and 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 Mhz. Note that the UDMA selftest is not applicable (or required) with SATA drives, but is left enabled by default.

SMART selftest 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 failing drive should be replaced if this error occurs.

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

```
$ tw_cli /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. This command will fail if no (empty) slot is available.

For ``rebuild'' background task description, see command **/cx show rebuild**.

For example:

```
$ tw_cli /c1 add rebuild=Sun:16:3
```

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

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

This command adds a new background *verify* task to be executed on 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. This command will fail if no (empty) slot is available.

For ``verify'' background task description, see command **/cx show verify**.

For example:

```
$ tw_cli /c1 add verify=Sun:16:3
```

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

/cx* add *selftest=ddd:hh

This command adds a new background *selftest* task to be executed on 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. This command will fail if no (empty) slot is available.

For ``selftest'' background task description, see command **/cx show selftest**.

For example:

```
$ tw_cli /c1 add selftest=Sun:16
```

Will add a selftest background task schedule to be executed on Sundays at 4:00 PM.

/cx* del *rebuild=slot_id

This command will remove (or unregister) the rebuild background task in slot *slot_id*.

For ``rebuild'' background task description, see command **/cx show rebuild**.

For example:

```
$ tw_cli /c1 del rebuild=2
```

Will remove rebuild background task in slot 2.

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

/cx* del *verify=slot_id

This command will remove (or unregister) the verify background task in slot *slot_id*.

For ``verify'' background task description, see command **/cx show verify**.

For example:

```
 $ tw_cli /c1 del verify=3
```

Will remove rebuild 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.

/cx **del** ***selftest=slot_id***

This command will remove (or unregister) the selftest background task in slot *slot_id*.

For ``selftest'' background task description, see command **/cx show selftest**.

For example:

```
 $ tw_cli /c1 del selftest=3
```

Will remove selftest 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.

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

This command will *enable* or *disable* all rebuild background tasks on controller */cx*. When enabled, only tabled scheduled tasks will be followed (or used). Any previous on-demand background tasks will be ignored.

This command also allows you to set priority of rebuild vs 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 7/8000 series, the rebuild rate also applies to verify and mediascan tasks.

For ``rebuild'' background task description, see command **/cx show rebuild**.

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

This command will *enable* or *disable* all verify background tasks on controller */cx*. When enabled, only tabled scheduled tasks will be followed (or used). Any previous on-demand background tasks will be ignored.

This command allows you to set priority of verification vs 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 models.

For ``verify'' background task description, see command **/cx show verify**.

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

This command will *enable* or *disable* all or a particular **task=selftest_task** (UDMA or SMART) on a specified controller */cx*. When enabled, only specified **task=selftest_task** task will be performed during a scheduled timeslot. If no task is specified, the command is applicable to both tasks.

For ``selftest'' background task description, see command **/cx show selftest**.

For example:

```
 $ tw_cli /c0 selftest=enable task=UDMA
```

Will enable 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 JBOD export policy is off, CLI would not be able to create JBODs and during reboot, the firmware will not export JBOD units to the operating system. JBOD Export Policy is only supported on 9000 series controllers. Previous models did not have such a policy enforcement feature.

/cx **set** ***ondegrade=cacheoff|follow*** **(9000S only)**

This command allows you to set a controller based cache policy. If the policy is set to *cacheoff*, then if a unit is degraded, firmware will disable the write-cache on the degraded unit, regardless of what the unit-based policy is. If the policy is set to *follow*, then if a unit is degraded, firmware will follow whatever policy has been set for that unit.

/cx **set** ***spinup=nn***

This command allows you to set a controller based disk spin up policy. The value must be a positive integer between 1 to the number of disks/ports supported on the controller (e.g. 4, 8, 12, 16). This policy is used to stagger spin ups of disks at boot time in order to spread the power consumption on the power supply. For example, given a spin up policy of 2, the controller will spin up two disks at a time, pause, and then spin up another 2 disks, etc, etc. The amount of time to pause can be specified with the spin up stagger time policy.

/cx **set** ***stagger=nn***

This command allows you to set a controller based disk spin up stagger time policy. The value must be a positive integer between 0 to 60 seconds. This policy in conjunction with disk spin up policy specifies how the controller should spin up disks at boot time.

/cx **set** ***autocarve=on|off*** **(9000 series)**

This command allows you to set the Auto-Carving policy to be on or off. When the Auto-Carving policy is on, any unit larger than the carvesize is created or migrated into one or more carvesize volumes and a remaining volume. Each volume can be treated as an individual disk with its own file system. The default carvesize is 2 TB. This feature is useful for operating systems limited to 2 TB filesystems.

For example a 3 TB array would be configured into a 2 TB and a 1 TB volumes with default carvesize. For a 5 TB array, two 2 TB volumes would be created plus a 1 TB volume.

When autocarve policy is off, all the new unit creation or migration consists of one single volume.

Example:

```
 //localhost> /c0 set autocarve=on
 Setting Auto-Carving Policy on /c0 to on ... Done.
```

/cx **set** ***carvesize=[1024..2048]*** **(9000 series)**

This command allows you to set the carve size in GB. This feature works together with the autocarve above. See ``*/cx* **set** *autocarve=on|off*'' command above for details.

Example:

```
//localhost> /c0 set carvesize=2000
Setting Auto-Carving Size on /c0 to 2000 GB ... Done.
```

***/cx* set *autorebuild=on|off* (9KSX/SE only)**

This command sets Auto-Rebuild policy to be on or off. If the policy is on, the firmware will choose drives in the following priority order for a candidate of a rebuild operation (on a degraded unit).

1. Smallest usable capacity spare.

2. Smallest usable unconfigured drive.

3. Smallest usable capacity failed drive.

If the policy is off, spares are the only candidates for rebuild operations.

Example:

```
//localhost> /c0 set autorebuild=on
Setting Auto-Rebuild Policy on /c0 to on ... Done.
```

***/cx* set *autodetect=on|off disk*=<p:-p>|all (9000 series)**

This command is associated with the stagger spin-up feature during hot-plug. With stagger spin-up enabled (see command **/cx set spinup** and **/cx set stagger**), during reset or power on, the controller will try to detect all drives that are present and spin them up staggered in time, allowing the spread of power consumption on the power supply. Upon drive hot-plug, that is, **not** on power-on or reset, the default behavior of the system is automatic detection of the drives and immediate spin-up. This command would change the default behavior and set the controller to spin-up as the system at power-on.

autodetect=on|off attribute configures the controller drive auto-detect setting. It should be set to *off* to initiate the sequence for the stagger spin-up during hot-plug process. After the drives are inserted or re-inserted to the ports (as specified in the second attribute decribed below), it should be set back to *on* to complete the configuration process for the controller to initiate the drive spin-up.

disk=<p:-p>|all attribute specifies one or many disks (i.e., drives or ports). If a port is empty (i.e., no drive inserted), the echo message of the command refers to a port, and if there is already a drive inserted the message refers to a disk. The example below shows that auto detect has been set to *off* to initiate stagger spin-up during hot-plug, where *port 3* was empty and *ports 5* and *6* had drives inserted.

```
//localhost>> /c0 set autodetect=off disk=3:5-6
Setting Auto-Detect on /c0 to [off] for port [3] and for disk [5,6]... Done
```

If ``disk=all'', then all of the drives or ports for that controller are specified. for example:

```
//localhost>> /c0 set autodetect=off disk=all
Setting Auto-Detect on /c2 to [off] for all disks/ports... Done.
```

To illustrate how the command is used, here is a usage scenario:

```
1. Issue command (set autodetect=off) to disable automatic detection of the
   ports for staggered spin-up.
2. Pull out the drives of the specified ports (if not empty).
3. Replace the drives previously removed at the ports specified.
4. Issue command (set autodetect=on) to enable auto detect of the ports with
   the newly inserted drives.
```

The above procedure would spin-up the newly inserted drives in a staggered manner. Please note that the command takes longer to complete for ports that do not have drives inserted.

/cx* start *mediascan

This command applies to 7000/8000 controllers. It provides media scrubbing for validating functionality of a disk. This includes bad block detection and remapping, etc. The commands starts a media scan operation on the specified controller */cx*.

/cx* stop *mediascan

This command applies to 7000/8000 controllers. It provides media scrubbing for validating functionality of a disk. This includes bad block detection and remapping, etc. The commands stops a media scan operation on the specified controller */cx*.

Logical Disk Object Messages

Logical Disk Object Messages are commands (a.k.a. methods/messages) that are sent to an instance of a Logical Disk (a.k.a. unit) such as */c0/u0*.

***/cx/ux* show**

This command shows summary information on the specified unit */cx/ux*. If the unit consists of sub-units as with the case of RAID-10, and RAID-50 arrays, then each sub-unit is further presented. If the Auto-Carving policy was on at the time the unit was created and the unit is over the carve size (default is 2 TB - 1), multiple volumes will be created and will be displayed at the end of the summary information.

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> /c0/u0 show

Unit     UnitType  Status         %RCmpl  %V/I/M  Port  Stripe  Size(GB)
------------------------------------------------------------------------
u0       RAID-50   OK             -       -       -     64K     596.05
u0-0     RAID-5    OK             -       -       -     64K     -
u0-0-0   DISK      OK             -       -       p0    -       149.10
u0-0-1   DISK      OK             -       -       p2    -       149.10
u0-0-2   DISK      OK             -       -       p3    -       149.10
u0-1     RAID-5    OK             -       -       -     64K     -
u0-1-0   DISK      OK             -       -       p4    -       149.10
u0-1-1   DISK      OK             -       -       p5    -       149.10
u0-1-2   DISK      OK             -       -       p6    -       149.10

//localhost> /c0/u1 show
```

```
Unit     UnitType  Status         %RCmpl  %V/I/M  Port  Stripe  Size(GB)
------------------------------------------------------------------------
u1       RAID-0    OK             -       -       -     64K     3576.06
u1-0     DISK      OK             -       -       p0    -       298.01
u1-1     DISK      OK             -       -       p1    -       298.01
u1-2     DISK      OK             -       -       p2    -       298.01
u1-3     DISK      OK             -       -       p3    -       298.01
u1-4     DISK      OK             -       -       p4    -       298.01
u1-5     DISK      OK             -       -       p5    -       298.01
u1-6     DISK      OK             -       -       p6    -       298.01
u1-7     DISK      OK             -       -       p7    -       298.01
u1-8     DISK      OK             -       -       p8    -       298.01
u1-9     DISK      OK             -       -       p9    -       298.01
u1-10    DISK      OK             -       -       p10   -       298.01
u1-11    DISK      OK             -       -       p11   -       298.01
u1/v0    Volume    -              -       -       -     -       2047.00
u1/v1    Volume    -              -       -       -     -       1529.06
```

_/cx/ux_ show Attribute Attribute ...

This command shows the current setting of the given *attribute(s)*. One or many attributes can be requested. An invalid attribute will terminate the loop. Possible attributes are: initializestatus, name(9000 series), qpolicy(9KSX/SE Only), rebuildstatus, serial(9000 series), status, storsave(9KSX/SE Only), verifystatus, volumes(9000 series), autoverify, cache, ignoreECC, and identify. The attributes volumes, name, serial, autoverify, and ignoreECC are applicable to 9000 series controllers; and the attributes qpolicy, storsave, and identify are only applicable to 9KSX/SE controllers.

_/cx/ux_ show _status_

This command presents the status of the specified unit.

Example:

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

_/cx/ux_ show _rebuildstatus_

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

Example:

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

_/cx/ux_ show _verifystatus_

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

Example:

```
//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.

Example:

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

_/cx/ux_ show _volumes_ (9000 series)

This command presents the number of volumes of the specified unit.

Example:

```
//localhost> /c0/u5 show volumes
/c0/u5 Volume(s) = 2
```

_/cx/ux_ show _name_ (9000 series)

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

Example:

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

_/cx/ux_ show _serial_ (9000 series)

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

Example:

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

_/cx/ux_ show _qpolicy_ (9KSX/SE only)

This command presents the queue policy of the firmware. If the queue policy is on, the firmware utilizes the drive queueing policy. Some drives do not support any queueing policy, this policy will have no effect on those drives.

For a spare unit, drive queuing is not meaningful or applicable. For example, when a spare becomes a true unit in migration, it would adopt the queue policy of the ``new'' unit. Thus, this commmand does not show the queue policy for the spare unit type.

Example:

```
//localhost> /c0/u5 show qpolicy
/c0/u5 Command Queuing Policy = on
```

_/cx/ux_ show _storsave_ (9KSX/SE only)

This command presents the storsave policy (protect|balance|perform) of the firmware on the unit.

For detail, see */cx/ux* **set** *storsave=protect|balance|perform* (9KSX/SE only).

Example:

```
 //localhost> /c0/u5 show storsave
 /c0/u5 Command Storsave Policy = protect
```

***/cx/ux* show *identify* (9KSX/SE only)**

This command is related to the unit set identify command and it shows the identify status of the specified unit.

For example: Example:

```
 //localhost> /c0/u0 show identify
 /c0/u0 Identify status = on.
```

***/cx/ux* show *autoverify* (9000 series)**

This command presents the current autoverify setting of the specified unit.

Example:

```
 //localhost> /c0/u0 show autoverify
 /c0/u0 Auto Verify Policy = off
```

/cx/ux* show *cache

This command presents the current write cache state of the specified unit.

Example:

```
 //localhost> /c0/u0 show cache
 /c0/u0 Cache State = on
```

***/cx/ux* show *ignoreECC* (9000 series)**

This command presents the current setting of the ignoreECC policy for the specified unit.

Example:

```
 //localhost> /c0/u0 show ignoreECC
 /c0/u0 Ignore ECC policy = off
```

***/cx/ux* show all**

This command shows the current setting of all above attributes.

If the Auto-Carving policy was on at the time the unit was created and the unit is over the carve size (default is 2 TB - 1), multiple volumes will be created and will be displayed at the end of the summary information.

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

 //localhost> /c0/u1 show

 Unit     UnitType  Status         %RCmpl  %V/I/M  Port  Stripe  Size(GB)
 ------------------------------------------------------------------------
 u1       RAID-0    OK             -       -       -     64K     3576.06
 u1-0     DISK      OK             -       -       p0    -       298.01
 u1-1     DISK      OK             -       -       p1    -       298.01
 u1-2     DISK      OK             -       -       p2    -       298.01
 u1-3     DISK      OK             -       -       p3    -       298.01
 u1-4     DISK      OK             -       -       p4    -       298.01
 u1-5     DISK      OK             -       -       p5    -       298.01
 u1-6     DISK      OK             -       -       p6    -       298.01
 u1-7     DISK      OK             -       -       p7    -       298.01
 u1-8     DISK      OK             -       -       p8    -       298.01
 u1-9     DISK      OK             -       -       p9    -       298.01
 u1-10    DISK      OK             -       -       p10   -       298.01
 u1-11    DISK      OK             -       -       p11   -       298.01
 u1/v0    Volume    -              -       -       -     -       2047.00
 u1/v1    Volume    -              -       -       -     -       1529.06
```

***/cx/ux* remove [*noscan*] [*quiet*]**

This command allows you to remove (or export) a **unit**. Exporting a unit will instruct the firmware to remove the specified unit from its pool of managed units, but retains the DCB (Disk Configuration Block) meta-data. As such the unit can later be imported back. *noscan* is used to not inform the OS of this change. Default is to inform the OS. The quiet option is for non-interactive mode.

Use caution when using this command. Units that are currently in use or mounted cannot be removed.

***/cx/ux* del [*noscan*] [*quiet*]**

This command allows you to delete a **unit**. Deleting a unit not only remove the specified unit from the controller's list of managed units, but also **destroys** the DCB (Disk Configuration Block) meta-data. Ports (or disks) associated with this unit will now be part of the free pool of managed disks. In another words, once the unit is deleted, all the data on the unit **can not be recovered**. *noscan* is used to not inform the OS of this change. Default is to inform the OS. The quiet option is for non-interactive mode.

Use caution when using this command. **This is a destructive command and should be used with extreme care**. Units that are currently in use or mounted should not be deleted.

***/cx/ux* start *rebuild disk=p* [*ignoreECC*]**

This command allows you to rebuild a DEGRADED unit by 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 for the operation to continue via

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

Note that the disk to be used to rebuild a unit, must be a SPARE or unconfigured disk.

/cx/ux **start** ***verify***

This command starts a background verification process on the specified unit */cx/ux*. The following shows the supported matrix 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 that controller model.

```
Model | Raid0 | Raid1 | Raid5 | Raid10 | Raid50 | Single | JBOD | Spare |
------+-------+-------+-------+--------+--------+--------+------+-------+
7K/8K |  No   |  Yes  |  Yes  |  Yes   |  N/A   | N/A    | No   | No    |
------+-------+-------+-------+--------+--------+--------+------+-------+
9K    |  Yes  |  Yes  |  Yes  |  Yes   |  Yes   | Yes    | Yes  | Yes   |
------+-------+-------+-------+--------+--------+--------+------+-------+
```

Note that if subsequent to this command, one enables the background verify task to follow the scheduled slots, then this on-demand task will be paused until the next scheduled timeslot.

/cx/ux **pause** ***rebuild***

This command allows you to **pause** the rebuild operation on the specified REBUILDING unit **/cx/ux**. 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** function is provided to enable 7000/8000 users to achieve functionality with use of OS provided schedulers such as `cron(8)` or, `at(1)` in Linux 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 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 **resume** function is provided to enable 7000/8000 users to achieve similar functionality with use of OS provided schedulers such as `cron(8)` or, `at(1)` in Linux or user supplied programs.

/cx/ux **stop** ***verify***

This command stops a background verification process on the specified unit */cx/ux*. The following shows the supported matrix 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 that controller model.

```
Model | Raid0 | Raid1 | Raid5 | Raid10 | Raid50 | Single | JBOD | Spare |
------+-------+-------+-------+--------+--------+--------+------+-------+
7K/8K |  No   |  Yes  |  Yes  |  Yes   |  N/A   | N/A    | No   | No    |
------+-------+-------+-------+--------+--------+--------+------+-------+
9K    |  Yes  |  Yes  |  Yes  |  Yes   |  Yes   | Yes    | Yes  | Yes   |
------+-------+-------+-------+--------+--------+--------+------+-------+
```

Note that if subsequent to this command, one enables the background verify task to follow the scheduled slots, then this on-demand task will be paused until the next scheduled timeslot.

/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 command allows you to turn on/off the autoverify operation on a specified unit */cx/ux*. Once the autoverify=on, the RAID firmware will pick a time to start the verify process on the unit. If the allocated schedule windows is enabled, the verify process becomes active during the scheduled windows. Otherwise, the firmware will decide when the verify needs to be paused or restarted again before it completes.

You can use the **show verify** command to display the existing schedule windows. The autoverify operation is a continuous verify operation, which takes place within the existing schedule windows (displayed with /cx show verify) if the schedule is enabled. While the ``/cx show verify'' command allows you to see the time for the verify operation, this command allows you to enable or disable the autoverify operation on the specified unit. This feature only applies to 9000 models.

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

This command allows you to turn the write cache on a specified unit */cx/ux*, on or off. This feature is supported on both 7000/8000 and 9000 models. The quiet option is for non-interactive mode. It can be used in conjunction with *set cache=on* command when the controller has no BBU installed. The following is the Raid Type-Model support matrix.

```
Model | Raid0 | Raid1 | Raid5 | Raid10 | Raid50 | Single | JBOD | Spare |
------+-------+-------+-------+--------+--------+--------+------+-------+
7K/8K |  Yes  |  Yes  |  Yes  |  Yes   |  N/A   | N/A    | Yes  | No    |
------+-------+-------+-------+--------+--------+--------+------+-------+
9K    |  Yes  |  Yes  |  Yes  |  Yes   |  Yes   | Yes    | Yes  | No    |
------+-------+-------+-------+--------+--------+--------+------+-------+
```

/cx/ux **set** ***identify=on|off*** **(9KSX/SE only)**

This command allows you to identify a unit within an enclosure. The LEDs of the drive slots that are associated with the specified unit would be identified. If the EPCT (Enclosure Port Configuration Table) equates identify with blinking (default), the LEDs associated with the drive slots of the specified unit would blink.

For example:

```
//localhost> /c0/u0 set identify=on
Sending Identify request for unit /c0/u0 to [on] ... Done.
```

/cx/ux **set** ***ignoreECC=on|off***

This command allows you to set the ignoreECC policy for a given unit such that during rebuild of the specified unit, which could begin automatically (if the unit is degraded and spare has been defined) or manually, to be applied to the rebuild operation. Setting overwriteECC to on means ignoreECC. This feature only applies to 9000 models.

/cx/ux **set** ***name=string*** **(9000 series)**

This command allows you to name the unit to an arbitrary name upto 21 characters. No space is allowed within the string. If user likes to use some special characters which the OS command shell reserves such as '<', '>', '!', and '&', etc in the name string, the user has to use quote ``'' around the name string in order to bypass the command shell. Users can use this name in conjunction with the unit serial number (which created at the unit creation time) to cross reference with the unit. It is user's responsibility to give unique or redundant names on all units. This feature only applies to 9000 models.

Note: Users can also name a JBOD. But the naming information will not persist across power cycles or resets due to the nature of JBODs.

_/cx/ux_ set _qpolicy=on|off_ (9KSX/SE only)

This command presents the queue policy of the firmware. If the queue policy is on, the firmware utilizes the drive queueing policy. Some drives do not support any queueing policy, this policy will have no effect on those drives.

For a spare unit, drive queuing is not meaningful or applicable. For example, when a spare undergo unit migration and becomes a true unit, it adopts the queue policy of the ``new'' unit. Thus, this commmand does not set the queue policy for the unit type spare.

Example:

```
//localhost> /c0/u5 set qpolicy = on

Setting Command Queuing Policy for unit /c0/u5 to [on] ... Done.
```

_/cx/ux_ set _storsave=protect|balance|perform_ [_quiet_] (9KSX/SE only)

This command sets the storsave policy of the firmware to be either protect, balance, or perform when the unit write cache is enabled.

This feature is available only within 9KSX/SE controller family or above. There is a tradeoff among the available settings. The following description about the settings should help you to decide which one is suitable to you and your application. The protect mode is the default setting.

protect -- provide the maximum data protection among the controller settings. When user sets storsave to protect mode, it means :

1. ``Write Cache'' will be disabled when the unit becomes ``DEGRADED'',

2. all data flushing from controller cache will be flushed to media, and

3. incoming FUA (Force Unit Access) host request will be ignored if a BBU is installed and enabled; Otherwise, will be honored.

perform -- provide the maximum performance and less data protection among the the controller settings. When user sets the storsave to perform mode, it means:

1. ``Write Cache'' will not be disabled when the unit becomes ``DEGRADED'',

2. all data flushing from controller cache will be flushed to disk, and

3. incoming FUA (Force Unit Access) host request will be honored.

When storsave is set to **perform**, a warning about the data loss in the event of power failure is giving to user to confirm the option. If user want to skip the confirmation, [*quiet*] option can be used to by pass the warning.

balance -- provide more data protection than perform mode but less data protection than protect mode. And provide better performance than protect mode but less performance than perform mode. When user sets the storsave to the balance mode, it means:

1. ``Write Cache'' will not be disabled when the unit becomes ``DEGRADED'',

2. all data flushing from controller cache will be flushed to media if a BBU is installed and enabled; Otherwise, will be flushed to disk only, and

3. incoming FUA (Force Unit Access) host request will be ignored if a BBU is installed and enabled; Otherwise, will be honored.

Example:

```
//localhost> /c0/u5 set storsave=protect

Setting Command Storsave Policy for unit /c0/u5 to [protect] ... Done.
```

_/cx/ux_ migrate type=_RaidType_ [disk=_p:-p_] [group=_3|4|5|6|7|8_] [stripe=_Stripe_] [_noscan_] [_nocache_] [_autoverify_]

This command allows you to migrate an existing unit (aka source) to a unit with *type=RaidType* (aka destination) to achieve a goal of either capacity expansion. The destination is subject to similar rules and policies during its creation time. For example a valid number of disks and parameters must be specified. The destination unit must use all source disks and potentially augment in the disk=*p:-p* list for a larger capacity. Un-specified 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 drives participating in RAID1 into multiple units of type single. The source unit name information will be duplicated to all destination units. But the source unit serial number will **not** be carried over to new unit. The new destination unit will have its own serial number.

This feature is only available with 9000 series of controllers.

Note: We do not recommend users to move a migrating unit to another controller before its migrate operation is completed. Any unclean shutdown on the migrating unit may lead the unit to a RECOVERY state in another controller which the migrate operation can not be continue until the unit is moved back to its original controller.

type=RaidType consists of the destination unit RAID type as in **raid0**, **raid1**, **raid5**, **raid10**, **raid50**, **raid6**, or **single**.

For example ``type=raid5'' indicates the destination unit is RAID-5.

The following table illustrates valid migration paths:

Src/Dst	Raid0	Raid1	Raid5	Raid10	Raid50	Single	JBOD	Spare	Raid6
Raid0	Y	N	Y	Y	Y	N	N	N	Y
Raid1	Y	N	Y	Y	Y	Y	N	N	Y
Raid5	Y	N	Y	Y	Y	N	N	N	Y
Raid10	Y	N	Y	Y	Y	N	N	N	Y
Raid50	Y	N	Y	Y	Y	N	N	N	Y
Single	Y	Y	Y	Y	Y	N	N	N	Y
JBOD	N	N	N	N	N	N	N	N	N

```
 Spare  |   N   |   N   |   N   |   N    |    N   |   N    |  N  |   N   |   N   |
--------+-------+-------+-------+--------+--------+--------+------+-------+-------+
 Raid6  |   Y   |   N   |   Y   |   Y    |    Y   |   N    |  N  |   N   |   Y   |
--------+-------+-------+-------+--------+--------+--------+------+-------+-------+
```

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 ``**:**'' or ``**-**'' as port index separators. A dash indicates a range and can be mixed with ``:''. For example **disk=0:1:2-5:9:12** indicates port 0, 1, 2 thru 5 (inclusive), 9 and 12.

group=3|4|5|6|7|8 is **only** applicable to **type=raid50** which consists of a number of disks per group. Recall that 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. 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. However given 12 disks, specifying 3 will create four RAID-5 under the RAID-0 level. Given 6 disks and grouping of 6 is not allowed, as you'll basically be creating a RAID-5.

The default disk **group** varies based on number of disks. For 6 & 9 disks, default is group=3. For 8 disks, default is group=4. For 10 or 15 disks, default is group=5. For 12 or 16 disks, default is group=4. For 14 disks, default is group=7. Case of 12 disks could be grouped with group=3, group=4, or group=6. Group=4 was set by default as it provides best net capacity and performance. Case of 15 disks could be grouped with group=3 or group=5. And case of 16 disks could be grouped with group=4 and group=8.

Note that RAID-10 has **group=2** always.

Stripe consists of the logical unit 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 (kilo bytes).

```
Model | Raid0   | Raid1  | Raid5  | Raid10  | JBOD | Spare | Raid50 | Single |
------+---------+--------+--------+---------+------+-------+--------+--------+
9K    |  16     |  N/A   |  16    |  16     | N/A  |  N/A  |  16    |  N/A   |
      |  64     |        |  64    |  64     |      |       |  64    |        |
      |  256    |        |  256   |  256    |      |       |  256   |        |
------+---------+--------+--------+---------+------+-------+--------+--------+
```

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

nocache switch instructs CLI disable the write cache on the migrated unit. Enabling write cache increases performance at the cost of high-availability.

autoverify switch enables the autoverify attribute on the unit that is to be migrated. For more details on this feature, refer to cx/ux set Commands section of this document.

For example:

In the case of split mirror:

```
$ tw_cli /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 RAID type of Single.

In the case of capacity expansion:

```
$ tw_cli /c0/u3 migrate type=raid10 disk=10-11 stripe=16
```

Indicates that the destination unit has a RAID type of raid10 and has the disks 10 and 11 in addition to all the disks in the existing unit u3.

The following shows how migrating units would be displayed...

```
3ware CLI> /c0 show

Unit  UnitType  Status         %RCmpl  %V/I/M  Stripe  Size(GB)  Cache  AVrfy
------------------------------------------------------------------------------
u0    RAID-5    OK             -       -       64K     596.004   ON     OFF
u2    SPARE     OK             -       -       -       149.042   -      OFF
u3    Migrator  MIGRATING      -       39      -       149.001   ON     OFF

Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
p0     OK               u0     149.05 GB   312581808     WD-WCANM1771318
p1     OK               u0     149.05 GB   312581808     WD-WCANM1757592
p2     OK               u0     149.05 GB   312581808     WD-WCANM1782201
p3     OK               u0     149.05 GB   312581808     WD-WCANM1753998
p4     OK               u2     149.05 GB   312581808     WD-WCANM1766952
p5     OK               u3     149.05 GB   312581808     WD-WCANM1882472
p6     OK               u0     149.05 GB   312581808     WD-WCANM1883862
p7     OK               u3     149.05 GB   312581808     WD-WCANM1778008
p8     OK               -      149.05 GB   312581808     WD-WCANM1770998
p9     NOT-PRESENT      -      -           -             -
p10    OK               u3     149.05 GB   312581808     WD-WCANM1869003
p11    OK               u3     149.05 GB   312581808     WD-WCANM1762464

3ware CLI> /c0/u0 show

Unit     UnitType  Status         %RCmpl  %V/I/M  Port  Stripe  Size(GB)
------------------------------------------------------------------------
u3       Migrator  MIGRATING      -       39      -     -       -

su3      RAID-1    OK             -       -       -     -       149.001
su3-0    DISK      OK             -       -       p5    -       149.001
su3-1    DISK      OK             -       -       p7    -       149.001
su3/v0   Volume    -              -       -       -     -       149.001

du3      RAID-10   OK             -       -       -     16K     298.002
du3-0    RAID-1    OK             -       -       -     -       -
du3-0-0  DISK      OK             -       -       p5    -       149.001
du3-0-1  DISK      OK             -       -       p7    -       149.001
du3-1    RAID-1    OK             -       -       -     -       -
```

```
du3-1-0  DISK      OK             -       -       p10   -       149.001
du3-1-1  DISK      OK             -       -       p11   -       149.001
du3/v0   Volume    -              -       -       -     -       149.001
```

The above report indicates that */c0/u3* is a migrating unit with 39% completion. The report also indicate that Source Unit *su0* is of type RAID-1 and Destination Unit *du0* is of type RAID-10.

Port Object Messages

Port Object Messages are commands (a.k.a. methods/messages) that are sent to an instance of a disk which attaches to a port such as */c0/p0*.

***/cx/px* show**

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

Example:

```
//localhost> /c0/p5 show
Port   Status           Unit   Size        Blocks        Serial
---------------------------------------------------------------
p5     OK               u5     149.05 GB   312581808     WD-WMACK1406498
```

The above report indicate that port 5 of controller 0 is attached to one Western Digital disk with status OK participating in unit 5.

/cx/px* show *Attribute Attribute ...

This command shows the current setting of the given *attribute(s)* on the specified port or disk. One or many attributes can be requested. Invalid attribute will terminate the loop. Possible attributes are: capacity, firmware, identify(9KSX/SE only), lspeed(9KSX/SE only), model, ncq(9KSX/SE only), serial, smart, and status.

/cx/px* show *status

This command presents the status of the specified port.

Example:

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

/cx/px* show *model

This command presents the model of the specified port.

Example:

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

/cx/px* show *serial

This command presents the serial number of the specified port.

Example:

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

/cx/px* show *firmware

This command presents the firmware version of the specified port.

Example:

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

***/cx/px* show *identify* (9KSX/SE only)**

This command presents the LED status of the port for 9KSX/SE only.

Example:

```
//localhost> /c0/p5 show identify

/c0/p5 Identify Status = on
```

***/cx/px* show *ncq* (9KSX/SE only)**

This command presents the NCQ (Native Command Queueing) information of the port.

Example:

```
//localhost> /c0/p5 show ncq

/c0/p5 NCQ Supported = No
/c0/p5 NCQ Enabled = No
```

***/cx/px* show *lspeed* (9KSX/SE only)**

This command presents the SATA link speed supported by the disk behind the port and the actual speed is set to.

Example:

```
//localhost> /c0/p5 show lspeed

/c0/p5 SATA Link Speed Supported = 3.0 Gb/s
/c0/p5 SATA Link Speed = 3.0 Gb/s
```

/cx/px* show *capacity

This command presents the capacity, both in human readable (such as GB) and block count of the specified port. Note that at this writing, human readable format is computed based on division by 1000 (not 1024) as is popular with hard disk vendors.

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. Note that this data is actually extracted live from the disk; as such this command could be used to get most recent data about presence or lack of a disk. Be aware that extracting SMART data will burden the I/O bandwidth.

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 that at this writing, we are not decoding the SMART data. Also note that if the disk attached to the specified port is not present or there are cabling problems reaching the disk, CLI will return an error. This could be one way of detecting whether a disk is present or not.

/cx/px* show *all

This command shows the current setting of all above attributes.

***/cx/px* remove [*noscan*] [*quiet*]**

This command allows you to remove (or export) a port **/cx/px** (or drive). Exporting a port will instruct the firmware to remove the specified port form its pool of managed ports, but retains the DCB (Disk Configuration Block) meta-data on the attached disk. You can import (or re-introduce) the port via **rescan**. *noscan* is used to not inform the OS of this change. Default is to inform the OS. The quiet option is for non-interactive mode.

Use caution when using this command. Drives, which are part of a redundant array, can be removed, but the array will be degraded. Non-redundant drives, which are part of a unit, can not be removed.

***/cx/px* set *identify=on|off* (9550SX and 9590SE only)**

This command sets the LED status of the port. If the identify is set to on, the firmware activates the setting of the corresponding LED of the port. If the setting of the configuration for LED is blinking for identify=on, this will blink the LED of the port.

Note: Enclosure services hardware is also required.

Example:

```
//localhost> /c0/p5 set identify=on

Setting Port Identify on /c0/p5 to [on] ... Done.
```

BBU Object Messages

BBU (Battery Backup Unit) Object Messages are commands (a.k.a. methods/messages) that are sent to an instance of a BBU such as */c0/bbu*. This object is only available on 9000S controllers where the BBU is actually installed.

***/cx/bbu* show**

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

For example:

```
//localhost> /cx/bbu show

Name  OnlineState  BBUReady  Status    Volt     Temp     Hours  LastCapTest
---------------------------------------------------------------------------
bbu   On           No        Testing   OK       OK       72     01-Jul-2004
```

Indicates the date the battery capacity was last measured is 01-Jul-2004. The battery is estimated to last for 72 hours from the last tested date. The BBU unit is currently testing the battery. Both voltage and temperature are normal. And the BBU is **not** ready to backup the write cache on the controller due to the testing.

/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 requested. Invalid attribute will terminate the loop. Possible attributes are: batinst, bootloader, cap, fw, lasttest, pcb, ready, serial, status, temp, and volt.

/cx/bbu* show *status

This command **show**s the status of the BBU. Possible values are:

Testing

Battery test is currently in progress. It 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 applicable RAID units are also disabled. If the test is completed with no error and the BBU returns back to WeakBat or OK state, the write cache will be resumed. If a Fault, Failed or an Error occurs during the test, the write cache remains at the disabled state until the problem is fixed.

Charging

BBU is currently charging the battery. The charging is started automatically by the BBU whenever necessary. During the charging, the BBU is **not** capable of backup operation and the write cache is disabled. Once charging is completed and the BBU returns back to OK status, the write cache will be resumed. If a FAULT or an ERROR occurs during the test, the write cache remains at the disabled state until the problem is fixed.

Fault

A battery fault is detected. At this state, the BBU is **not** capable of backup operation and the write cache is disabled. We recommend you to replace the battery and/or the BBU board to fix the problem as soon as possible so that the write cache will be enabled again.

Error

Other BBU error is detected. At this state, the BBU is **not** capable of backup operation and the write cache is disabled. We recommend you to replace the battery and/or the BBU board to fix the problem as soon as possible so that the write cache will be enabled again.

Failed

The battery failed a test. At this state, the BBU is **not** capable of backup operation and the write cache is disabled. We recommend you to replace the battery and/or the BBU board to fix the problem as soon as possible so that the write cache will be enabled again.

WeakBat

BBU is functioning normally which means it is online and capable of backing up the write cache. But the battery is **weak** and should be **replaced**.

OK

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

-

Battery is not present or BBU unit is not installed.

/cx/bbu* show *batinst

This command **show**s the date when the current battery was installed.

/cx/bbu* show *lasttest

This command **show**s the date the battery capacity was last measured.

/cx/bbu* show *volt

This command **show**s 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 need to be concerned.

/cx/bbu* show *temp

This command **show**s 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 need to be concerned.

/cx/bbu* show *cap

This command **show**s the battery capacity in hours.

/cx/bbu* show *serial

This command **show**s the BBU serial number.

/cx/bbu* show *fw

This command **show**s the BBU board firmware version number.

/cx/bbu* show *pcb

This command **show**s the PCB revision number on the BBU unit.

/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 above attributes on the BBU board.

For example:

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

   /c1/bbu Firmware Version          = BBU: 1.04.00.007
   /c1/bbu Serial Number             = Engineering Sample.
   /c1/bbu Online State              = On
   /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 Intallation 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 is disabled. The performance of all the units under the controller may be impacted because their write IOs are not cached. Once the test is completed with no error and the BBU returns back to OK state, the write cache will be resumed. The quiet option is for non-interactive mode.

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

Note: The test cannot be terminated before it completes.

***/cx/bbu* enable**

This command enables the BBU detection on the controller. And the controller will utilize BBU functionality in the event of power failure if BBU is there and ready.

***/cx/bbu* disable [quiet]**

This command disables the BBU detection on the controller. In this situation, the controller ignores the existence of the BBU when the BBU detection is disabled. In another words, when a BBU attaches to a controller and the BBU detection is disabled, the storage management software will show user that there is no BBU installed on this controller. This is because the BBU detection is disabled. The quiet option is for non-interactive mode.

Enclosure Services Commands

The Enclosure Services Commands are a set of methods and messages associated with the enclosure object that provide information about the enclosure and its elements as well as identification of drive slots in an enclosure system.

These commands supports the 9550SX, 9590SE and 9650SE controllers. Moreover, the EPCT (Enclosure Port Configuration Table) of the enclosure is assumed to have been downloaded to the controller.

Enclosure Object Messages are commands (a.k.a. methods/messages) that are sent to an instance of an enclosure such as /e0. The enclosure element object messages are commands sent to an instance of the enclosure element such as /e0/slot0. The following lists the commands that are enclosure-general and element-specific. The latter includes commands for the slot, fan, and temperature sensor elements.

***/ex* show**

This command shows summary information on the specified enclosure */ex*. This report consists of four parts; the **Enclosure** summary listing the present elements, a **Fan** summary section listing of all present fans, a **Temperature Sensor** summary section listing of all present temperature sensors and a **Slot** summary section listing of slots and associated information for the specified enclosure.

Typical output looks like:

```
 //localhost> /e0 show

 Encl        Controllers
 ------------------------
 e0          /c0

 Fan       Status
 ---------------------
 fan0      OK

 TempSensor       Temperature
 ------------------------------
 temp0            24~C(75~F)

 Slot      Status          Port          Identify
 ---------------------------------------------------
 slot0     OK              /c0/p0        No
 slot1     OK              /c0/p1        Yes
 slot2     NO-DEVICE       -             No
 slot3     NO-DEVICE       -             No
```

/ex* show *Attribute Attribute ...

This command shows the current setting of the given *attribute(s)*. One or many attributes can be requested. An invalid attribute will terminate the loop. Possible attributes are: controllers, slots, fans, and temp.

/ex* show *controllers

This command reports the controller associated with enclosure */ex*.

Example:

```
 //localhost> /e0 show controllers
 /e0 Connected to /c0 controller.
```

/ex* show *slots

This command reports the slots and their associated information in enclosure */ex*.

Example:

```
 //localhost> /e0 show slots

 Slot      Status          Port          Identify
 ---------------------------------------------------
 slot0     OK              /c0/p0        No
 slot1     OK              /c0/p1        Yes
 slot2     NO-DEVICE       -             No
 slot3     NO-DEVICE       -             No
```

/ex* show *fans

This command reports the fans and their associated information in enclosure */ex*.

```
Example:

 //localhost> /e0 show fans

 Fan       Status
 ---------------------
 fan0      OK
```

/ex* show *temp

This command reports the temperature sensors and their associated information in enclosure */ex*.

Example:

```
 //localhost> /e0 show temp

 TempSensor       Temperature
 ------------------------------
 temp0            24~C(75~F)
```

/ex* show *all

This command shows the current setting of all attributes.

Enclosure Element Slot

The slot commands provide information about the slot elements in the enclosure unit.

***/ex/slotx* show**

This command shows slot information on the specified enclosure */ex*. The slot name is followed by its status. If a slot has been inserted with a drive and no fault has been detected, the status is OK. If the slot is empty the status would indicate NO-DEVICE. The port that is correlated to the slot is indicated in the next column. If no device is found in that slot that column would be indicated with a dash (-). The next column shows whether the specified slot has been identified. For example:

```
//localhost> /e0/slot1 show

Slot      Status          Port          Identify
----------------------------------------------------
slot1     OK              /c0/p1        Yes
```

ex/slotx **show** ***identify***

This command shows the identify status of the specified slot. For example:

```
//localhost> /e0/slot1 show identify
/e0/slot1 Identify status = on.
```

/ex/slotx **set** ***identify=on|off***

This command identifies the specified slot by setting the identify attribute to either on or off. Setting it on would blink the LED of the specified drive slot, provided that the EPCT has been set up with the corresponding default of blinking for identify. For example:

```
//localhost> /e0/slot1 set identify=on
Sending Identify request to Drive Slot /e0/slot0 to [on] ... Done.
```

Enclosure Element Fan

The fan commands provide information about the fan elements in the enclosure unit.

/ex/fanx **show**

This command shows information about the fan elements in the specified enclosure */ex*. The fan name is followed by its status. If a fan is on and no fault has been detected, the status would indicate OK. For example:

```
//localhost> /e0/fan0 show

Fan       Status
------------------
fan0      OK
```

Enclosure Element Temperature Sensor

The temperature sensor commands provide information about the temperature sensor elements in the enclosure unit.

/ex/tempx **show**

This command shows information about the temperature sensor elements in the specified enclosure /ex. The temperature sensor name is followed by the temperature sensed in the enclosure unit. For example:

```
//localhost> /e0/temp0 show

TempSensor       Temperature
------------------------------
temp0            24~C(75~F)
```

Help Commands

This command set provides brief on-line help. At top level of the command set, it is considered to be in the Shell Object. The Shell object has focus, show, flush, rescan, and commit commands. In addition to the Shell Object, we also have other objects such as /cx, /cx/ux, /cx/px, /cx/bbu, /ex, /ex/slotx, /ex/fanx, and /ex/tempx. Using the help command on objects, the help command shows 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.

```
 //localhost> help /cx

   /cx show
   /cx show attribute [attribute ...]           where attributes are:
            achip|allunitstatus|autocarve(9000 series)|bios|driver|firmware
            autorebuild(9KSX/SE only)|carvesize(9000 series)
            drivestatus|exportjbod|ctlbus(9KSX/SE only)
            memory|model|monitor|numdrives|numports|numunits|unitstatus
            ondegrade(9000S only)|pcb|pchip|serial|spinup|stagger
   /cx show all               where all means attributes and configurations.
   /cx show diag
   /cx show alarms [reverse]
   /cx show rebuild                   (9000 series)
   /cx show verify                    (9000 series)
   /cx show selftest                  (9000 series)

   /cx add type=<RaidType> disk=<p:-p..> [stripe=<Stripe>] [noscan] [nocache] [group=<3|4|5|6|7|8>] [autoverify] [ignoreECC] [name=string
           RaidType = { raid0, raid1, raid5, raid10, raid50, single, spare, JBOD(7000/8000 only),
                                                              raid6 (9650SE only) }
   /cx add rebuild=ddd:hh:duration     (9000 series)
   /cx add verify=ddd:hh:duration      (9000 series)
   /cx add selftest=ddd:hh             (9000 series)

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

   /cx set exportjbod=on|off                              (9000 series)
   /cx set ondegrade=cacheoff|follow                      (9000S only)
   /cx set spinup=nn                                      (9000 series)
   /cx set stagger=nn                                     (9000 series)
   /cx set autocarve=on|off                               (9000 series)
   /cx set carvesize=[1024..2048]                         (9000 series)
   /cx set autorebuild=on|off                             (9KSX/SE only)
   /cx set rebuild=enable|disable|<1..5>  (enable|disable for 9000 series)
   /cx set verify=enable|disable|<1..5>   (enable|disable for 9000 series)
   /cx set selftest=enable|disable [task=UDMA|SMART]      (9000 series)

   /cx update fw=filename_with_path [force]               (9000 series)
   /cx flush
   /cx commit (** Windows only **) (previously 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>
```

However, user can also get help with ``?'' or ``help'' as the user progresses with the specific object or a command after an object.

For example: User progresses to /c0 show and needs help on what specific attribute syntax, user can use '?' to get help as following:

```
//localhost> /c0 show ?

  /cx show
  /cx show attribute [attribute ...]           where attributes are:
           achip|allunitstatus|autocarve(9000 series)|bios|driver|firmware
           autorebuild(9KSX/SE only)|carvesize(9000 series)
           drivestatus|exportjbod|ctlbus(9KSX/SE only)
           memory|model|monitor|numdrives|numports|numunits|unitstatus
           ondegrade(9000S only)|pcb|pchip|serial|spinup|stagger
  /cx show all               where all means attributes and configurations.
  /cx show diag
  /cx show alarms [reverse]
  /cx show rebuild                    (9000 series)
  /cx show verify                     (9000 series)
  /cx show selftest                   (9000 series)

//localhost>
```

help

This command provide a table of contents, providing an overall navigational help. Typical output looks like:

```
//localhost> help

Copyright(c) 2004-2006 Applied Micro Circuits Corporation(AMCC). All rights reserved.

AMCC/3ware CLI (version 2.00.04.007)

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 controllers only).
quit      Exits the CLI.
           ---- Primary Command Syntax ----
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).
update    Update controller firmware from an image file.
commit    Commit dirty DCB to storage on controller(s).
/cx       Controller specific commands.
/cx/ux    Unit specific commands.
/cx/px    Port specific commands.
/cx/bbu   BBU specific commands.               (9000 controllers only)
/ex       Enclosure specific commands.                  (9KSX/SE only)
/ex/slotx Slot specific commands.
/ex/fanx  Fan  specific commands.
/ex/tempx Temperature sensor specific commands.

Certain commands are qualified with constraints of controller type/model support.
Please consult the tw_cli documentation for explanation of the controller-qualifiers.

The controller-qualifiers of the Enclosure commands (/ex) also apply to Enclosure
Element specific commands (e.g., /ex/elementx).

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

//localhost>
```

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 Shell Object Messages section for more on show.

help *flush*

This command provides specific *flush* related help, illustrating various ways to use the *flush* command. See the Shell Object Messages section for more.

help *rescan*

This command provides specific *rescan* related help, illustrating various ways to use the *rescan* command. See the Shell Object Messages section for more.

help *update*

This command provides specific *update* related help. See the Shell Object Messages section for more.

help *commit*

This command provides specific *commit* related help, illustrating various ways to use the *commit* command. See the Shell Object Messages section for more.

help *focus*

This command provides specific *focus* related help, illustrating various ways to use the *focus* command. See the Shell Object Messages section for more.

help */cx*

This command provides specific controller */cx* related help, illustrating various commands associated with the controller */cx*. See the Controller Object Messages section for more.

help */cx/ux*

This command provides specific unit */cx/ux* related help, illustrating various commands to use on a unit */cx/ux*. See the Controller Object Messages section for more.

help */cx/px*

This command provides specific */cx/px* related help, illustrating various ways to use the */cx/px* command. See the Port Object Messages section for more.

help ***/cx/bbu***

This command provides specific */cx/bbu* related help, illustrating various ways to use the */cx/bbu* command. See the BBU Object Messages section for more.

help ***/ex***

This command provides specific enclosure */ex* related help, illustrating various commands associated with the enclosure */ex*. See the Enclosure Services Commands section for more.

help ***/ex/slotx***

This command provides specific slot */ex/slotx* related help, illustrating various ways to use the */ex/slotx* command. See the Enclosure Services Commands section for more.

help ***/ex/fanx***

This command provides specific fan */ex/fanx* related help, illustrating various ways to use the */ex/fanx* command. See the Enclosure Services Commands section for more.

help ***/ex/tempx***

This command provides specific temperature sensor */ex/tempx* related help, illustrating various ways to use the */ex/tempx* command. See the Enclosure Services Commands section for more.

Command Logging

This feature logs various changeable controller commands from both CLI and 3DM2 into a file.

Setting the environment variable TW_CLI_LOG to ON or OFF will enable or disable various controller commands logging into a log file called **tw_mgmt.log**.

Log environment:

When one of the following environment variable TW_CLI_LOG is set,

```
If Bash, then "export TW_CLI_LOG=ON"
If csh, then "setenv TW_CLI_LOG ON"
If Windows, then "set TW_CLI_LOG=ON"
```

Various changeable controller commands from both CLI and 3DM2 are logged into the log file **tw_mgmt.log**.

Log location:

In Linux and FreeBSD, the log file is in /var/log directory.

In Windows, the log file is in 3DM2 installation directory if 3DM2 is installed in the system; Otherwise, it is in the current user home directory.

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.

ERRATA

Meta-Character Warning:

If you wish to use CLI in single command mode (not interactive), make sure to avoid collision with your command interpreter (OS shell) by escaping the meta-characters (such as ?, <, >, @, &, *, etc) appropriately with single quote around them.

For example, given the

$ tw_cli /c0 ?

This is a case of single command usage where the user intends to get help on Controller related commands. While this is a valid CLI command, but since the arguments to CLI are first processed by the shell, then some shells like `csh(1)` will interpret the '?' as a meta-character to be used toward file completion and if no file is found with a single character, then shell will complain before the arguments are even passed down to CLI.

One solutions of this problem can be :

$ tw_cli help /cx

or

$ tw_cli '/c0 ?'

Note: Some of the OS shell does not have this problem such as bash.

Reporting Style

```
tw_cli(8) reporting has changed (hopefully for better). The intent has been to
provide a consistent tabular reporting so that relevant and important information
(such as B<info>) are made available as fast as possible. For example, firmware,
PCB, PCHIP and similar information have been removed from the info summary report,
as this type of information is not frequently needed.
```

The new style also accommodates automation much better by providing consistent columns with or without values so that it could be easily parsed. The intent is to make CLI yet another API (to approach it).

However to accommodate current automations around tw_cli and to ease the migration, the old behavior can still be requested by setting **TW_CLI_STYLE** environment variable to **OLD** as follows:

```
 If Bash, then "export TW_CLI_STYLE=OLD"
 If csh, then "setenv TW_CLI_STYLE OLD"
 if Windows, then "set TW_CLI_STYLE=OLD"
```

This backward compatibility window, will be communicated by official AMCC/3ware representatives.

Initialization Process Control

On the 9K series of controllers, the rebuild scheduling controls both rebuild and initialize processes if it is enabled. Currently, `tw_cli(8)` does not have any direct command to pause or resume an initialization process. If such action is needed, use the rebuild scheduling to handle it.

```
=head1 ENVIRONMENT VARIABLES
```

TW_CLI_STYLE setting this variable to OLD, will provide the old reporting style. TW_CLI_INPUT_STYLE setting this variable to OLD, will disable focus feature in the interactive mode.

AUTHOR

Marian Choy

SEE ALSO

```
 AMCC/3ware CLI User Guide
 AMCC/3ware User Guide
 AMCC/3ware Installation Guide
 http://www.amcc.com
```