VIOLIN SYSTEMS

Aria Command Line Interface Reference

7.5.0.1

Document Number: 530-0263-00 Rev 03

LEGAL NOTICE

Violin Systems LLC
2560 N. 1st Street, Suite 300
San Jose, CA 95131
USA

Contents

Preface

This preface outlines the organization of this book, describes document conventions, and provides information about additional resources.

Intended Audience

This guide is intended for experienced systems administrators. Violin Systems assumes that you are experienced in installing and servicing high-performance storage systems.

Contact Violin Systems Customer Support for any assistance with installing and servicing this system. See Contacting Violin Systems on page 4 for contact information.

What's New in This Version

Each software release includes release notes that identify new features in the software, as well known and resolved issues.

To obtain the most current version of the release notes, go to the Violin Support Center http://www.violinsystems.com/support-services/.

Note: Examine the release notes before you begin an installation and configuration process.

Document Organization

This guide is organized into the following sections:

- Chapter 1, Using the Aria Command Line Interface—Introduces the Aria CLI, describes the command modes, and lists syntax conventions and abbreviations.
- Chapter 2, Aria CLI Command List—Lists the commands in the Aria CLI, organized according to functional category.
- Chapter 3, Alphabetical Directory of Commands—Provides descriptions, syntax, and examples for each command in the Aria CLI.

Reference Documents

In addition to this guide, the following Violin Systems documents comprise the documentation suite that will assist you with setting up, using and servicing Violin Systems products. These guides are available for download from the Violin Systems Support site at https://www.violinsystems.com/support-services/.

This document...	Provides this information...
Release Notes	This document describes the new features, resolved issues, known limitations and software upgrade instructions for the current release.
XVS Installation Guide	This guide provides instructions for installing an XVS in an equipment rack and completing the system setup and configuration.
XVS User Guide	This guide provides instructions for managing, monitoring, and maintaining the XVS platforms using the Violin Symphony and Command Line Interface (CLI).
XVS Service Guide	This guide describes how to replace the system components in an XVS.
Concerto Command Line Interface Reference	This guide is a reference for the Concerto OS software commands used to configure, manage, and monitor the Memory Gateways.

Reference Documents

Document Conventions

Safety Icons

The table below summarizes warning, caution, and note icons used in this document and includes sample text.

Safety Icons

Icon	Sample Text
WARNING!	**WARNING!** Only authorized, qualified, and trained personnel should attempt to work on this equipment.
Caution:	**Caution:** Follow the listed safety precautions when working on the Violin device.
Note:	**Note:** Read through this entire chapter and plan your installation according to your location before installing the equipment. The following procedures and the order in which they appear are general installation guidelines only.

Typographical Conventions

The following typographic conventions are used in this guide:

Format	Meaning
Bold	Command names.
Italic	Provides emphasis and identifies document titles.
`Courier`	Examples and output.
`Courier bold`	Input you must type exactly as shown.
`<Courier>`	Information for which you must supply a value.
`[ ]`	Optional command parameters are enclosed within square brackets.
|	Separates a set of command choices from which only one may be chosen.
`{ }`	Required command parameters that must be specified are enclosed within curly brackets.

Typographical Conventions

Security

Violin Systems LLC, cannot be responsible for unauthorized use of equipment and will not make allowance or credit for unauthorized use or access.

Contacting Violin Systems

To obtain additional information or technical support for Violin Systems products, contact us at:

- Phone: 1-855-VIOLIN-5 (1-855-846-5465)
- International: +1 650-396-1500 Extension 3
- Web site: http://www.violinsystems.com
- Email: support@violinsystems.com

When contacting Violin Systems Customer Support, please have the following information available:

- Model and serial number of the system for which you are requesting support.
- Software version.
- A brief description of the problem.

CHAPTER 1 Using the Aria Command Line Interface

This chapter contains the following sections:

Note: This manual covers the commands available in the Aria CLI on the Array Controller Module (ACM). It does not cover the commands on the Violin Systems Gateway. For information about using the CLI on the Violin Systems Gateway (that is, the `isscli` commands), see the *Concerto Command Line Interface Reference*.

CLI Shorthand Method

Commands can be expressed in shorthand form in the CLI. Each keyword can be abbreviated by omitting its final letters, as long as the remaining letters are unique within the CLI command set. For example, the commands to display the system date and time or the hostname can be abbreviated as:

`sh clo`	`show clock`
`sh h`	`show hosts`

Additional letters can be included but none can be skipped; for example, the **show clock** command can be typed as `sho clo` or `sh clock` or various other combinations, but not as `shw clk`.

Other commands that are frequently typed in shorthand include:

`en`	`enable`
`conf t`	`configure terminal`
`ex`	`exit`

If the command is shortened too much, an error message appears, and help is offered. For example, the abbreviation `sh cl` could mean `show clock` or `show cluster` so it generates this error message:

```
> sh cl

% Ambiguous command "cl".

Type "sh cl?" for help.
```

Note: When scripting CLI commands, the shorthand versions should *not* be used, since commands that appear in a future release could potentially change the acceptable shorthand version of a given command.

Getting Help

In any mode of the CLI, you can query for help by using the **help** command or a question mark.

Enter `help` at the prompt for a summary of how to use question marks to obtain context-sensitive help, as described here. Just entering a question mark `?` by itself provides a list of available commands corresponding to the current mode. (Modes are described in Command Modes on page 8.)

You can also query for options of a specific command by typing in the command, following it with a space, and adding a question mark. After displaying a list of options, the command line echoes the string and puts the cursor after it, ready for more input.

For example, in Standard mode you can enter `cli ?` at the command line and see the following output.

```
> cli ?
clear-history          Clear the command history for the current user
session                Configure CLI options for this session only

> cli
```

If the command is complete without further options or values, `<cr>` is displayed on a separate line and the command is echoed at the prompt. Pressing the Enter key (also known as carriage return, `<cr>` will then issue the command if no values are required, or `<value required>` will be displayed.

For example:

```
> cli session ?
auto-logout         Configure keyboard inactivity timeout for automatic logout
paging              Configure the ability to view text one screen at a time
prefix-modes        Configure the CLI's prefix modes feature for this session
progress            Configure progress updates for long operations
terminal            Set terminal parameters
x-display           Set the display to use for X Windows applications

> cli session paging ?
enable              Enable paging

> cli session paging enable ?
<cr>

> cli session terminal ?
length              Set the number of lines for this terminal
resize              Resize the CLI terminal settings (to match with real terminal)
type                Set the terminal type
width               Set the width of this terminal in characters

> cli session terminal width ?
<number of characters>

> cli session terminal width 60
>
```

When `<value required>` is displayed and only specific values can be used (such as interface names or a port identifier), those values will be displayed on new lines after `<value required>`. Similarly, when the command is complete but could include additional options, `<cr>` and the options are displayed, each on a separate line.

Tab Completion of Commands

Commands can be completed by typing in the first few letters then pressing the Tab key (<tab>). Pressing Tab once completes the command if there is only one way to complete it; otherwise it expands the command to the next point of uncertainty. At that point, pressing Tab again displays a list of possible completions, which might be keywords or values, or both.

h<tab>	Completes the **help** keyword.
sh<tab>	Completes the show keyword (but s<tab> does not, because more than one command starts with the letter “s”).
show<tab>	Lists options of the **show** command that can immediately follow the **show** keyword.
sh<tab><tab>	Completes the **show** keyword and lists options of the **show** command.
s<tab><tab>	Lists all available commands that begin with the letter **s**.

For a list of all commands currently available, press the Tab key twice at the prompt. In Standard mode, for example, press the Tab key twice to list these commands:

```
> <tab><tab>
cli          exit         no           show         telnet       traceroute
enable       help         ping         slogin       terminal
```

Command Modes

The CLI can be in one of three modes, which determine the set of commands that can be executed. Commands that are not currently available do not show in help or completion, and generally behave as if they do not exist.

Standard Mode

When the CLI is launched, it begins in Standard mode. This is the most restrictive mode and only has commands to query a restricted set of state information. In this mode, you cannot take any actions that would directly affect the system, nor can you change any configuration.

User accounts with the **unpriv** role are restricted to Standard mode.

Enable Mode

The **enable** command moves the CLI to Enable mode. This mode has commands to view all state information and take certain kinds of actions, such as rebooting the system or configuring some system parameters, but it excludes commands that configure the cluster. Its commands are a superset of those in Standard mode.

The **exit** command (in Enable mode) closes the CLI. The **disable** command returns to Standard mode.

User accounts with the **monitor** role can use all Enable mode commands.

Config Mode

The **configure terminal** command moves the CLI from Enable mode to Configure (Config) mode.

- On the cluster's master node, Config mode has a full unrestricted set of commands to view anything, take any action, or change any configuration. Its commands are a superset of those in Enable mode.
- On nodes other than the master, Config mode only includes commands that operate on the local node. Using a global command on a standby or normal node either has a temporary local effect (which is overridden as soon as the node synchronizes with the master node) or produces an error message that identifies the master node where the command can be used.

The **exit** command moves the CLI from Config mode to Enable mode. Using the **exit** command twice closes the CLI, or you can use the **quit** command to close the CLI directly.

User accounts with the **admin** role can use all Config mode commands.

User accounts with the **security** role have access to `show` commands and the following additional set of configuration commands: `aaa`, `ldap`, `license`, `radius-server`, `tacacs-server`, `username`.

Prompt and Response Conventions

The prompt format is:

```
<hostname> [<cluster name>: <role>] <prompt>
```

The prompt begins with the hostname of the node and, in brackets, the cluster name and role of the node in that cluster (master, standby, normal, or unknown). The end of the prompt string indicates the command mode the CLI is in: `>` for standard mode, `#` for Enable mode, or `(config) #` for Config mode.

For example, if the hostname of the master node is `gate1` and the cluster name is `vmgCluster` then the prompts for each of the CLI modes are:

Standard mode:	`gate1 [vmgCluster: master] >`
Enable mode:	`gate1 [vmgCluster: master] #`
Config mode:	`gate1 [vmgCluster: master] (config) #`

The role can be master, standby, normal, or unknown.

An asterisk (*) before the command prompt indicates that some configuration changes have not yet been saved to the active configuration file.

For example, when changes need to be saved, the command prompt for Config mode changes to this:

Config (unsaved):	`* gate1 [vmgCluster: master] (config) #`

Most configuration commands that succeed in doing what was asked do not print any response, so the next thing you see after pressing Enter is another command prompt. You can verify the effect of a configuration command by using its corresponding **show** command to display current settings.

If an error occurs in executing a command, the response begins with `%` followed by some text describing the error.

Abbreviations for Large Numbers

The following abbreviations are used for large numbers in the output displays of various **show** and **stats** commands:

B	bytes
kB	kilobytes (1024^1 = 1,024 bytes)
MB	megabytes (1024^2 = 1,048,576 bytes)
GB	gigabytes (1024^3 = 1,073,741,824 bytes)
TB	terabytes (1024^4 = 1,099,511,627,776 bytes)
PB	petabytes (1024^5 = 1,125,899,906,842,624 bytes)

and so on for E (exabytes), Z (zettabytes), and Y (yottabytes). Single-letter abbreviations such as k, M, or G are sometimes used to conserve space, or for units other than bytes.

Key to Command Parameters

This section is a key to the meaning and format of parameter values and other attributes of the CLI commands. Parameter values are shown in angle brackets and listed alphabetically below.

`<cluster-id>`	A string specifying the name of a cluster.
`<domain-name>`	A domain name, such as `vmem.com`.
`<hostname>`	A hostname, such as `hexagon.vmem.com`.
`<ifname>`	An interface name, such as "eth0", "eth1", "lo" (loopback), and so on.
`<ip-address>`	An IPv4 address, such as `192.168.0.1`.
`<severity>`	A syslog logging severity level. Possible values, from least to most severe, are: “debug”, “info”, “notice”, “warning”, “error”, “crit”, “alert”, “emerg”.
`<mac-address>`	A MAC address. The segments may be 8 bits or 16 bits at a time, and may be delimited by ":" or ".". So you could say "11:22:33:44:55:66", "1122:3344:5566", "11.22.33.44.55.66", or "1122.3344.5566".
`<netmask>`	A network mask (such as “255.255.255.0”) or mask length prefixed with a slash (such as “/24”). These two express the same information in different formats.

`<network prefix>`	An IPv4 network prefix specifying a network. This is used in conjunction with a netmask to determine which bits are significant. For example, “192.168.0.0”.
`<port>`	A TCP or UDP port number.
`<reg-exp>`	An extended regular expression as defined by the grep man page. (The value you provide here is passed on to `grep -E`.)
`<url>`	Either a normal URL, using any protocol that wget supports, including HTTP, HTTPS, FTP, and TFTP; or a pseudo-URL specifying an SCP file transfer. The SCP pseudo-URL format is: `scp://username:password@hostname/path/filename` The path is an absolute path. Paths relative to the user’s home directory are not currently supported. The implementation of FTP does not support authentication, so you should use SCP if you require authentication. If you omit the `:password` part, you may be prompted for the password in a follow up prompt, where you can type it securely (without the characters being echoed). This prompt will occur only if the **cli default prompt empty-password** setting is `true`; otherwise, the CLI assumes you do not want any password. If you include the `:S` character, this is taken as an explicit declaration that the password is empty, and you will not be prompted in any case.

CHAPTER 2 Aria CLI Command List

This chapter lists the commands in the Aria Command Line Interface (CLI). Click on a command for more information about the command, including CLI syntax, usage notes, and examples.

The commands in this chapter are divided into the following general feature categories:

System Administration

Basic Administrative Commands

Command	Description
hostname	Sets the system hostname for the Violin Array.
terminal	Configures terminal settings for the current CLI session.
write terminal	Displays the running configuration on the screen.
write memory	Saves the running configuration to the active configuration file.
license install	Adds or removes keys for licensed features on the Violin Array.
license delete	Removes a license key for a specified licensed feature on the Violin Array.
show licenses	Displays licenses for features installed on the system, along with associated license keys
clock set	Configures the system clock on the Violin Array.
clock timezone	Sets the system time zone.
show clock	Displays the current system time, date, and configured time zone.
reload	Shuts down or reboots the Violin Array.
banner login	Specifies text that is displayed when a user logs into the Violin Array.
banner motd	Specifies a string of text to be displayed on the Violin Array as the message of the day (MOTD).
ftp-server enable	Enables or disables the FTP server on the Violin Array.
ftp-server ssl enable	Enables the SSL protocol for FTP server.
ftp-server ssl min-version	Sets the minimum version of TLS protocol used by the FTP server either to version TLS 1 or TLS 1.2.
show ftp-server	Indicates whether the FTP server has been enabled on the Violin Array.
help	Displays basic information about how to get information about commands in the Violin CLI.
show banner	Displays the text configured for the login banner and the message of the day (MOTD) banner.

Table 2.1 System Administration Commands

Command	Description
show running-config	Displays the running configuration on the screen.
show terminal	Displays information about the terminal settings for the current CLI session.
show version	Displays information about the system software running on the Violin Array.

Table 2.1 System Administration Commands

CLI Mode Commands

Command	Description
email upload	Exits Standard mode in the Violin Array CLI and enters Enable mode.
disable	Exits Enable mode in the Violin Array CLI and returns to Standard mode.
configure terminal	Enters Config mode on the Violin Array.
exit	Exits the current command mode on the Violin Array.

Table 2.2 CLI Mode Commands

Violin Web Interface Configuration

Command	Description
web enable	Enables or disables the Violin Web Interface on the device.
web http enable	Enables or disables HTTP access to the Violin Web Interface.
web https enable	Enables or disables HTTPS access to the Violin Web Interface.
web auto-logout	Specifies the amount of idle time allowed for the Violin Web Interface.
web http port	Specifies the TCP port used for HTTP access to the Violin Web Interface.
web http redirect	Causes HTTP requests made to the Violin Array to be redirected to HTTPS.
web httpd listen	Configures the system to accept HTTP connections only on specific interfaces.
web httpd ssl min-version	Sets the minimum version of TLS protocol used by the web server either to version TLS 1 or TLS 1.2.
web https certificate regenerate	Regenerates the certificate used for HTTPS connections.

Table 2.3 Web Interface Configuration Commands

Command	Description
web https port	Specifies the TCP port used for HTTPS access to the Violin Web Interface.
web proxy	Specifies settings for a Web proxy connection.
web proxy auth	Specifies authentication settings used for connecting to a Web proxy.
web session renewal	Specifies the length of time before Web session cookies are automatically regenerated.
web session timeout	Specifies the length of time before a Web session expires.
show web	Displays the settings for the Violin Web interface.

Table 2.3 Web Interface Configuration Commands

User Configuration

Command	Description
username	Creates or removes a local user account.
username capability	Sets the access privileges for a local user account.
username disable	Administratively disables a local user account, or modifies its login privileges.
username full-name	Applies a descriptive name to a local user account.
username nopassword	Enables access to a local user account without having to enter a password
username password	Configures the password for a local user account.
show usernames	Lists the local user accounts configured on the system, and displays information about the capabilities and status for each account.
show users	Lists information about the users currently logged into the system.
show users history	Displays the user login history for the Violin Array.
show whoami	Displays the username of the currently logged-in user, and the capabilities that user has.

Table 2.4 User Configuration Commands

SSH Commands

Command	Description
slogin	Initiates an SSH client connection to a specified host.
ssh client generate identity	Generates a new identity (DSAv2 private and public keys) for a specified user account.

Table 2.5 SSH Commands

Command	Description
ssh client global host-key-check	Configures how the system checks connecting SSH clients against its known hosts file
ssh client global known-host	Adds or removes entries in the system's global known hosts file.
ssh client identity user	Sets private and public keys for a specified user account.
ssh client user authorized-key	Adds or removes a key in the list of authorized SSHv2 RSA or DSA public keys for a user account.
ssh client user identity	Sets or generates RSAv2 and DSAv2 public and private keys for a user account.
ssh client user known-host remove	Removes entries in the known hosts file for a user account.
ssh server enable	Enables or disables the SSH server.
ssh server host-key	Sets or generates RSAv1, RSAv2, or DSAv2 public and private host keys for the SSH server.
ssh server listen	Configures the system to accept SSH connections only on specific interfaces.
ssh server min-version	Sets the version of SSH used by the SSH server, either version 1, or version 1 and 2.
ssh server ports	Specifies a list of one or more ports on which the SSH server listens for connections.
ssh server x11-forwarding	Enables X forwarding on the SSH server for connecting SSH clients.
show ssh client	Displays information about the configuration for SSH clients and keys for local user accounts.
show ssh server	Displays information about the configuration for the SSH server and host keys.

Table 2.5 SSH Commands

Telnet Commands

Command	Description
telnet	Invokes the Telnet client on the Violin Array.
show telnet-server	Indicates whether the Telnet server has been enabled on the Violin Array.

Table 2.6 Telnet Commands

Network Configuration

Basic Network Configuration

Command	Description
arp	Adds a static entry to the ARP cache on the Violin Array.
clear arp-cache	Removes the dynamic ARP entries from the ARP cache.
show arp	Displays the contents of the ARP cache.
ip default-gateway	Sets the default route for the Violin Array.
show ip default-gateway	Displays the currently active default route.
ip dhcp	Configures how the DHCP client on the Violin Array interacts with a DHCP server on the network,
show ip dhcp	Displays the DHCP configuration settings for the Violin Array.
ip domain-list	Adds a domain name to the list of domains that the Violin Array uses when resolving hostnames.
ip host	Configures static mappings between hosts and IPv4 addresses.
ip map-hostname	Ensures static host mapping for the current hostname.
show hosts	Shows values configured for host-related commands.
ip name-server	Configures the address of a name server to be used by the Violin Array.
ip route	Configures a static route on the Violin Array.
show ip route	Displays the routing table in the system, including dynamic routes and any active static routes.
service small-servers	Enables a number of TCP/UDP “small server” services.
show services small-servers	Displays the status of the TCP/UDP “small server” services

Table 2.7 Network Configuration Commands

Ethernet Configuration

Command	Description
interface alias	Sets an alias on a specified Ethernet interface on the Violin Array.
interface comment	Adds a comment to the configuration for an Ethernet interface on the Violin Array.
interface dhcp	Enables DHCP for an Ethernet interface on the Violin Array.
interface duplex	Configures the duplex setting for an Ethernet interface.
interface ip address	Sets the IP address and netmask for an Ethernet interface.
interface shutdown	Disables an Ethernet interface on the Violin Array.
interface speed	Sets the speed for an Ethernet interface on the Violin Array.
interface zeroconf	Enables zero configuration networking for an Ethernet interface.
show interfaces	Displays configuration information and traffic statistics for the Ethernet interfaces on the Violin Array.

Table 2.8 Ethernet Configuration Commands

SNMP Configuration

Command	Description
snmp-server enable	Activates SNMP or individual SNMP components on the Violin Array.
snmp-server community	Sets the community name required to be supplied with SNMP requests to the system.
snmp-server contact	Sets the syscontact variable served from the System MIB in MIB-II.
snmp-server host	Specifies information about hosts that will receive SNMP traps from the Violin Array.
snmp-server listen enable	Enables the interface listen list for SNMP connections.
snmp-server listen interface	Specifies the list of interfaces on which SNMP connections are accepted.
snmp-server location	Sets the syslocation variable served from the System MIB in MIB-II.
snmp-server port	Sets the UDP port for the SNMP agent.
snmp-server traps	Specifies settings for sending traps to hosts configured to receive them from the Violin Array.
snmp-server traps send-test	Sends a test SNMP trap to all configured trap sinks.
snmp-server user v3	Specifies identity and security parameters for an SNMPv3 user on the Violin Array.
snmp-server user v3 enable	Enables or disables an SNMPv3 user on the Violin Array.

Table 2.9 SNMP Commands

Command	Description
show snmp	Displays information about the SNMP configuration on the Violin Array.
show snmp engineID	Displays the value of the local SNMPv3 engine ID.
show snmp user	Displays information about SNMPv3 users configured on the Violin Array

Table 2.9 SNMP Commands

NTP Configuration

Command	Description
ntp enable	Enables or disables NTP on the Violin Array.
ntp disable	Disables or enables Network Time Protocol (NTP) on the Violin Array.
ntp peer	Configures settings for an NTP peer on the Violin Array.
ntp server	Configures settings for an NTP server on the Violin Array.
ntpdate	Sets the system clock using a specified NTP server.
show ntp	Displays NTP status on the Violin Array, along with information about the configured NTP servers and peers.

Table 2.10 NTP Configuration Commands

Cluster Configuration

Command	Description
cluster master	Configures interface, address, and auto-discovery settings for the master node in a cluster
cluster name	Sets the name of the cluster.
cluster port	Sets the service port for the cluster
show cluster	Displays cluster status on the Violin Array.
show cluster configured	Displays global cluster configuration settings on the Violin Array.

Table 2.11 Cluster Configuration Commands

Array Configuration

Command	Description
array balance	Starts or schedules RAID rebuilds for VCMs in a Violin Array.
array cooling hot	Sets a policy to adjust fan speed based on the temperature of the chassis.
array format capacity	Formats the Violin Array to a specific storage capacity.
array modules	Powers Violin Array modules or module types on and off; can also be used to establish a CLI connection to an internal Memory Gateway.
array reboot	Reboots the Violin Array or specific types of modules on the Violin Array.
array serial-logging	Enables serial logging of VCMs and Memory Gateways.
array shutdown	Shuts down all of the modules in the Violin Array and turns off the LEDs on the front panel.
array upgrade modules	Upgrades the software running on Violin Array modules.
array vimm-debug-log-collect	Causes VIMM statistics to be uploaded daily as part of log collection
pcie connect	Configures the PCIe routing mode for connecting the Violin Array to a Memory Gateway.
show pcie	Displays PCIe connection information for a Violin Array.
show array	Displays information about hardware components on a Violin Array.
show array balance	Displays the current RAID rebalance settings and the balance status of the VCMs in a Violin Array.
show array serial-logging	Indicates which MGs and VCMs on a Violin Array have serial logging enabled.
show out-of-service	Lists the VIMMs on a Violin Array where the `donotuse` flag has been activated.
show vimms	Displays status, alarm, and inventory information about VIMMs installed on a Violin Array.

Table 2.12 Violin Array Configuration Commands

Configuration File Management

Command	Description
configuration copy	Copies a configuration file to a specified target file.
configuration delete	Deletes a configuration file.
configuration fetch	Retrieves a configuration file from a remote location or USB drive and saves it on the Violin Array.
configuration jump-start	Runs the initial-configuration wizard.
configuration jump-start-file	Merges the common settings from a specified configuration file into the running configuration.
configuration move	Moves a configuration file to a specified target file.
configuration new	Creates a new configuration file.
configuration revert	Reverts the configuration to a previous version.
configuration switch-to	Loads a configuration from a specified file and changes it to be the active configuration file.
configuration text fetch	Downloads a text configuration file (list of CLI commands) from a remote location.
configuration text file	Manages configuration text files on the Violin Array.
configuration text generate	Creates a configuration text file from the current active configuration or a saved configuration.
configuration upload	Sends the active configuration or a specified configuration file to a remote location.
configuration write	Saves the running configuration to the active configuration file.
show configuration	Displays the active saved configuration for the Violin Array.
show configuration files	Lists the configuration files stored on the Violin Array and also can display the contents of a specified configuration file

Table 2.13 Configuration Files CLI Commands

Software Upgrade Commands

Command	Description
cluster upgrade	Upgrades the system software on the nodes in a cluster.
array upgrade staged vimms	Enables VIMMs to be upgraded as part of the non-disruptive upgrade (NDU) procedure.
cluster continue staged-upgrade	(A) Continues a suspended VIMM staged upgrade; (G) Completes a staged upgrade of a cluster by upgrading the software on the second half of the cluster.
cluster suspend staged-upgrade	Suspends an ongoing staged VIMM upgrade.
usb mount	Mounts a USB mass storage device used for upgrading a Violin Array.
usb eject	Unmounts and ejects a USB mass storage device used for upgrading a Violin Array.
monitor	Displays messages on the terminal screen as the VCMs on a Violin Array are booted or upgraded.
boot system	Specifies which boot partition supplies the software image loaded the next time the Violin Array is rebooted.
boot bootmgr password	Configures a password to control access to boot manager parameters.
boot next fallback-reboot	Configures the behavior of the Violin Array in the event that a software image fails to load correctly.
show bootvar	Displays information about the software image files and boot options on the Violin Array.
show images	Displays image information for the Violin Array.
show upgrade	Displays upgrade configuration status for VIMMs and VCMs.

Table 2.14 Software Upgrade Commands

Diagnostic Functions

Basic Diagnostic Commands

Command	Description
locate	Turns on the ID LED on the front and rear of the Violin Array.
ping	Sends ICMP echo requests to remote hosts on the network.
traceroute	Displays the list of hops a packet takes to reach a specified host.
monitor	Displays messages on the terminal screen as the VCMs and VIMMs in a Violin Array are booted or upgraded.
show alarms	Displays the active system alarms on a Violin Array.
show array modules	Displays status, alarm, and inventory information about individual modules or types of modules installed on a Violin Array.
show chassis info	Displays basic information about a Violin Array chassis.
show files system	Displays information about the used and available space on the `/config` and `/var` filesystems on the Violin Array.
show inventory	Lists information about all of the ACMs, VCMs, and VIMMs installed in a Violin Array.
show locate	Show the status of the ID LED on the front and rear of the Violin Array.
show memory	Displays information about memory usage on the Violin Array.
show system alarms	Displays the active system alarms on a Violin Array.

Table 2.15 Diagnostic CLI Commands

Sysdump File Commands

Command	Description
debug generate dump	Creates a debug dump (sysdump) file on the Violin Array.
file debug-dump	Manages debug dump (sysdump) files on the Violin Array.
file tcpdump	Manages TCP dump files on the Violin Array.
show files	Lists or displays the debug dump (sysdump), TCP dump, and statistics report files accumulated on the Violin Array.

Table 2.16 Sysdump File Commands

Violin Utilities

Command	Description
varray	Displays status information for a Violin Array.
vcounts	Displays data transfer counters for a Violin Array.
vdiag	Runs a series of diagnostic tests to get information about network connectivity, link status, configuration state, temperature, and active alarms for the modules installed in the Violin Array.
veeprom	Displays hardware information about the Violin Systems Gateway.
vincident	Collects information from the Violin Array that can be sent to Violin Customer Support to determine the cause of performance issues.
vinfo	Displays device and target information for the Violin Array.
vinventory	Lists information about all of the ACMs, VCMs, and VIMMs installed in a Violin Array.
vpartial	Displays the number of read/write I/O requests processed and the number of partial 4kB flash pages.
vring	Displays the Violin Array DMA ring buffer.
vstat	Displays the status of the connection and the ready status of a Violin Array.
vvimms	Displays status information for Violin Array VIMMs.

Table 2.17 Violin Utilities

AAA Commands

Authentication Commands

Command	Description
aaa authentication login default	Specifies one or more authentication methods a user is subject to when logging into the system.
show aaa	Displays the configuration settings for Authentication and Authorization.

Table 2.18 Authentication Commands

Authorization Commands

Command	Description
aaa authorization map default-user	Specifies the name of a local account whose privileges are granted to users who log in using a non-local authentication method.
aaa authorization map order	Specifies how the system uses attributes received from a remote authentication server to map authenticated users to a local user account.
show aaa	Displays the configuration settings for Authentication and Authorization.

Table 2.19 Authorization Commands

System Logging

Command	Description
logging	Configures the Violin Array to send syslog messages to a remote syslog server.
logging fields seconds	Adds and configures a “seconds” field in logging event messages.
logging files delete	Deletes log files accumulated on the system.
logging files rotation criteria	Specifies the conditions for automatically rotating stored log files.
logging files rotation force	Forces an immediate rotation of the log files.
logging files rotation max-num	Sets how many log files are kept on the system.
logging files upload	Transfers a current or archived log file to a specified remote host.
logging files upload-auto	Enables automatic uploading of log files to a remote host.
logging files upload-auto immediate	Performs a one-time upload of logs to the configured destination site.
logging format	Sets the format for log messages on the system.
logging level audit mgmt	Specifies the severity of log messages that get placed in the audit log.
logging level cli commands	Sets the severity level at which CLI commands that the user executes are logged.
logging local	Sets the minimum severity of log messages to be saved in log files on local persistent storage.
logging local override class	Sets or removes a per-class override on the logging level.
logging receive	Enables the system to receive log messages from another host.

Table 2.20 System Logging Commands

Command	Description
logging trap	Sets the minimum severity of log messages sent to syslog servers.
logging local override class	Sets or removes a per-class override on the logging level.
show log	Displays the contents of the current log file.
show log continuous	Shows an ongoing display of the current log file.
show log files	Displays the contents of locally archived log files.
show logging	Displays configuration settings for the system logging feature.
show logging files upload-auto	Displays configuration settings for automatic uploading of log files to a remote host.

Table 2.20 System Logging Commands

E-mail Alert Notifications

Command	Description
email auth	Enables SMTP authentication for the Violin Array, and sets the username and password to be used for SMTP authentication.
email callhome	Configures the Violin Array to generate and send automatic support notifications over e-mail to Violin Technical Support, as well as which events generate the e-mails.
email consolidate	Enables and configures settings for e-mail alert consolidation.
email dead-letter	Sets how the system handles e-mails that cannot be sent.
email domain	Sets the domain name to be used as the source for e-mail notifications.
email mailhub	Specifies the hostname or IP address of the mail relay to be used to send notification e-mails.
email mailhub-port	Specifies the port to use with the mail relay for sending notification e-mails.
email notify event	Enables e-mail notifications for specified events.
email notify recipient	Specifies e-mail addresses to receive notification e-mails.
email return-addr	Specifies the return address for the e-mails sent from the Violin Array.
email return-host	Specifies whether to include the system hostname in the return address for the e-mails sent from the Violin Array.
email send-test	Sends a test e-mail to all of the configured notification e-mail recipients.

Table 2.21 E-mail Notification Commands

Command	Description
email ssl min-version	Sets the minimum version of TLS protocol used by the email server either to version TLS 1 or TLS 1.2.
show email	Displays the settings for sending notification e-mails when various informational or failure events occur on the Violin Array.

Table 2.21 E-mail Notification Commands

Statistics Reporting

Command	Description
file stats	Manages statistics report files on the Violin Array.
show stats alarm	Displays the status and configuration settings for statistics-based alarms on the Violin Array.
show stats chd	Displays configuration settings for computed historical datapoint (CHD) datasets.
show stats cpu	Displays utilization statistics for the CPUs on the Violin Array.
show stats sample	Displays the enabled/disabled status and sampling interval for sample datasets on the Violin Array.
stats alarm	Enables and configures thresholds for alarms on the Violin Array.
stats alarm clear	Clears a specified alarm on the Violin Array.
stats alarm rate-limit	Configures settings for limiting the number of alarm events that are generated over specific time periods.
stats alarm rate-limit reset	Resets the rate-limiting counters and time for the specified alarm.
stats chd	Configures a computed historical datapoint (CHD) dataset.
stats chd clear	Clears all of the data collected for a specified computed historical datapoint (CHD) dataset.
stats clear-all	Clears data for all sample datasets, CHD datasets, and resets status for all alarms.
stats export	Exports collected statistics to a CSV (comma-separated value) file.
stats sample	Enables data collection for a specified sample dataset and configures its sampling interval.
stats sample clear	Clears all of the data collected for a specified sample dataset.

Table 2.22 Statistics Reporting Commands

Scheduled Jobs

Command	Description
job command	Configures CLI commands to be executed in a scheduled job.
job comment	Adds a comment string to a scheduled job.
job enable	Enables a job for execution.
job execute	Executes a specified job.
job fail-continue	Configures a job to continue executing its CLI commands in the event that a command fails.
job name	Assigns a name to a job.
job schedule	Sets an execution schedule for a job.
show jobs	Displays command sequences, schedules, and configuration settings for configured jobs.

Table 2.23 Scheduled Jobs CLI Commands

CLI Configuration

Command	Description
cli session terminal	Configures terminal settings for the current CLI session.
cli default auto-logout	Specifies the default amount of idle time allowed for CLI sessions on the Violin Array.
cli default paging	Enables the capability to view CLI output one page at a time.
cli default progress	Configures progress reports to appear at the end of each page when long streams of CLI output are displayed.
cli default prompt	Configures whether the Violin Array prompts for confirmation before it performs certain operations.
cli default show config-hidden	Causes all hidden CLI commands to be visible.
cli session auto-logout	Specifies the amount of idle time allowed for the current CLI session.
cli session paging	Enables the capability to view CLI output one page at a time, for the current CLI session only.
cli session progress	Configures progress reports to appear at the end of each page when long streams of CLI output are displayed, for the current CLI session only.
cli session x-display	Sets the display to use for X Window applications, such as VNC viewer, for virtual machines running on the Violin Array.
cli clear-history	Clears the command history for the current user.
show cli	Displays the settings configured for CLI sessions on the Violin Array.

Table 2.24 CLI Configuration Commands

CHAPTER 3 Alphabetical Directory of Commands

This chapter lists all of the commands available on the Array Controller Module (ACM) on the XVS.

aaa authentication login default

The **aaa authentication login default** command specifies one or more authentication methods a user is subject to when logging into the system.

Syntax

```
[no] aaa authentication login default <method-list>
```

where `<method-list>` is one or more authentication methods. The following authentication methods are valid:

`local`	Local authentication
`radius`	RADIUS authentication
`tacacs+`	TACACS+ authentication
`ldap`	LDAP authentication

The system attempts to authenticate the user using the methods specified in the command. The order in which the methods are specified is the order in which the authentication is attempted.

Command Mode

Config mode

Example

The following shows an example of an authentication method list where a user is first authenticated using the local user database. If local authentication fails, then RADIUS authentication is performed.

```
(config) # aaa authentication login default local radius
```

aaa authentication login local enable

The **aaa authentication login local enable** command enables local authentication for users logging into the Violin Array. When local authentication is enabled, users are authenticated using the local user database.

Syntax

```
[no] aaa authentication login local enable
```

Command Mode

Config mode

Example

The following example enables local authentication on the Violin Array.

```
(config) # aaa authentication login local enable
```

aaa authorization map default-user

The **aaa authorization map default-user** command specifies the name of a local account whose privileges are granted to users who log in using a non-local authentication method.

When a user is authenticated using a non-local method (such as RADIUS or TACACS+) and does not have a local account, this command specifies which local account the authenticated user will be logged on with. If the user has a local account, this mapping is ignored.

How the remote-to-local account mapping is used depends on the setting of the **aaa authorization map order** command. It is used if the **aaa authorization map order** command is set to **local-only**, or if the **aaa authorization map order** command is set to **remote-first**, and no valid mapping attribute is returned from the authentication server.

Syntax

```
[no] aaa authorization map default-user <user account>
```

where `<user account>` is a user account configured on the system.

Command Mode

Config mode

Example

The following example sets the authorization map account to **admin**. When a user logs onto the system using a non-local authentication method, the user is granted the privileges of the admin user.

```
(config) # aaa authorization map default-user admin
```

aaa authorization map order

The **aaa authorization map order** command specifies how the system uses attributes received from a remote authentication server to map authenticated users to a local user account.

Syntax

```
[no] aaa authorization map order remote-first | remote-only | local-only
```

where:

`remote-first`	If a local-user mapping attribute is returned from the authentication server, and it is a valid local username, map the authenticated user to the local user specified in the attribute. Otherwise, if the attribute is not present or not valid locally, map the authenticated user to the account specified by the **aaa authorization map default-user** command. (This is the default behavior.)
`remote-only`	Only try to map a remote authenticated user if the authentication server sends a local-user mapping attribute. If the local-user mapping attribute does not specify a valid local user, no further mapping is tried.
`local-only`	All remotely authenticated users will be mapped to the user specified by the **aaa authorization map default-user** command. Any vendor attributes received from the authentication server are ignored.

Command Mode

Config mode

Example

The following example configures the system to map remotely authenticated users to the account specified by the **aaa authorization map default-user** command, regardless of whether a local-user mapping attribute was received from the authentication server. In this example, remotely authenticated users are granted the privileges of the admin user.

```
(config) # aaa authorization map order local-only
(config) # aaa authorization map default-user admin
```

arp

The **arp** command adds a static entry to the ARP cache on the Violin Array.

Syntax

```
arp <ip-address> <mac-address>
no arp <ip-address>
```

The **no** form of the command removes the static ARP entry from the ARP cache. Only static ARP entries can be removed from the ARP cache. To remove dynamic ARP entries, use the **clear arp-cache** command.

Command Mode

Config mode

Example

The following example adds a static entry to the ARP cache.

```
(config) # arp 10.1.9.1 00:1f:c9:53:ea:f2
```

array balance

The **array balance** command starts a RAID rebuild for each VCM in a Violin Array that requires a rebuild, or can enable automatic RAID rebuilds when the system detects an unbalanced RAID group. You can also set up RAID rebuilds to occur on a scheduled basis.

A RAID group is considered “unbalanced” if two or more VIMMs in a RAID group share the same root VIMM. This may cause the system to perform less than optimally. Balance of the system can be restored by “moving” VIMMs from RAID rebuilds. If necessary, multiple RAID rebuilds can be performed at the same time. Note that a RAID rebuild can take three hours or longer.

Syntax

```
array balance
[no] array balance enable
array balance schedule once time <hh>:<mm>:<ss> [date <yyyy>/<mm>/<dd>]
no array balance schedule once
array balance schedule weekly [day-of-week <day>] [time <hh>:<mm>:<ss>]
no array balance schedule weekly
```

where:

`enable`	Enables automatic balancing. When unbalanced RAID groups are detected, they are automatically rebuilt. Automatic balancing is enabled by default. The **no** form of the command disables automatic balancing.
`schedule once`	Schedules a one-time RAID rebuild at the specified time and date. The **no** form of the command cancels the scheduled RAID rebuild.
`schedule weekly`	Schedules RAID rebuilds to occur weekly on a specified day of the week at a specified time. If you do not specify a day and time, the RAID rebuilds occur on Saturdays at 1:00 a.m. The **no** form of the command cancels the scheduled RAID rebuilds.

Entering the **array balance** command with no options immediately starts a starts a RAID rebuild for each VCM that requires a rebuild. More than one rebuild may be required to balance the system.

Command Mode

Config mode

Examples

The following example starts a rebuild of any unbalanced RAID groups. If any RAID groups are unbalanced, a single rebuild is performed to restore balance. You may need to run the command more than once to complete the rebalancing process.

```
(config) # array balance
```

The following example configures the system to start a RAID rebuild every Sunday at 6:00 a.m.

```
(config) # array balance schedule weekly day-of-week sun time 06:00:00
```

The following example disables automatic RAID rebalancing, which is enabled by default.

```
(config) # no array balance enable
```

array cooling hot

The **array cooling hot** command sets the cooling policy for the Memory Array. The cooling policy adjusts the speed of the fans based on the temperature of the chassis.

Syntax

`array cooling hot <device> temp <temperature>`

where:

`<device>`	Device is the acm, ambient, mg, vcm, or vimm
`<temperature>`	Temperature is the per device threshold. When this threshold is reached, the fans are raised to a higher speed to cool the Memory Array.

Example

In this example, the temperature of the vcm is changed from 67 to 66.

The vcm temperature is **67**, as shown by the `show array cooling` command:

```
# show array cooling
Temperature Thresholds
  acm         : cool 65, hot 67, alarm 70
  mg          : cool 65, hot 67, alarm 70
  vcm         : cool 64, hot 67, alarm 70
  vimm        : cool 65, hot 67, alarm 70
  ambient     : cool 30, hot 32, alarm 35
```

Change the vcm temperature from **67** to **66**.:

```
# array cooling hot vcm temp 66
Temperature Thresholds
  acm         : cool 65, hot 67, alarm 70
  mg          : cool 65, hot 67, alarm 70
  vcm         : cool 64, hot 66, alarm 70
  vimm        : cool 65, hot 67, alarm 70
  ambient     : cool 30, hot 32, alarm 35
```

Support Devices

Violin Systems Array

array format capacity

The **array format capacity** command formats the Violin Array to a specific storage capacity. Using this command, it is possible to format the storage capacity for all four of the Violin Array's vRAID Controllers into one large array entity.

Note that the Violin Array is pre-formatted to specific storage capacities, depending on the type of VIMMs in the system. A single level cell (SLC) system is formatted to 65%, and a multi-level cell (MLC) system is formatted to 84%. For optimum system performance, do not change these values.

Syntax

```
array format capacity <percentage>
```

where `<percentage>` is the storage capacity percentage for the system. Supported storage capacity percentages are 50%, 65%, 78%, 84%, and 87%. The recommended storage capacity percentages are 65 for SLC systems and 84 for MLC systems.

Command Mode

Config mode

Example

The following example sets the storage capacity percentage for the system to 78%.

```
(config) # array format capacity 78
```

array modules

The **array modules** command performs operations on specified Violin Array modules or module types, such as powering individual modules on and off.

Syntax

```
[no] array modules id <module-id> enable
no array modules id <module-id> out-of-service
array modules id <mg-id> slogin
[no] array modules type <module-type> enable
array modules id <module-id> erase
```

where:

`<module-id> enable`	Powers on the module with the specified `<module-id>`. The `<module-id>` can be a VIMM, VCM, Memory Gateway, or host bus adapter. The **no** form of the command powers off the specified module and activates the `out-of-service` flag for the specified VIMM or the VIMMs managed by the specified VCM. The `out-of-service` flag indicates that the module should not be used or may need to be replaced. If errors occur during the module enable/disable process, they are displayed within progress bars on screen.
`out-of-service`	When specified with the **no array modules id** command, clears the `out-of-service` flag for the specified VIMM or the VIMMs managed by the specified VCM, then power-cycles the VCM.
`<mg-id> slogin`	Establishes a CLI session to the internal Memory Gateway with the specified `<mg-id>`.
`<module-type> enable`	Powers on all modules of the specified `<module-type>`. The `<module-type>` can refer to all of the VIMMs, VCMs, internal Memory Gateways, Array Controller Modules, Front Panel Modules, or host bus adapters on the system. The **no** form of the command powers off all of the modules of the specified type.
`<module-id> erase`	Erases all user data on the specified `<module-id>`, preventing its use for potential RAID correction. The `<module-id>` must be a VIMM in either the “maintenance” or “recovery candidate” state. Once erased, the VIMM is available for use as a spare in a full RAID rebuild.

Command Mode

Config mode

Examples

The following example powers on one of the VIMMs installed in the Violin Array.

```
(config) # array modules id vimm24 enable
```

The following example powers off the VIMM and activates the `out-of-service` flag for the VIMM.

```
(config) # no array modules id vimm24 enable
```

The following example clears the `out-of-service` flag for the VIMM.

```
(config) # no array modules id vimm24 out-of-service
```

The following example powers off all of the VIMMs in the Violin Array.

```
(config) # no array modules type vimm enable
```

array reboot

The **array reboot** command reboots the Violin Array or specific types of modules on the Violin Array.

Syntax

```
array reboot [acms-only] [mgs-only] [vcms-only]
array reboot <mg-id> | <acm-id>
```

where:

`<acm-id>`	Reboots only the specified ACM.
`acms-only`	Reboots the Array Controller Modules on the Violin Array.
`<mg-id>`	Reboots only the specified MG.
`mgs-only`	Reboots the internal Memory Gateways on the Violin Array.
`vcms-only`	Reboots the vRAID Controller Modules and VIMMs on the Violin Array.

Entering the **array reboot** command with no options reboots all components of the Violin Array.

Command Mode

Config mode

Examples

The following example reboots the Violin Array.

```
(config) # array reboot
```

The following example reboots the internal Memory Gateways in the Violin Array.

```
(config) # array reboot mgs-only
```

array serial-logging

The **array serial-logging** command enables serial logging of VCMs and Memory Gateways.

Syntax

```
array serial-logging {vcm-a | vcm-b | vcm-d | mg-a | mg-b | all vcms | all
mgs | all}
```

array shutdown

The **array shutdown** command shuts down all of the modules in the Violin Array and turns off the LEDs on the front panel. This command must be entered on the Master ACM.

Syntax

```
array shutdown
```

Command Mode

Config mode

Example

The following example shuts down the Violin Array.

```
(config) # array shutdown
```

array upgrade modules

The **array upgrade modules** command upgrades the software running on Violin Array modules. You can upgrade the software on a specific VCM, or you can upgrade the software on all VCMs at once.

Syntax

```
array upgrade modules {id <module-id> | type vcm}
```

where:

`id <module-id>`	Upgrades the specified VCM.
`type vcm`	Upgrades all of the VCMs on the Violin Array.

Command Mode

Config mode

Examples

The following example upgrades a VCM on the Violin Array.

```
(config) # array upgrade modules id vcm-a
```

The following example upgrades all of the VCMs on the Violin Array.

```
(config) # array upgrade modules type vcm
```

array upgrade staged vimms

The **array upgrade staged vimms** command enables VIMMs to be upgraded as part of the non-disruptive upgrade (NDU) procedure.

Syntax

```
[no] array upgrade staged vimms
```

The **no** form of the command disables the VIMMs from being upgraded during the NDU procedure. This is the default setting.

Command Mode

Config mode

Example

The following example enables a staged upgrade of VIMMs.

```
(config) # array upgrade staged vimms
```

array vimm-debug-log-collect

The **array vimm-debug-log-collect** command causes VIMM statistics to be uploaded daily as part of log collection.

Note: This command is for Violin Systems Customer Support use only.

Syntax

```
[no] array vimm-debug-log-collect enable
```

The **no** form of the command disables VIMM log collection. VIMM log collection is disabled by default.

Command Mode

Config mode

Example

The following example enables VIMM log collection on a Violin Array.

```
(config) # array vimm-debug-log-collect enable
```

banner login

The **banner login** command specifies text that is displayed when a user logs into the Violin Array. The command sets the contents of the `/etc/issue` and `/etc/issue.net` files.

Syntax

```
banner login <message-text>
no banner login
```

To specify more than one word in the `<message-text>`, enclose the words in quotes. The **no** form of the command removes the banner from the configuration.

Command Mode

Config mode

Example

The following example configures a login banner for the Violin Array.

```
(config) # banner login "Welcome to Violin"
```

banner motd

The **banner motd** command specifies a string of text to be displayed on the Violin Array as the message of the day (MOTD). The command sets the contents of the `/etc/motd` file.

Syntax

```
banner motd <message-text>
no banner motd
```

To specify more than one word in the `<message-text>`, enclose the words in quotes. The **no** form of the command removes the banner from the configuration.

Command Mode

Config mode

Example

The following example configures a MOTD banner for the Violin Array.

```
(config) # banner motd "Go Giants!"
```

boot bootmgr password

The **boot bootmgr password** command configures a password to control access to boot manager parameters.

Syntax

```
boot bootmgr password [<cleartext-password>]
boot bootmgr password 0 [<cleartext-password>]
boot bootmgr password 7 <encrypted-password>
no boot bootmgr password
```

where:

`<cleartext-password>`	Specifies a password in clear text.
`0`	Indicates the password will be specified in cleartext.
`7`	Indicates the password will specified in encrypted form.
`<encrypted-password>`	Specifies a password in encrypted form. Two types of password encryption are supported, SHA-1 and MD5.

If you enter the **boot bootmgr password** or **boot bootmgr password 0** commands and press the Return key, the CLI prompts you for the cleartext password. Use this as an alternative to entering the password on the command line.

Command Mode

Config mode

Examples

The following examples set the boot manager password to “violin”.

```
(config) # boot bootmgr password 0 violin
```

```
(config) # boot bootmgr password
Password: violin
 Confirm: violin
```

The following example specifies the boot manager password in encrypted format.

```
(config) # boot bootmgr password 7 $6$DOFZf9UV$zKRV0WvjA09.VjO9SV7p22M
```

boot next fallback-reboot

The **boot next fallback-reboot** command configures the behavior of the Violin Array in the event that a software image fails to load correctly. By default, if booting the image in one partition fails, then the system boots using the image in the other partition as a fallback. The **no** form of this command disables this behavior.

Disabling the boot fallback function is useful if you need to downgrade the software image on the Violin Array; it prevents the system from loading the software image in the other partition when the downgraded software fails to apply the configuration.

This command applies to the next boot only; after the next boot, the boot fallback function returns to the default behavior.

Syntax

```
[no] boot next fallback-reboot enable
```

By default, the boot fallback function is enabled. The **no** form of the command disables it, which prevents the Violin Array from booting with the image in the other partition if the initial boot fails.

Command Mode

Enable mode, Config mode

Example

The following example disables the boot fallback function for the next reboot of the Violin Array.

```
# no boot next fallback-reboot enable
```

boot system

The **boot system** command specifies which of the two boot partitions supplies the software image loaded the next time the Violin Array is rebooted.

Syntax

```
boot system location <partition>
[no] boot system next
```

The `<partition>` is one of the two boot partitions on the Violin Array. The **next** option sets the boot partition used for the next reboot to be the next one after the partition used for the last reboot.

The **no** form of the command resets the next boot setting to the default, which is to boot using the same partition that was used for the last reboot.

Command Mode

Config mode

Example

The following example configures the Violin Array to load the image in boot partition 2 the next time the device is rebooted.

```
(config) # boot system location 2
```

clear arp-cache

The **clear arp-cache** command removes the dynamic ARP entries from the ARP cache. Only dynamic ARP entries are removed with this command. To remove static ARP entries, use the **no arp** command.

Syntax

```
clear arp-cache
```

Command Mode

Enable mode, Config mode

Example

The following example clears the dynamic ARP entries from the ARP cache.

```
# clear arp-cache
```

cli clear-history

The **cli clear-history** command clears the command history for the current user. (To view the command history, press the up-arrow key.)

Syntax

```
cli clear-history
```

Command Mode

Enable mode, Config mode

Example

The following example clears the command history for the current user.

```
# cli clear-history
```

cli default auto-logout

The **cli default auto-logout** command specifies the default amount of idle time allowed for CLI sessions on the Violin Array. After the specified amount of time has elapsed, the user is automatically logged out.

Syntax

```
cli default auto-logout <minutes>
no cli default auto-logout
```

where:

`<minutes>`	Is the number of minutes of inactivity before the user is logged out of the Violin CLI. You can specify from 0 – 525,600 minutes (one year). Specifying 0 disables auto-logout. The default is 150 minutes.

The **no** form of the command disables auto-logout.

Command Mode

Config mode

Example

The following example sets the timeout value for the Violin CLI to 60 minutes.

```
(config) # cli default auto-logout 60
```

cli default paging

The **cli default paging** command enables the capability to view CLI output one page at a time. After a page of CLI output is displayed on the screen, the CLI user is prompted to display the next page, and so on until the CLI output is complete. The paging setting does not apply to output generated by a script.

Syntax

```
[no] cli default paging enable
```

The **no** form of the command disables CLI paging. CLI paging is enabled by default.

Command Mode

Config mode

Example

The following example enables paging for CLI output.

```
(config) # cli default paging enable
```

cli default progress

The **cli default progress** command configures progress reports to appear at the end of each page when long streams of CLI output are displayed.

Syntax

```
[no] cli default progress enable
```

The **no** form of the command disables CLI progress reports. CLI progress reports are enabled by default.

Command Mode

Config mode

Example

The following example enables progress reports for long streams of CLI output.

```
(config) # cli default progress enable
```

cli default prompt

The **cli default prompt** command configures whether the Violin Array prompts for confirmation before it performs certain operations, such as rebooting, resetting to factory defaults, or exiting the CLI while there are unsaved configuration changes.

Syntax

```
[no] cli default prompt {confirm-reload | confirm-reset | confirm-unsaved
| empty-password}
```

where:

`confirm-reload`	Prompts for confirmation before rebooting the device.
`confirm-reset`	Prompts for confirmation before resetting the configuration to factory defaults.
`confirm-unsaved`	Prompts for confirmation before rebooting when there are unsaved configuration changes.
`empty-password`	Enables prompting for a password in cases where a password is permitted, but the user has not supplied one. This applies to pseudo-URLs of the form: `{scp or sftp}://username[:password]@hostname/filename` where the `:password` part was omitted. If this option is enabled, the CLI asks for a password to be entered. If the prompt is disabled, the CLI assumes there is no password.

The **no** form of the command disables prompting for the specified option.

Command Mode

Config mode

Example

The following example configures the Violin Array to prompt for confirmation before rebooting.

```
(config) # cli default prompt confirm-reload
(config) # reload
Confirm reload? [yes]
```

cli default show config-hidden

The **cli default show config-hidden** command causes all hidden CLI commands to be visible.

Syntax

```
[no] cli default show config-hidden enable
```

The **no** form of the command re-hides the hidden commands.

Command Mode

Config mode

Example

The following example makes all commands visible in the CLI.

```
(config) # cli default show config-hidden enable
```

cli session auto-logout

The **cli session auto-logout** command specifies the amount of idle time allowed for the current CLI session. After the specified amount of time has elapsed, the user is automatically logged out.

Syntax

```
cli session auto-logout <minutes>
no cli session auto-logout
```

where:

`<minutes>`	Is the number of minutes of inactivity before the user is logged out of the Violin CLI. You can specify from 0 – 525,600 minutes (one year). Specifying 0 disables auto-logout. The default is 150 minutes.

The **no** form of the command disables auto-logout.

Command Mode

Enable mode, Config mode

Example

The following example sets the timeout value for the current CLI session to 60 minutes.

```
(config) # cli session auto-logout 60
```

cli session paging

The **cli session paging** command enables the capability to view CLI output one page at a time, for the current CLI session only. After a page of CLI output is displayed on the screen, the CLI user is prompted to display the next page, and so on until the CLI output is complete.

Syntax

```
[no] cli session paging enable
```

The **no** form of the command disables CLI paging. CLI paging is enabled by default.

Command Mode

Enable mode, Config mode

Example

The following example enables paging for CLI output for the current CLI session.

```
(config) # cli session paging enable
```

cli session progress

The **cli session progress** command configures progress reports to appear at the end of each page when long streams of CLI output are displayed. This command applies to the current session only.

Syntax

```
[no] cli session progress enable
```

The **no** form of the command disables CLI progress reports. CLI progress reports are enabled by default.

Command Mode

Enable mode, Config mode

Example

The following example enables progress reports for long streams of CLI output for the current CLI session.

```
(config) # cli session progress enable
```

cli session terminal

The **cli session terminal** command configures terminal settings for the current CLI session. The settings configured with this command override the settings auto-detected for the terminal by the Violin Array.

Syntax

```
cli session terminal {length <lines> | resize | type <terminal-type> |
width <characters>}
no cli session terminal type
```

where:

`length <lines>`	Sets the number of lines per page for the terminal.
`resize`	Detects the size of the terminal and adjusts to the appropriate settings.
`type <terminal-type>`	Sets the terminal type. Enter **cli session terminal type ?** for a list of supported terminal types.
`width <characters>`	Sets the maximum number of characters per line for the terminal.

The **no** form of the command clears the terminal type setting, causing the terminal configuration for the session to be equivalent to a “dumb terminal”.

Command Mode

Enable mode, Config mode

Example

The following example configures the number of lines per page for the current CLI session.

```
(config) # cli session terminal length 128
```

cli session x-display

The **cli session x-display** command sets the display to use for X Window applications, such as VNC viewer, for virtual machines running on the Violin Array.

Syntax

```
cli session x-display full <display>
no cli session x-display
```

where:

`<display>`	Specifies the display to use; for example, `localhost:0.0`.

The **no** form of the command disables the display.

Command Mode

Enable mode, Config mode

Example

The following example sets command sets the display to use for X Window applications.

```
(config) # cli session x-display full localhost:0.0
```

clock set

The **clock set** command configures the system clock on the Violin Array.

Syntax

```
clock set <hh>:<mm>:<ss> [<yyyy>/<mm>/<dd>]
```

Setting the date is optional. If you do not specify the date, it is not changed from the current system date.

Command Mode

Config mode

Example

The following example configures the time and date on the Violin Array.

```
(config) # clock set 12:12:12 2018/11/03
```

clock timezone

The **clock timezone** command sets the system time zone.

Syntax

```
clock timezone <continent> <city>
clock timezone <continent> <country> <city>
clock timezone <continent> <region> <country> <city>
clock timezone <ocean> <island>
clock timezone UTC
clock timezone UTC-offset <offset-from-UTC>
no clock timezone
```

The system time zone may be entered in a number of ways: as a specific city, region, time zone name, UTC, or an offset from UTC. The **no** form of the command removes the specified time zone from the configuration.

Command Mode

Config mode

Examples

The following example configures the system time zone as U.S. Pacific time.

```
(config) # clock timezone America North United_States Pacific
```

The following example configures the system time zone as UTC -8 hours.

```
(config) # clock timezone UTC-offset UTC-8
```

The following example configures the system to use the time zone in Nome, Alaska.

```
(config) # clock timezone America North United_States Other Nome
```

cluster continue staged-upgrade

The **cluster continue staged-upgrade** command completes a staged upgrade of a cluster, upgrading the software on the second half of the cluster. You enter this command on the cluster master after the first half of the cluster has been upgraded successfully.

If issued during a non-disruptive upgrade (NDU) that is currently paused due to a system interruption, this command will resume the NDU.

Syntax

```
cluster continue staged-upgrade
```

Command Mode

Enable mode, Config mode

Example

After the first half of the cluster has been upgraded, enter the following command on the cluster master to upgrade the second half of the cluster.

```
# cluster continue staged-upgrade
```

cluster id

The **cluster id** command resets the identifier for a cluster to 15000-0000-0000. This command allows you to reset the cluster ID to the default value in case an Array has a different cluster ID.

Syntax

```
cluster id 15000-0000-0000 [global]
no cluster id
```

The **no** form of the command resets the cluster ID to the system default. The **global** option, when specified on the cluster master, sets the cluster ID on all of the nodes in the cluster.

Command Mode

Config mode

Example

The following example sets the cluster ID for the Violin Array.

```
(config) # cluster id 15000-0000-0000
```

cluster master

The **cluster master** command configures interface, address, and auto-discovery settings for the master node in a cluster.

Syntax

```
[no] cluster master address vip <ip-address> <netmask>
[no] cluster master auto-discovery
[no] cluster master interface <ifname>
```

where:

`vip <ip-address> <netmask>`	Sets the cluster master virtual IP address and netmask. The virtual IP address is installed by the cluster master, and all other cluster nodes ensure they do not install the virtual IP address.
`auto-discovery`	Enables or disables auto-discovery of the cluster master. Auto-discovery is enabled by default.
`interface <ifname>`	Sets the interface to which the master virtual address is assigned.

Command Mode

Config mode

Example

The following example sets the cluster master virtual IP address and netmask.

```
(config) # cluster master address vip 10.10.10.1 /24
```

cluster name

The **cluster name** command sets the name of the cluster. The cluster name has an equivalent function to a hostname.

Syntax

```
cluster name <name>
no cluster name
```

The **no** form of the command removes the cluster name from the configuration.

Command Mode

Config mode

Example

The following example sets the name of the cluster for which this node is the master.

```
(config) # cluster name vmemcluster1
```

cluster port

The **cluster port** command sets the service port for the cluster.

Syntax

```
cluster port <number>
no cluster port
```

The **no** form of the command resets the service port for the cluster to the default of 60102.

Command Mode

Config mode

Example

The following example sets the service port for the cluster to 4310.

```
(config) # cluster port 4310
```

cluster suspend staged-upgrade

The **cluster suspend staged-upgrade** command suspends an ongoing staged VIMM upgrade. It takes a while for the upgrade to suspend. Use the "`show alarms`" command to monitor progress.

Syntax

```
cluster suspend staged-upgrade
```

Command Mode

Enable mode, Config mode

Example

The following example suspends a staged VIMM upgrade

```
(config) # cluster suspend staged-upgrade
```

cluster upgrade

The **cluster upgrade** command upgrades the system software on the nodes in a cluster. This command must be entered on the master node in the cluster.

Syntax

```
cluster upgrade <image-or-url> {immediate | staged} [force]
```

where:

`<image-or-url>`	Is a software image file on the local system, or a URL for a software image file on a remote system. If you specify a URL, it must be a standard URL with a protocol that `wget` supports, including HTTP, HTTPS, FTP, and TFTP. You can also use a pseudo-URL specifying an SCP file transfer. The URL path must be absolute. URL paths that are relative to a local home directory are not supported.
`immediate`	Simultaneously upgrades all of the Memory Gateways in the cluster, then restarts them. During the upgrade process, clients will be unable to maintain connections to exported LUNs until the cluster is restarted.
`staged`	Performs a non-disruptive or staged upgrade of the ACMs, VCMs followed by the VIMMs in the Violin Array. The command is rejected if certain NDU prerequisites are not met. If rejected, hints are provided to remedy the problem.
`force`	Proceeds with the upgrade even if one or more of the nodes in the cluster are not reachable from the cluster master.

Command Mode

Config mode

Example

The following example performs a non-disruptive upgrade using a software image on a remote server.

```
(config) # cluster upgrade http://www.remote-host/A7.5.0.1.img staged
```

cluster upgrade abort

The **cluster upgrade abort** command aborts an ACM upgrade when the "`cluster upgrade <image-or-url> immediate`" command has been issued.

This command is useful in certain situations, such as a mis-typed upgrade image file name or the wrong array is being upgraded.

Syntax

```
cluster upgrade abort
```

Command Mode

Config mode

Example

The following example aborts an ongoing ACM upgrade.

```
(config) # cluster upgrade abort
Aborted background ACM upgrade with pid xxxxx.
```

Note: In the above example, "`pid xxxxx`" represents the process ID of the background upgrade.

configuration copy

The **configuration copy** command copies a configuration file to a specified target file. The active configuration cannot be the target of a copy operation, but can be the source, in which case the original remains active.

Syntax

```
configuration copy <source-file> <destination-file>
```

Command Mode

Config mode

Example

The following example copies a configuration file.

```
(config) # configuration copy config1 config1.bak
```

configuration delete

The **configuration delete** command deletes a configuration file. The current active configuration file cannot be deleted.

Syntax

```
configuration delete <filename>
```

Command Mode

Config mode

Example

The following example deletes a configuration file.

```
(config) # configuration delete config1.bak
```

configuration fetch

The **configuration fetch** command retrieves a configuration file from a remote location or USB drive and saves it on the Violin Array.

Syntax

```
configuration fetch {<url> | usb <filename>} [<local-filename>]
```

where:

`<url>`	Is a URL for a configuration file on a remote system. This must be a standard URL with a protocol that `wget` supports, including HTTP, HTTPS, FTP, and TFTP. You can also use a pseudo-URL specifying an SCP file transfer. The URL path must be absolute. URL paths that are relative to a local home directory are not supported.
`usb <filename>`	Is a configuration file on a USB drive connected to the Violin Array.
`<local-filename>`	Saves the configuration with the specified filename. If a `<local-filename>` is not specified, the file is saved with the same name it had on the source location.

Command Mode

Config mode

Example

The following example downloads a configuration file from a remote location.

```
(config) # configuration fetch scp://username@hostname/path/configfile
```

configuration jump-start

The **configuration jump-start** command runs the initial-configuration wizard. The initial-configuration wizard is automatically invoked whenever the CLI is launched when the active configuration file is fresh (that is, not modified from its initial contents). This command invokes the wizard on demand.

Syntax

```
configuration jump-start
```

Command Mode

Config mode

Example

The following example runs the initial-configuration wizard.

```
(config) # configuration jump-start
```

configuration jump-start-file

The **configuration jump-start-file** command allows you to manually configure the array from a file on a USB drive connected to the array.

Syntax

```
configuration jump-start-file <filename>
```

The `<filename>` may be /var/opt/violin/auto_config/file.cfg or usb://file.cfg.

Command Mode

Config mode

Example

The following example shows issuing the command and then proceeding with auto-configuration.

```
(config) # configuration jump-start-file usb://file.cfg
This system is already configured.
This action will wipe out existing network configuration.
Reconfiguring IPs may result in loss of network management access.

Type 'YES' to confirm proceeding with auto-configuration for this system: YES
(config) #
```

If callhome is set up, updates are sent on the progress of auto-configuration.

configuration merge

The **configuration merge** command merges the common settings from a specified configuration file into the running configuration. The merge operation does not modify any configuration files.

Syntax

```
configuration merge <filename>
```

The `<filename>` is the configuration file whose settings are merged into the running configuration.

Command Mode

Config mode

Example

The following example merges the settings in a configuration file into the running configuration.

```
(config) # configuration merge config1
```

configuration move

The **configuration move** command moves a configuration file to a specified target file. The active configuration cannot be the target of a move operation.

Syntax

```
configuration move <source-file> <destination-file>
```

Command Mode

Config mode

Example

The following example moves a configuration file.

```
(config) # configuration move config1 config1.old
```

configuration new

The **configuration new** command creates a new configuration file. The new file can be pre-configured with your existing licenses and SSH host keys, as well as network connectivity settings.

Syntax

```
configuration new <filename> [factory [keep-basic] [keep-connect]]
```

where:

`<filename>`	Is the name of the configuration file to be created. If you specify no other options, the new configuration file will contain factory defaults and your existing licenses and host keys.
`factory`	Creates a new configuration file with only factory defaults.
`keep-basic`	Creates a new configuration file with factory defaults and licenses and SSH host keys.
`keep-connect`	Creates a new configuration file with factory defaults and settings for network connectivity (interfaces, routes, and ARP entries).

Command Mode

Config mode

Example

The following example creates a new configuration file that contains factory defaults, plus your existing licenses and SSH host keys.

```
(config) # configuration new newconfig
```

configuration revert

The **configuration revert** command reverts the configuration to a previous state, either to the factory default, or to the previously saved version.

Syntax

```
configuration revert saved
configuration revert factory [keep-basic] [keep-connect]
```

where:

`saved`	Reverts the running configuration to the most recently saved version of the active configuration file.
`factory`	Reverts both the running and saved configurations to factory defaults.
`keep-basic`	Reverts both the running and saved configurations to factory defaults, with your existing licenses and SSH host keys.
`keep-connect`	Reverts both the running and saved configurations to factory defaults, with your existing settings for network connectivity (interfaces, routes, and ARP entries).

Command Mode

Config mode

Examples

The following example reverts the running configuration to the most recently saved version of the active configuration file.

```
(config) # configuration revert saved
```

The following example reverts both the running and saved configurations to factory defaults.

```
(config) # configuration revert factory
```

configuration switch-to

The **configuration switch-to** command loads a configuration from a specified file and changes it to be the active configuration file. Note that the current running configuration is lost, and not automatically saved to the previous active configuration file.

Syntax

```
configuration switch-to <filename> [keep-basic]
```

where:

`<filename>`	Is the file to make the active configuration file.
`keep-basic`	Keeps the existing licenses and cluster ID when switching to the new configuration.

Command Mode

Config mode

Example

The following example loads a configuration from a file and makes it the active configuration.

```
(config) # configuration switch-to newconfig
```

configuration text fetch

The **configuration text fetch** command downloads a text configuration file (list of CLI commands) from a remote location.

Syntax

```
configuration text fetch <url> [apply] [filename <local-filename>]
[discard] [fail-continue] [verbose]
```

where:

`<url>`	Is a URL for the configuration file to download. This must be a standard URL with a protocol that `wget` supports, including HTTP, HTTPS, FTP, and TFTP. You can also use a pseudo-URL specifying an SCP file transfer. The URL path must be absolute. URL paths that are relative to a local home directory are not supported.
`apply`	Executes the commands in the downloaded file.
`<local-filename>`	Saves the configuration file with the specified filename. If a `<local-filename>` is not specified, the file is saved with the same name it had on the source location.
`discard`	Deletes the downloaded file after applying the commands in it.
`fail-continue`	Continues applying commands even if one fails.
`verbose`	Displays the commands and output from the commands as they are executed.

Command Mode

Config mode

Example

The following example downloads a configuration file from a remote location and executes the commands in the file.

```
(config) # configuration text fetch scp://username@hostname/configfile.txt apply
```

configuration text file

The **configuration text file** command allows you to manage configuration text files on the Violin Array.

Syntax

```
configuration text file <filename> apply [fail-continue] [verbose]
configuration text file <filename> delete
configuration text file <filename> rename <destination-filename>
configuration text file <filename> upload <destination-url>
```

where:

`<filename>`	Is the name of a configuration text file stored on the Violin Array.
`apply`	Executes the commands in the file.
`fail-continue`	Continues executing commands even if one fails.
`verbose`	Displays the commands and output from the commands as they are executed.
`delete`	Deletes the specified configuration text file from the system.
`rename`	Renames a configuration text file to `<destination-filename>`.
`<destination-url>`	Specifies the URL where the configuration text file should be uploaded. You can specify an FTP, TFTP, SCP, or SFTP URL.

Command Mode

Config mode

Example

The following example executes the commands in a configuration text file, then deletes the file from the system.

```
(config) # configuration text file configfile.txt apply
(config) # configuration text file configfile.txt delete
```

configuration text generate

The **configuration text generate** command creates a configuration text file from the current active configuration or a saved configuration.

Syntax

```
configuration text generate active {running | saved} [save <destination-filename> | upload <destination-url>]
configuration text generate file <source-filename> [save <destination-filename> | upload <destination-url>]
```

where:

`active`	Generates a configuration text file from the current active configuration.
`running`	Generates the configuration text file from the running configuration file.
`saved`	Generates the configuration text file from the most recently saved configuration file.
`file <source-filename>`	Generates a configuration text file from an inactive configuration file stored on the Violin Array.
`save <destination-filename>`	Saves the configuration text file locally with the specified filename.
`upload <destination-url>`	Specifies the URL where the configuration text file should be uploaded. You can specify an FTP, TFTP, SCP, or SFTP URL.

Command Mode

Config mode

Example

The following example generates a configuration text file from the running configuration and saves it to a local file.

```
(config) # configuration text generate active running save config1.txt
```

configuration upload

The **configuration upload** command sends the active configuration or a specified configuration file to a remote location.

Syntax

```
configuration upload {<filename> | active} <destination-url>]
```

where:

`<filename>`	Is a configuration file stored on the Violin Array.
`active`	Indicates the active configuration should be uploaded.
`<destination-url>`	Specifies the URL where the configuration should be uploaded. You can specify an FTP, TFTP, SCP, or SFTP URL.

Command Mode

Config mode

Example

The following example uploads the active configuration to a remote location.

```
(config) # configuration upload active scp://username@hostname/configfile.txt
```

configuration write

The **configuration write** command saves the running configuration to the active configuration file. This command is functionally similar to the **write memory** command.

Syntax

```
configuration write [local]
configuration write to {<filename> [no-switch] | usb <filename>}
```

where:

`local`	Saves the configuration on the local node only, rather than saving it on all of the nodes in the cluster
`<filename>`	Saves the configuration to the specified `<filename>`. The saved configuration then becomes the active configuration.
`no-switch`	Saves the configuration but leaves the current configuration active.
`usb <filename>`	Saves the configuration to a USB drive connected to the Violin Array.

Command Mode

Config mode

Example

The following example saves the running configuration to a file and makes it the active configuration.

```
(config) # configuration write to config1.txt
```

configure terminal

The **configure terminal** command enters Config mode on the Violin Array. You must have admin privileges to enter Config mode.

Syntax

```
configure terminal
```

Command Mode

Enable mode

Example

The following example enters Config mode.

```
# conf t
(config) #
```

debug generate dump

The **debug generate dump** creates a debug dump (sysdump) file on the Violin Array. You can manage the files generated with this command using the **file debug-dump** command.

Syntax

```
debug generate dump
```

Command Mode

Enable mode, Config mode

Example

The following example generates a debug dump file, uploads it to a URL using SCP, then deletes it from the system.

```
# debug generate dump
Generated dump sysdump-vmg01-20130125-201123.tgz

# file debug-dump upload sysdump-vmg01-20130125-201123.tgz scp://
username@hostname/path/sysdump.tgz

# file debug-dump delete sysdump-vmg01-20130125-201123.tgz
```

disable

The **disable** command exits Enable mode in the Violin Array CLI and returns to Standard mode.

Syntax

```
disable
```

Command Mode

Enable mode

Example

The following example exits Enable mode and returns to Standard mode.

```
# disable
>
```

email auth

The **email auth** command enables SMTP authentication for the Violin array and sets the username and password to be used for SMTP authentication.

Syntax

```
[no] email auth enable
[no] email auth password [<password>]
[no] email auth username <username>
```

where:

`enable`	Enables SMTP authentication when the Violin Array sends e-mails. The **no** form of the command disables authentication for sending e-mails.
`password [<password>]`	Sets the password used with SMTP authentication. If you do not enter the password in the command, the CLI prompts you to enter one. The **no** form of the command clears the password.
`username <username>`	Sets the username used with SMTP authentication. The **no** form of the command clears the username.

Command Mode

Config mode

Examples

The following example enables SMTP authentication and sets the username and password.

```
(config) # email auth enable
(config) # email auth username violin
(config) # email auth password
Password: *****
```

email callhome auth

The **email callhome auth** command enables SMTP authentication for callhome e-mails sent by the Violin Array, and sets the username and password to be used for SMTP authentication.

Syntax

```
[no] email callhome auth enable
[no] email callhome auth password [<password>]
[no] email callhome auth username <username>
```

where:

`enable`	Enables SMTP authentication when the Violin Array sends callhome e-mails. The **no** form of the command disables authentication for sending callhome e-mails.
`password [<password>]`	Sets the password used with SMTP authentication. If you do not enter the password in the command, the CLI prompts you to enter one. The **no** form of the command clears the password.
`username <username>`	Sets the username used with SMTP authentication. The **no** form of the command clears the username.

Command Mode

Config mode

Examples

The following example enables SMTP authentication for callhome e-mails and sets the username and password.

```
(config) # email callhome auth enable
(config) # email callhome auth username violin
(config) # email callhome auth password
Password: *****
```

email callhome

The **email callhome** command configures the Violin Array to generate and send automatic support notifications over e-mail (callhome e-mails) to Violin Technical Support, as well as which events generate the callhome e-mails.

Syntax

```
[no] email callhome enable
[no] email callhome event <event>
[no] email callhome mailhub <hostname-or-ip-address>
[no] email callhome recipient <email-address>
```

where:

`enable`	Enables the Violin Array to send callhome e-mails. The **no** form of the command disables sending callhome e-mails.
`event <event>`	Specifies an event that generates a callhome email. Enter **email callhome event ?** to list the possible events. The **no** form of the command disables sending callhome e-mails for the event.
`mailhub <hostname-or-ip-address>`	Specifies the hostname or IP address of the mail relay to be used to send callhome e-mails.
`recipient <email-address>`	Is the e-mail address to which the system will send callhome messages.

Command Mode

Config mode

Example

The following example enables callhome e-mails and specifies an event that will generate an callhome e-mail.

```
(config) # email callhome enable
(config) # email callhome event kernel-crash
```

email consolidate

The **email consolidate** command enables and configures settings for e-mail alert consolidation. E-mail alert consolidation allows you to reduce the number of alert e-mails sent by the Violin Array by combining a specified number of alerts received over a specified number of seconds into a single e-mail.

Syntax

```
[no] email consolidate
email consolidate events <number-of-events>
email consolidate period <seconds>
```

where:

`<number-of-events>`	Sets the maximum number of alerts that can be consolidated into one e-mail. The default is 5 events.
`<seconds>`	Sets the time period over which the e-mail alerts are accumulated. The system collects alerts (up to the `<number-of-events>` setting) for this number of seconds, then sends a single e-mail alert containing the accumulated alerts.

If you enter the **email consolidate** command without options, it enables the e-mail alert consolidation feature. The e-mail alert consolidation feature is disabled by default. The **no** form of the command disables the feature.

Command Mode

Config mode

Example

The following example enables the e-mail alert consolidation feature, and sets the number of events and alert accumulation period. In the example, the system accumulates up to 10 alerts over a 20-second period, then sends the accumulated alerts as a single e-mail.

```
(config) # email consolidate
(config) # email consolidate events 10
(config) # email consolidate period 20
```

email dead-letter

The **email dead-letter** command sets how the system handles e-mails that cannot be sent; for example, due to a failed mailhub. You can configure whether to keep the unsent e-mails, and how long to keep them before they are deleted.

Syntax

```
[no] email dead-letter enable
email dead-letter cleanup max-age <duration>
no email dead-letter cleanup max-age
```

where:

`enable`	Enables the Violin Array to save e-mails that could not be sent. The **no** form of the command disables this feature. By default, the feature is disabled.
`<duration>`	Specifies how long the unsent e-mails are kept on the system, in days, hours, minutes, seconds. The **no** form of the command disables deleting unsent mails based on age.

Command Mode

Config mode

Example

The following example enables the Violin Array to save e-mails that cannot be sent. The e-mails are kept for 1 day and 12 hours.

```
(config) # email dead-letter enable
(config) # email dead-letter cleanup max-age 1d12h0m0s
```

email domain

The **email domain** command sets the domain name to be used as the source for e-mail notifications. The specified domain name is used in conjunction with the system hostname to form the source email address.

The rules are as follows:

- If an email domain is specified using this command, it is always used. If the hostname has any dots in it, everything to the right of the first dot is stripped, and the e-mail domain is appended.
- Otherwise, if the hostname has dots in it, it is used as is.
- Otherwise, the currently-active system domain name is used. This can come either from the resolver configuration, or from state dynamically instantiated by DHCP.

Syntax

```
[no] email domain <hostname-or-ip-address>
```

The **no** form of the command removes the specified domain name, so that the currently-active system domain name is used instead.

Command Mode

Config mode

Example

The following example configures the domain name to be used as the source for e-mail notifications from the Violin Array.

```
(config) # email domain vmem
```

email mailhub

The **email mailhub** command specifies the hostname or IP address of the mail relay to be used to send notification e-mails.

Syntax

```
[no] email mailhub <hostname-or-ip-address>
```

The **no** form of the command clears the mailhub setting from the configuration.

Command Mode

Config mode

Example

The following example specifies the IP address of the mail relay to be used to send notification e-mails from the Violin Array.

```
(config) # email mailhub 10.10.10.10
```

email mailhub-port

The **email mailhub-port** command specifies the port to use with the mail relay for sending notification e-mails.

Syntax

```
[no] email mailhub-port <port>
```

The **no** form of the command resets the mail port to the default of TCP port 25.

Command Mode

Config mode

Example

The following example specifies the port to use with the mail relay for sending notification e-mails.

```
(config) # email mailhub-port 587
```

email notify event

The **email notify event** command enables e-mail notifications for specified events.

Syntax

```
[no] email notify event <event>
```

where `<event>` specifies an event that will generate a notification e-mail. Enter the command **email notify event ?** to list the possible events. The **no** form of the command disables sending notification e-mails for the event.

Command Mode

Config mode

Example

The following example specifies an event that will generate a notification e-mail.

```
(config) # email notify event kernel-crash
```

Example

The following example specifies an event that will not generate a notification e-mail.

```
(config) # no email notify event unexpected-brownout
```

email notify recipient

The **email notify recipient** command specifies e-mail addresses to receive notification e-mails.

Syntax

```
[no] email notify recipient <email-address> class {failure | info}
[no] email notify recipient <email-address> detail
```

where:

`<email-address>`	Is the e-mail address to which the system will send notification messages.
`failure`	Send notifications to the e-mail address when events classified as failure-type events occur.
`info`	Send notifications to the e-mail address when events classified as info-type events occur.
`detail`	Specifies that the recipient receive detailed (as opposed to summarized) e-mail notifications.

The **no** form of the command disables the setting for the e-mail address.

Command Mode

Config mode

Example

The following example configures the system to send e-mail notifications to a recipient when failure-type events occur.

```
(config) # email notify recipient admin@vmem.com class failure
```

email return-addr

The **email return-addr** command specifies the return address for the e-mails sent from the Violin Array.

Syntax

```
[no] email return-addr <string>
```

If the `<string>` contains the `@` character, it is used as-is for the return address. If it does not, then `@<hostname>.<domain>` is appended to complete the address. The **no** form of the command resets the return address to the default of `do-not-reply`.

Command Mode

Config mode

Example

The following example specifies the return address for the e-mails sent from the Violin Array.

```
(config) # email return-addr email-notifications@VMEM01.vmem.com
```

email return-host

The **email return-host** command specifies whether to include the system hostname in the return address for the e-mails sent from the Violin Array. This setting applies only if the full return address is not specified with the **email return-addr** command.

Syntax

```
[no] email return-host
```

The **no** form of the command configures the system not to include the hostname in the return e-mail address.

Command Mode

Config mode

Example

The following example configures the system to include the hostname in the return e-mail address from the Violin Array.

```
(config) # email return-host
```

email send-test

The **email send-test** command sends a test e-mail to all of the configured notification e-mail recipients. This is useful to make sure the configuration works without having to wait for an event to occur.

Syntax

```
email send-test
```

Command Mode

Enable mode, Config mode

Example

The following example sends a test e-mail to the configured notification e-mail recipients.

```
(config) # email send-test
```

email ssl min-version

The **email ssl min-version** command sets the minimum version of TLS protocol used by the email server either to version TLS 1 or TLS 1.2. The default TLS protocol is TLS 1.2 and it can be changed if your environment does not support TLS 1.2.

Syntax

```
email ssl min-version <tls1 | tls1.2>
```

Command Mode

Config mode

Example

The following example sets the minimum TLS protocol to version TLS 1.2.

```
(config) # email ssl min-version tls1.2
```

email upload

The **email upload** command configures the Violin array to generate and send an upload bundle to Violin technical support when callhome and log upload is enabled for an event.

Syntax

```
[no] email upload enable
[no] email upload event <event>
```

where:

`enable`	Enables the Violin Array to send an upload bundle when callhome and log upload is enabled for an event.
`event <event>`	Specifies an event that generates a log upload. Enter **email upload event ?** to list the possible events. The **no** form of the command disables sending a log upload bundle for the event.

Command Mode

Config mode

Example

The following example enables log upload and specifies an event that will generate an upload.

```
(config) # email upload enable
(config) # email upload event kernel-crash
```

enable

The **enable** command exits Standard mode in the Violin Array CLI and enters Enable mode.

Syntax

```
enable
```

Command Mode

Standard mode

Example

The following example exits Standard mode and enters Enable mode.

```
> enable
#
```

eventlog enable

The **eventlog enable** command enables recording of system events, such as administrative logins and alarms, to the event log. You can view information about the events in the event log with the **show eventlog** command.

Syntax

```
[no] eventlog enable
```

Command Mode

Config mode

Example

The following example enables recording of system events to the event log.

```
(config) # eventlog enable
```

exit

The **exit** command exits the current command mode on the Violin Array.

Syntax

```
exit
```

When entered in Config mode, the **exit** (as well as the **no configure** command) command returns to Enable mode. When entered in Standard or Enable mode, the **exit** command logs you out of the CLI. To move from Enable mode to Standard mode, use the **disable** command.

Command Mode

Standard mode, Enable mode, Config mode

Example

The following example exits Config mode and returns to Enable mode.

```
(config) # exit
#
```

file debug-dump

The **file debug-dump** command allows you to manage debug dump (sysdump) files on the Violin Array.

Syntax

```
file debug-dump {delete | email | upload-usb} <filename>
file debug-dump upload <filename> <destination-url>
```

where:

`delete`	Deletes the specified debug dump file from the system.
`email`	E-mails the specified debug dump file to the list of e-mail addresses configured to receive informational events, set with the **email notify recipient** command.
`upload-usb`	Copies the specified debug dump file to a USB drive connected to the Violin Array.
`<filename>`	Is the name of a debug dump file generated with the **debug generate dump file** command.
`<destination-url>`	Specifies the URL where the debug dump file should be uploaded. You can specify an FTP, TFTP, SCP, or SFTP URL.

Command Mode

Enable mode, Config mode

Example

The following example generates a debug dump file, uploads it to a URL using SCP, then deletes it from the system.

```
# debug generate dump
Generated dump sysdump-vmg01-20130125-201123.tgz

# file debug-dump upload sysdump-vmg01-20130125-201123.tgz scp://
username@hostname/path/sysdump.tgz

# file debug-dump delete sysdump-vmg01-20130125-201123.tgz
```

file stats

The **file stats** command allows you to manage statistics report files on the Violin Array.

Syntax

```
file stats delete <filename>
file stats move <source-filename> to <destination-filename>
file stats upload <filename> <destination-url>
```

where:

`delete`	Deletes the specified statistics report file from the system.
`<filename>`	Is the name of a statistics report file generated with the **stats export** command.
`move`	Renames a statistics report file from `<source-filename>` to `<destination-filename>`.
`<destination-url>`	Specifies the URL where the statistics report file should be uploaded. You can specify an FTP, TFTP, SCP, or SFTP URL.

Command Mode

Enable mode, Config mode

Example

The following example generates a statistics report file, uploads it to a URL using SCP, then deletes it from the system.

```
# stats export csv memory filename memstats.csv
Generated report file: memstats.csv

# file stats upload memstats.csv scp://username@hostname/path/memstats.csv

# file stats delete memstats.csv
```

file tcpdump

The **file tcpdump** command allows you to manage TCP dump files on the Violin Array.

Syntax

```
file tcpdump delete <filename>
file tcpdump upload <filename> <destination-url>
```

where:

`delete`	Deletes the specified TCP dump file from the system.
`<filename>`	Is the name of a TCP dump file generated with the **tcpdump** command.
`<destination-url>`	Specifies the URL where the TCP dump file should be uploaded. You can specify an FTP, TFTP, SCP, or SFTP URL.

Command Mode

Enable mode, Config mode

Example

The following example generates a TCP dump file, uploads it to a URL using SCP, then deletes it from the system.

```
# tcpdump -c 4 -w tcpdump.txt
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
4 packets captured
4 packets received by filter
0 packets dropped by kernel

# file tcpdump upload tcpdump.txt scp://username@hostname/path/tcpdump.txt

# file tcpdump delete tcpdump.txt
```

file var

The **file var** command allows you to set thresholds for available space in the `/var` filesystem on the Violin Array, as well as remove dump files, snapshots, and log files when necessary to clear space in the `/var` filesystem. To show the available space in the `/var` filesystem, use the **show files var** command.

Syntax

```
file var cleanup
file var critical-threshold <percent>
file var warning-threshold <percent>
```

where:

`cleanup`	Removes the dump files, snapshots, and log files from the `/var` filesystem on the Violin Array.
`critical-threshold <percent>`	Specifies a percentage of space in the `/var` filesystem that when surpassed causes the system to automatically free space by deleting dump files, snapshots, and log files.
`warning-threshold <percent>`	Specifies a percentage of the space in the `/var` filesystem that when surpassed causes the system to issue a warning message.

Command Mode

Enable mode, Config mode

Examples

The following example removes the dump files, snapshots, and log files from the `/var` filesystem on the Violin Array.

```
(config) # file var cleanup
```

The following example causes a warning message to be issued when the amount of used space in the `/var` filesystem exceeds 75 percent.

```
(config) # file var warning-threshold 75
```

The following example causes dump files, snapshots, and log files to be deleted automatically when the amount of used space in the `/var` filesystem exceeds 95 percent.

```
(config) # file var critical-threshold 95
```

ftp-server enable

The **ftp-server enable** command enables or disables the FTP server on the Violin Array. When the FTP server is enabled, you can transfer files to and from the Violin Array using an FTP client program.

Syntax

```
[no] ftp-server enable
```

The **no** form of the command disables the FTP server. By default, the FTP server is disabled.

Command Mode

Config mode

Example

The following example enables the FTP server.

```
(config) # ftp-server enable
```

ftp-server ssl enable

The **ftp-server ssl enable** command enables the SSL protocol for FTP server. The SSL protocol is disabled by default.

Syntax

```
[no] ftp-server ssl enable
```

The **no** form of the command disables the SSL protocol for the FTP server.

Command Mode

Config mode

Example

The following example enables the SSL for FTP server.

```
(config) # ftp-server ssl enable
```

ftp-server ssl min-version

The **ftp-server ssl min-version** command sets the minimum version of TLS protocol used by the FTP server either to version TLS 1 or TLS 1.2. The default TLS protocol is TLS 1.2 and it can be changed if your environment does not support TLS 1.2.

Syntax

```
ftp-server ssl min-version <tls1 | tls1.2>
```

Command Mode

Config mode

Example

The following example sets the minimum TLS protocol to version TLS 1.2.

```
(config) # ftp-server ssl min-version tls1.2
```

help

The **help** command displays basic information about how to get information about commands in the Violin CLI.

Syntax

```
help
```

Command Mode

Standard mode, Enable mode, Config mode

Example

The following example displays the help text for the Violin CLI.

```
# help
You may request context-sensitive help at any time by pressing '?'
on the command line.  This will show a list of choices for the
word you are on, or a list of top-level commands if you have not
typed anything yet.

If "<cr>" is shown, that means that what you have entered so far
is a complete command, and you may press Enter (carriage return)
to execute it.

Try the following to get started:
  ?
  show ?
  show c?
  show clock?
  show clock ?
  show interfaces ?     (from enable mode)
```

hostname

The **hostname** command sets the system hostname for the Violin Array.

Syntax

```
hostname <hostname>
no hostname
```

The **no** form of the command removes the hostname from the system.

Command Mode

Config mode

Example

The following example sets the system hostname.

```
(config) # hostname VMEM01
```

interface alias

The **interface alias** command sets an alias on a specified Ethernet interface on the Violin Array.

Syntax

```
interface <ifname> alias <index-number> ip address <ip-address> <netmask>
no interface <ifname> alias <index-number>
```

The **no** form of the command removes the alias from the configuration.

Command Mode

Config mode

Example

The following command adds an alias address for interface eth2:

```
(config) # interface eth2 alias 1 ip address 10.10.10.10 /32
```

interface comment

The **interface comment** command adds a comment to the configuration for an Ethernet interface on the Violin Array.

Syntax

```
interface <ifname> comment <comment>
[no] interface <ifname> comment
```

The **no** form of the command removes the comment from the configuration.

Command Mode

Config mode

Example

The following command configures a comment for interface eth2:

```
(config) # interface eth2 comment this_is_a_comment
```

interface dhcp

The **interface dhcp** command enables DHCP for an Ethernet interface on the Violin Array.

Syntax

```
[no] interface <ifname> dhcp [renew]
```

The **no** form of the command disables DHCP for the interface. The **renew** option renews the DHCP-assigned address for the interface.

Command Mode

Config mode

Example

The following command enables DHCP for interface eth2:

```
(config) # interface eth2 dhcp
```

interface duplex

The **interface duplex** command configures the duplex setting for an Ethernet interface.

Syntax

```
interface <ifname> duplex half | full | auto
```

The **no** form of the command resets the duplex setting for the interface to the default of auto.

Command Mode

Config mode

Example

The following command sets the duplex for interface eth2 to half.

```
(config) # interface eth2 duplex half
```

interface ip address

The **interface ip address** command sets the IP address and netmask for an Ethernet interface. Use this command to set the IP address for an ACM.

Syntax

```
interface <ifname> ip address <ip-address> <netmask>
no interface <ifname> ip address
```

The **no** form of the command removes the IP address from the interface configuration.

Command Mode

Config mode

Example

The following command sets an IP address and netmask for interface eth2.

```
(config) # interface eth2 ip address 12.120.130.1 /24
```

interface shutdown

The **interface shutdown** command disables an Ethernet interface on the Violin Array.

Syntax

```
[no] interface <ifname> shutdown
```

The **no** form of the command re-enables the interface.

Command Mode

Config mode

Example

The following command disables interface eth2.

```
(config) # interface eth2 shutdown
```

interface speed

The **interface speed** command sets the speed for an Ethernet interface on the Violin Array.

Syntax

```
[no] interface <ifname> speed <speed-in-Mbit/sec> | auto
```

The **no** form of the command resets the interface speed to the default of auto (automatically detect speed).

Command Mode

Config mode

Example

The following command sets the speed of interface eth2 to 1000 Mbit/sec.

```
(config) # interface eth2 speed 1000
```

interface zeroconf

The **interface zeroconf** command enables zero configuration networking for an Ethernet interface.

Syntax

```
[no] interface <ifname> zeroconf
```

The **no** form of the command disables zero configuration networking the interface.

Command Mode

Config mode

Example

The following command enables zero configuration networking for interface eth2.

```
(config) # interface eth2 zeroconf
```

ip default-gateway

The **ip default-gateway** command sets the default route for the Violin Array.

Syntax

```
ip default-gateway <ip-address> [<ifname>]
no ip default-gateway
```

Specifying an `<ifname>` causes the default route to apply to traffic on that interface. The **no** form of the command removes the default route from the configuration.

Command Mode

Config mode

Example

The following command configures a default route for the Violin Array.

```
(config) # ip default-gateway 10.10.10.10
```

ip dhcp

The **ip dhcp** command configures how the DHCP client on the Violin Array interacts with a DHCP server on the network.

Syntax

```
[no] ip dhcp [default-gateway yield-to-static] [send-hostname] [hostname
<hostname>] [primary-intf <ifname>]
```

where:

`default-gateway yield-to-static`	Causes the Violin Array to ignore the default gateway assignment from the DHCP server if there is already a default gateway configured with the **ip default gateway** command.
`send-hostname`	Enables the Violin Array to supply a hostname to the DHCP server during negotiation.
`<hostname>`	Specifies the hostname to be supplied to the DHCP server when the **send-hostname** option is configured.
`primary-intf <ifname>`	Specifies the interface on the Violin Array that will accept non-interface-specific configuration via DHCP.

Command Mode

Config mode

Example

The following command configures the Violin Array to supply the hostname `VMG01` during DCHP negotiation with a DHCP server.

```
(config) # ip dhcp send-hostname
(config) # ip dhcp hostname VMG01
```

ip domain-list

The **ip domain-list** command adds a domain name to the list of domains that the Violin Array uses when resolving hostnames.

Syntax

```
[no] ip domain-list <domain-name>
```

The **no** form of the command removes a domain from the list.

Command Mode

Config mode

Example

The following command adds vmem.com to the list of domains used for resolving hostnames.

```
(config) # ip domain-list vmem.com
```

ip host

The **ip host** command configures static mappings between hosts and IPv4 addresses.

Syntax

```
[no] ip host <hostname> <ip-address>
```

The **no** form of the command removes an entry from the list of statically mapped hosts.

Command Mode

Config mode

Example

The following command maps the host vmem01 to address 10.10.10.10.

```
(config) # ip host vmem01 10.10.10.10
```

ip map-hostname

The **ip map-hostname** command ensures static host mapping for the current hostname.

Syntax

```
[no] ip map-hostname
```

The **no** form of the command configures the system not to ensure static host mapping for the current hostname.

Command Mode

Config mode

Example

The following command ensures static host mapping for the current hostname.

```
(config) # ip map-hostname
```

ip name-server

The **ip name-server** command configures the address of a name server to be used by the Violin Array.

Syntax

```
[no] ip name-server <ip-address>
```

The **no** form of the command removes the name server from the configuration.

Command Mode

Config mode

Example

The following command adds 10.10.10.10 to the list of name servers used by the system.

```
(config) # ip name-server 10.10.10.10
```

ip route

The **ip route** command configures a static route on the Violin Array.

Syntax

```
ip route <network prefix> <netmask> <next hop IP address or interface>
no ip route <network prefix> <netmask> [<destination>]
```

The **no** form of the command removes the static route from the configuration.

Command Mode

Config mode

Examples

The following command configures a static route on the Violin Array.

```
(config) # ip route 10.10.10.0 /24 10.10.10.1
```

job command

The **job command** command configures CLI commands in a job. Jobs are sequences of Violin CLI commands that can be executed according to a schedule, similar to a `cron` operation in a UNIX-like environment.

Syntax

```
job <job-id> command <sequence-number> <command-string>
no job <job-id> command <sequence-number>
```

where:

`<job-id>`	Specifies an identifier for the job. If a job with the specified `<job-id>` does not already exist, it is created by the system.
`<sequence-number>`	Sets the order in which the command is executed relative to the other commands in the job. Commands are executed in ascending sequence order.
`<command-string>`	Is a Violin CLI command to be executed. If the command consists of more than one word, enclose the command string in quotes.

The **no** form of the command removes the command with the specified `<sequence-number>` from the job with the specified `<job-id>`.

Command Mode

Config mode

Example

The following example creates a job and adds two commands to it.

```
(config) # job 101 command 1 "show ver"
(config) # job 101 command 2 "show services small-servers"
```

job comment

The **job comment** command adds a comment string to a job. The comment string is visible in **show** command output related to the job.

Syntax

```
job <job-id> comment <comment-string>
no job <job-id> comment
```

where:

`<job-id>`	Specifies an identifier for the job. If a job with the specified `<job-id>` does not already exist, it is created by the system.
`<comment-string>`	Is a comment to be associated with the job If the comment consists of more than one word, enclose the comment string in quotes.

The **no** form of the command removes the comment from the job with the specified `<job-id>`.

Command Mode

Config mode

Example

The following example adds a comment to a job.

```
(config) # job 101 comment "Test Job"
```

job enable

The **job enable** command enables a job for execution. A job must be enabled before it can be executed. If the job has been scheduled for execution, this command enables the job and places it in the “pending” state.

Syntax

```
job <job-id> enable
no job <job-id> enable
```

where:

`<job-id>`	Specifies an identifier for the job. If a job with the specified `<job-id>` does not already exist, it is created by the system.

The **no** form of the command disables the job and prevents it from being executed.

Command Mode

Config mode

Example

The following example enables a job.

```
(config) # job 101 enable
```

job execute

The **job execute** command immediately executes a specified job. The job must be enabled before it can be executed. If the job has been scheduled for execution, the schedule for the job is not changed.

Syntax

```
job <job-id> execute
```

where:

`<job-id>`	Specifies an identifier for the job. If a job with the specified `<job-id>` does not already exist, it is created by the system.

Command Mode

Enable mode, Config mode

Example

The following example executes a job.

```
# job 101 execute
```

job fail-continue

The **job fail-continue** command configures a job to continue executing its CLI commands in the event that a command fails. By default, the job halts if a command fails.

Syntax

```
job <job-id> fail-continue
no job <job-id> fail-continue
```

where:

`<job-id>`	Specifies an identifier for the job. If a job with the specified `<job-id>` does not already exist, it is created by the system.

The **no** form of the command causes the job to halt when a command fails.

Command Mode

Config mode

Example

The following example configures a job to continue running when a command fails.

```
(config) # job 101 fail-continue
```

job name

The **job name** command assigns a name to a job. The job name is visible in **show** command output related to the job.

Syntax

```
job <job-id> name <name-string>
no job <job-id> name
```

where:

`<job-id>`	Specifies an identifier for the job. If a job with the specified `<job-id>` does not already exist, it is created by the system.
`<name-string>`	Is a name to be associated with the job. If the name consists of more than one word, enclose the name string in quotes.

The **no** form of the command removes the name from the job with the specified `<job-id>`.

Command Mode

Config mode

Example

The following example adds a comment to a job.

```
(config) # job 101 name "Sample Job"
```

job schedule

The **job schedule** command sets an execution schedule for a job. You can schedule a job to be executed once, daily, weekly, monthly, or periodically at fixed intervals.

Syntax

```
[no] job <job-id> schedule type <frequency>
[no] job <job-id> schedule daily time <hh>:<mm>:<ss>
[no] job <job-id> schedule monthly {day-of-month <day> | interval <months>
| time <hh>:<mm>:<ss>}
[no] job <job-id> schedule once time <hh>:<mm>:<ss> [date <yyyy>/<mm>/
<dd>]
[no] job <job-id> schedule periodic interval <time>
[no] job <job-id> schedule weekly {day-of-week <day> | time
<hh>:<mm>:<ss>}
```

The **no** form of the command deletes the execution schedule for the job.

Command Mode

Config mode

Examples

The following example schedules a job to run once a week on Fridays.

```
(config) # job 101 schedule weekly day-of-week fri
```

The following example schedules a job to run every three months.

```
(config) # job 101 schedule monthly interval 3
```

The following example schedules a job to run once at a specified date and time.

```
(config) # job 101 schedule once time 12:12:12 date 2013/02/12
```

license delete

The **license delete** command removes a license key for a specified licensed feature on the Violin Array.

Syntax

```
license delete <license-number>
```

The `<license-number>` refers to a license key. Enter **license delete ?** to list the license keys by number.

Command Mode

Config mode

Example

The following example deletes a license key.

```
(config) # license delete ?
<license number>
1                       VSHARE
2                       RESTRICTED_CMDS
(config) # license delete 2
```

license install

The **license install** command adds or removes keys for licensed features on the Violin Array.

Syntax

```
[no] license install <license-key>
```

The **no** form of the command removes an installed license key.

Command Mode

Config mode

Example

The following example installs a license key.

```
(config) # license install LK2-RESTRICTED_CMDS-7X87-SAMPLE-LICENSE-KEY
```

locate

The **locate** command turns on the ID LED on the front and rear of the Violin Array.

Syntax

```
locate
```

Command Mode

Enable mode, Config modes

Example

The following example turns on the ID LED.

```
# locate
```

logging

The **logging** command configures the Violin Array to send syslog messages to a remote syslog server.

Syntax

```
[no] logging <ip-address>
```

Where `<ip-address>` is the IP address of the remote syslog server. You must specify an IP address and not a hostname. The **no** form of the command disables sending of log messages to the specified syslog server.

Command Mode

Config mode

Example

The following example sends syslog messages to the server at 10.10.10.10.

```
(config) # logging 10.10.10.10
```

logging fields seconds

The **logging fields seconds** command adds and configures a “seconds” field to logging event messages, which indicates the time the event occurred in terms of the number of seconds (and optionally fractions of a second, up to six decimal places) since the Epoch began.

Syntax

```
[no] logging fields seconds enable
logging fields seconds fractional-digits {1 | 2 | 3 | 6}
logging fields seconds whole-digits {1 | 6 | all}
```

The **no** form of the command removes the seconds field from subsequent log entries. By default, the seconds field does not appear in log entries.

The **fractional-digits** parameter specifies how precise the seconds value is recorded, in number of digits to the right of the decimal point: 1, 2, 3, or 6, truncated from the right.

The **whole-digits** parameter specifies how many digits to the left of the decimal point will appear in the seconds field for each log entry, either 1 or 6, truncated from the left. The **all** keyword causes the full number of seconds to appear for each log message.

Command Mode

Config mode

Example

The following example enables the seconds field for log messages and records the time a logged event occurs in the number of seconds and fractions of a second since the Epoch began. The fractional value is recorded to six decimal places.

```
(config) # logging fields seconds enable
(config) # logging fields seconds fractional-digits 6
```

logging files delete

The **logging files delete** command allows you to delete log files accumulated on the system.

Syntax

```
logging files delete current
logging files delete oldest [<number-of-log-files>]
```

The **current** keyword deletes the currently active log file. The **oldest** parameter allows you to delete the either the single oldest log file, or the oldest `<number-of-log-files>` from the system.

Command Mode

Enable mode, Config mode

Example

The following example deletes the three oldest log files from the system.

```
(config) # logging files delete oldest 3
```

logging files rotation criteria

The **logging files rotation criteria** command specifies the conditions for automatically rotating the log files stored on the Violin Array.

Syntax

```
logging files rotation criteria frequency {daily | weekly | monthly}
logging files rotation criteria size <size in megabytes>
logging files rotation criteria size-pct <percentage>
```

where:

`frequency`	Rotates the active log file daily, weekly, or monthly.
`size <size in megabytes>`	Rotates the active log file when it reaches a threshold size. The size of the file is checked hourly, so if it passes the threshold in the middle of the hour, it will not be rotated until the end of the hour.
`size-pct <percentage>`	Rotates the active log file when its size reaches a specified percentage of the system `/var` partition size.

By default, a Violin Array rotates the log across five log files, with a 5 percent `size-pct` setting. If you change the setting, the Memory reverts to the default the next time it is booted.

Command Mode

Config mode

Example

The following example configures the system to automatically rotate the log file when the active log file reaches 100 megabytes.

```
(config) # logging files rotation criteria size 100
```

logging files rotation force

The **logging files rotation force** command forces an immediate rotation of the log files. Note that entering this command does not affect the schedule for automatically rotating the log files if the rotation criteria is set to **frequency**; the next automatic rotation will still occur at the scheduled time.

Syntax

```
logging files rotation force
```

Command Mode

Enable mode, Config mode

Example

The following example forces an immediate rotation of the log files.

```
(config) # logging files rotation force
```

logging files rotation max-num

The **logging files rotation max-num** command sets how many log files are kept on the system.

Syntax

```
logging files rotation max-num <number-of-files>
```

If the number of log files on the system exceeds the `<number-of-files>`, the system will delete as many as necessary to bring it down to this number, starting with the oldest log file.

Command Mode

Config mode

Example

The following example limits the number of log files on the system to 12.

```
(config) # logging files rotation max-num 12
```

logging files upload

The **logging files upload** command transfers a current or archived log file to a specified remote host.

Syntax

```
logging files upload {current | <log-file-number>} <destination-url>
```

where:

`current`	Uploads the messages in the currently active log file.
`<log-file-number>`	Specifies the number of an archived log file to upload. The archived log files are compressed in gzip format.
`<destination-url>`	Specifies the URL where the log file should be uploaded. You can specify an FTP, TFTP, SCP, or SFTP URL.

Command Mode

Enable mode, Config mode

Example

The following example uploads the current log file to a remote host using SCP.

```
(config) # logging files upload current scp://username@hostname/path/logfile.txt
```

logging files upload-auto

The **logging files upload-auto** command enables automatic uploading of log files to a remote host.

Note that log files are uploaded from all gateways simultaneously. Therefore, you should ensure that the host accepting the files can allow the required number of simultaneous connections. For example, you may want to increase the maximum open FTP connections parameter on the remote host.

Syntax

```
[no] logging files upload-auto <control parameters>
```

The `<control parameters>` are used to control upload interval and information gathered. Control parameters are:

`email-address <address>`	Uploads the log files to the specified e-mail address.
`email-split-size`	Set the maximum email split size for file upload in MBs.
`enable`	Enables automatic log gathering and uploading.
`include-cinfo`	Includes cache information with the uploaded logs.
`include-cores`	Includes core files with the uploaded logs.
`include-dump`	Includes system dump files with the uploaded logs.
include-mgs	Includes mg log files with the uploaded logs.
`interval <num> {days | hours}`	Sets how often the log files are uploaded.
`max-size`	Sets the maximum size for file uploads, in MB.
`protocol {email | ftp | ftp-epsv | http | https | scp}`	Specifies the protocol used for uploading the log file.
`remote-dir <directory>`	The directory on the remote host where the files will be uploaded.
`remote-site`	The name of the remote host; for example, support.vmem.com.
`upload-delay`	Set a delay after an event trigger before uploading files.
`upload-throttle`	Set a throttle to suppress uploading files.
`user <name> <password>`	The username and password used for uploading files to the remote host.

Command Mode

Config mode

Example

The following example configures automatic uploading of log files once every three days to a remote host using FTP.

```
(config) # logging files upload-auto enable
(config) # logging files upload-auto protocol ftp
(config) # logging files upload-auto remote-site support.vmem.com
(config) # logging files upload-auto user anonymous support@vmem.com
(config) # logging files upload-auto interval 3 days
```

logging files upload-auto immediate

The **logging files upload-auto immediate** command performs a one-time upload of logs to the configured destination site.

Syntax

```
logging files upload-auto immediate cancel
logging files upload-auto immediate [file {current | <log-file-number>}]
[local] [mode local | remote] [no-reports] [module type {acm | mg}]
[module id <module-id>]
```

where:

`cancel`	Cancels the current automatic log file upload, if one is active.
`file {current | <log-file-number>}`	Uploads the messages in the currently active log file or an archived log file.
`local`	Uploads the messages in the currently active log from the local system.
`mode local | remote`	Uploads the messages in the currently active log to the local system or to the remote host configured as the destination for automatic log file uploads.
`no-reports`	Omits reports generated by utilities such as **vincident -a** from the log file upload.
`module type {acm | mg}`	Uploads the messages in the currently active log file for either the ACMs or MGs on the Violin Array.
`module id <module-id>`	Uploads the messages in the currently active log file for a specified module.

Command Mode

Enable mode, Config mode

Examples

The following example uploads the current log file to the local system.

```
# logging files upload-auto immediate local file current mode local
```

The following example uploads an archived log file to a remote system.

```
(config) # logging files upload-auto remote-site support.vmem.com
(config) # logging files upload-auto immediate file 1 mode remote
```

logging format

The **logging format** command sets the format for log messages on the system, either standard or WebTrends Enhanced Log File Format (WELF).

Syntax

```
[no] logging format {standard | welf [fw-name <hostname>]}
```

The default format for log messages is **standard**; the **no** form of the command resets the log file format to standard.

For WELF, you can specify the firewall name associated with each log message. If no firewall name is set, the hostname is used by default.

Command Mode

Config mode

Example

The following example sets the format for log messages to WELF.

```
(config) # logging format welf
```

logging level audit mgmt

The **logging level audit mgmt** command specifies the severity of log messages that get placed in the audit log.

Syntax

```
[no] logging level audit mgmt <severity>
```

where `<severity>` is the severity level at which log messages get placed in the audit log.

Command Mode

Config mode

Example

The following example sets the severity for audit log messages to “warning”.

```
(config) # logging level audit mgmt warning
```

logging level cli commands

The **logging level cli commands** command sets the severity level at which CLI commands that the user executes are logged.

Syntax

```
[no] logging level cli commands <severity>
```

The default severity level for user-entered CLI commands is **notice** for the ACM CLI.

Command Mode

Config mode

Example

The following example sets the severity level in the system log for user-entered CLI commands to “notice”.

```
(config) # logging level cli commands notice
```

logging local

The **logging local** command sets the minimum severity of log messages to be saved in log files on local persistent storage.

Syntax

```
logging local <severity>
[no] logging local
```

The **no** form of the command disables local logging altogether.

Command Mode

Config mode

Example

The following example sets the minimum severity level for locally saved log messages to “alert”.

```
(config) # logging local alert
```

logging local override class

The **logging local override class** command sets or removes a per-class override on the logging level. All classes that do not have an override set use the global logging level set with the **logging local** command; any classes that do have an override will do as the override specifies.

In the context of this command, class is a synonym for syslog facility. It allows log messages to be divided up according to their origin. Syslog classes include mgmt-core (for mgmtd alone), mgmt-back (for other back-end components), and mgmt-front (for front-end components, utilities, and tests).

Syntax

```
[no] logging local override class
logging local override class <class> priority {<severity> | none}
[no] logging local override class <class>
```

Specifying **none** for the priority causes nothing from this class to be logged. The **no** form of the command disables the override for the class.

Command Mode

Config mode

Example

The following example configures events in the Kernel class to be logged in local storage with a priority of “info”.

```
(config) # logging local override class kern priority info
```

logging receive

The **logging receive** command allows this system to receive log messages from another host. When this command is enabled, only log messages matching or exceeding the minimum severity specified with the **logging local** command are logged, regardless of what is sent from the remote host.

Syntax

```
[no] logging receive
```

The **no** form of the command disables receiving of syslog messages from remote hosts. Reception of logging messages from remote hosts is disabled by default.

Command Mode

Config mode

Example

The following example enables reception of logging messages from remote hosts.

```
(config) # logging receive
```

logging syslog-facility

The **logging syslog-facility** command configures the Array to use a specific syslog facility.

Syntax

```
logging syslog-facility default | local<n>
```

Specifying **default** for the syslog facility causes applications to use their default syslog facility.

Command Mode

Config mode

Example

The following example configures the Array to use syslog facility local7.

```
(config) # logging syslog-facility local7
```

logging trap

The **logging trap** command sets the minimum severity of log messages sent to syslog servers.

Syntax

```
logging [<ip-address>] trap <severity>
[no] logging trap
```

Where `<ip-address>` is the IP address of a syslog server. Specifying an IP address causes the command to apply only to the specified syslog server. The **no** form of the command disables sending of log messages to syslog servers.

Command Mode

Config mode

Example

The following example sets the minimum severity level for log messages sent to syslog servers to “alert”.

```
(config) # logging trap alert
```

logging trap override class

The **logging trap override class** command sets or removes a per-class override on the logging level for syslog messages sent to a specific server. All classes that do not have an override set use the global logging level set with the **logging local** command; any classes that do have an override will do as the override specifies.

In the context of this command, class is a synonym for syslog facility. It allows log messages to be divided up according to their origin. Syslog classes include mgmt-core (for mgmtd alone), mgmt-back (for other back-end components), and mgmt-front (for front-end components, utilities, and tests).

Syntax

```
[no] logging <ip-address> trap override class
logging <ip-address> trap override class <class> priority {<severity> | none}
[no] logging <ip-address> trap override class <class>
```

Where `<ip-address>` is the IP address of a syslog server. Specifying **none** for the priority causes nothing from this class to be logged. The **no** form of the command disables the override for the class.

Command Mode

Config mode

Example

The following example configures events in the Kernel class to be logged on syslog server 10.10.10.10 with a priority of “info”.

```
(config) # logging 10.10.10.10 trap override class kern priority info
```

monitor

The **monitor** command displays messages on the terminal screen as the VCMs on a Violin Array are booted or upgraded. The messages are displayed until the boot or upgrade process is complete, or CTRL-C is pressed.

Syntax

```
monitor {boot | upgrades | vimm-upgrades}
```

where:

`boot`	Displays messages on the terminal screen as the VCMs in a Violin Array are booted.
`upgrades`	Displays messages on the terminal screen as the VCMs in a Violin Array are upgraded.
`vimm-upgrades`	Displays messages on the terminal during the VIMM staged upgrade process.

Command Mode

Config mode

Example

The following displays boot-related messages on the terminal screen when a VCM is rebooted.

```
(config) # monitor boot
Press control-C to return...

vcm-a:
System booting (0.0% complete)
Data plane disabled
Scheduler paused
Port 2 (x4) negotiated to 0 lanes
```

ntp disable

The **ntp disable** command disables or enables Network Time Protocol (NTP) on the Violin Array.

Syntax

```
[no] ntp disable
```

The **no** form of the command enables NTP. By default, NTP is disabled on the Violin Array.

Command Mode

Config mode

Examples

The following example disables NTP on the Violin Array.

```
(config) # ntp disable
```

The following example enables NTP on the Violin Array.

```
(config) # no ntp disable
```

ntp enable

The **ntp enable** command enables or disables Network Time Protocol (NTP) on the Violin Array.

Syntax

```
[no] ntp enable
```

The **no** form of the command disables NTP. By default, NTP is disabled.

Command Mode

Config mode

Example

The following example enables NTP on the Violin Array.

```
(config) # ntp enable
```

ntp peer

The **ntp peer** command configures settings for an NTP peer on the Violin Array.

Syntax

```
ntp peer <ip-address> [version <number>]
no ntp peer <ip-address>
[no] ntp peer <ip-address> disable
```

Allowable NTP version numbers are 3 and 4. If no NTP version number is specified when adding a peer, the default is 4. The **no** form of the command removes the NTP peer from the configuration.

The **disable** keyword disables or re-enables the NTP peer. Peers start enabled; disabling is just a way of making them temporarily inactive without losing their configuration.

Command Mode

Config mode

Example

The following example specifies an NTP peer on the Violin Array.

```
(config) # ntp peer 10.10.10.10
```

ntp server

The **ntp server** command configures settings for an NTP server on the Violin Array.

Syntax

```
ntp server <ip-address> [version <number>]
no ntp server <ip-address>
[no] ntp server <ip-address> disable
```

Allowable NTP version numbers are 3 and 4. If no NTP version number is specified when adding a server, the default is 4. The **no** form of the command removes the NTP server from the configuration.

The **disable** keyword disables or re-enables the NTP server. Servers start enabled; disabling is just a way of making them temporarily inactive without losing their configuration.

Command Mode

Config mode

Example

The following example specifies an NTP server on the Violin Array.

```
(config) # ntp server 10.1.1.10
```

ntpdate

The **ntpdate** command sets the system clock using a specified NTP server. This is a one-time operation and does not cause the clock to be kept in sync on an ongoing basis.

Syntax

```
ntpdate <ip-address>
```

Note that if you enter the **ntpdate** command while NTP is already enabled, it will generate an error, since the socket it requires is already in use.

Command Mode

Enable mode, Config mode

Example

The following example sets the system clock using an NTP server at 10.1.1.10.

```
# ntpdate 10.1.1.10
```

password strong

The **password strong** command turns on the criteria check for password strength. A strong password is between eight and 127 characters long, contains both uppercase and lowercase letters and at least one numeral.

Syntax

```
[no] password strong
```

The **no** form of the command turns off the criteria check.

Command Mode

Config mode

Example

The following example turns on criteria check for password strength.

```
# password strong
```

pcie connect

The **pcie connect** command configures the PCIe routing mode for connecting the Violin Array to a Memory Gateway.

Syntax

```
pcie connect {direct-attach | mg}
```

where:

`direct-attach`	Sets the PCIe routing mode to “direct-attach” mode. This mode is four PCIe direct connections, with two on each ACM. The four PCIe ports run at gen-2 speed with a PCIe lane width of x8 lanes per port.
`mg`	Sets the PCIe routing mode to Memory Gateway (mg) mode. In mg mode, the Violin Array uses its internal Memory Gateway modules rather than Memory Gateways connected to the PCIe ports.

Command Mode

Config mode

Example

The following example sets the PCIe routing mode for the Violin Array to “direct-attach” mode.

```
(config) # pcie connect direct-attach
```

ping

The **ping** command sends ICMP echo requests to remote hosts on the network.

Syntax

```
ping [<options>] [<hostname>]
```

where:

`<options>`	Are one or more options for the **ping** command. Enter the **ping** command without any parameters to list the available options. See this link for a description of the options.
`<hostname>`	Is the hostname of the remote system. Press CTRL+C to stop the ping requests.

Command Mode

Standard mode, Enable mode, Config mode

Example

The following shows an example of using the **ping** command to ping a remote Violin Array.

```
> ping vma01.vmem.com
PING vma01.vmem.com (10.1.9.2) 56(84) bytes of data.
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=4 ttl=64 time=0.047 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=5 ttl=64 time=0.044 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=6 ttl=64 time=0.043 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=7 ttl=64 time=0.052 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=8 ttl=64 time=0.044 ms
64 bytes from vma01.vmem.com (10.1.9.2): icmp_seq=9 ttl=64 time=0.049 ms
^C
--- vma01.vmem.com ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 7997ms
rtt min/avg/max/mdev = 0.039/0.044/0.052/0.008 ms
```

reload

The **reload** command shuts down or reboots the Violin Array.

Syntax

```
reload [force] [noconfirm] [halt [noconfirm]]
```

where:

`force`	Reboots the Violin Array immediately.
`halt`	Shuts down the device.
`noconfirm`	Reboots or shuts down the system without prompting to save configuration changes.

Command Mode

Enable mode, Config mode

Example

The following example reboots the Violin Array and prompts to save configuration changes if applicable.

```
# reload
```

service small-servers

The **service small-servers** command enables various TCP/UDP “small server” services: `echo`, `chargen`, `discard`, `daytime`, and `time`. These services are bundled together and cannot be enabled or disabled individually.

Syntax

```
[no] service {tcp-small-servers | udp-small-servers}
```

The **no** form of the command disables the TCP or UDP small server services. The TCP or UDP small server services are disabled by default.

Command Mode

Config mode

Example

The following example enables the `echo`, `chargen`, `discard`, `daytime`, and `time` services on TCP.

```
(config) # service tcp-small-servers
```

show aaa

The **show aaa** command displays the configuration settings for Authentication and Authorization, as well as information about any configured RADIUS servers.

Syntax

```
show aaa
```

Command Mode

Enable mode, Config mode

Example

The following example displays the current AAA configuration.

```
# show aaa
AAA authorization:
   Default User: admin
   Map Order: remote-first
Authentication method(s):
   local
No accounting methods configured.
```

The following fields are displayed by the command:

`Default User`	The name of the local account whose privileges are granted to users who log in using a non-local authentication method.
`Map Order`	How the system uses attributes received from a remote authentication server to map authenticated users to a local user account. This can be one of the following: `remote-first` If a local-user mapping attribute is returned from the authentication server, and it is a valid local username, map the authenticated user to the local user specified in the attribute. Otherwise, if the attribute is not present or not valid locally, map the authenticated user to the account specified by the **aaa authorization map default-user** command. `remote-only` Only try to map a remote authenticated user if the authentication server sends a local-user mapping attribute. If the local-user mapping attribute does not specify a valid local user, no further mapping is tried. `local-only` All remotely authenticated users will be mapped to the user specified by the **aaa authorization map default-user** command. Any vendor attributes received from the authentication server are ignored.
`Authentication method(s)`	The list of authentication methods a user is subject to when logging into the system, in the order they are tried.

Note: The CLI does not have a command to specify the acct-port (accounting port).

show alarms

The **show alarms** command displays the active system alarms on a Violin Array.

Syntax

```
show alarms [<alarm-type>]
```

By default, all system alarms are displayed. Specifying the `<alarm-type>` option limits the display to alarms in that category.

Command Mode

Enable mode, Config mode

Examples

The following example displays the system alarms for the Violin Array.

```
# show alarms
--- show alarm for hostname-acma at Wed Nov 13 15:54:11 2018

No ACM alarms
No FPM alarms
vcm-a
    No alarms
vcm-b
    No alarms
vcm-c:
Temp unreadable, last at 44 C
System booting (0.0% complete)
Data plane disabled
Scheduler paused
Port 1 (x4) negotiated to 0 lanes
Port 2 (x4) negotiated to 0 lanes
vcm-d
    No alarms
No VIMM alarms
No FPM alarms
No MG alarms.
No HBA alarms
No PCM alarms
No PSU alarms
No FAN alarms
No RAID alarms
```

The following example displays the VIMM-related alarms for the Violin Array.

```
# show alarms vimm
--- show alarm for hostname-acma at Wed Nov 13 15:54:11 2018

VIMM alarms:
VIMM 0 temp unreadable, last at 37 C
VIMM 1 temp unreadable, last at 38 C
VIMM 2 temp unreadable, last at 38 C
VIMM 3 temp unreadable, last at 39 C
```

show arp

The **show arp** command displays the contents of the ARP cache.

Syntax

```
show arp [static]
```

By default, both static and dynamic ARP cache entries are displayed. The **static** option limits the display to static ARP entries only.

Command Mode

Enable mode, Config mode

Examples

The following example displays the contents of the ARP cache.

```
# show arp
ARP cache contents
  IP 10.1.8.1 maps to MAC 00:19:BB:06:66:00 (interface eth1)
  IP 10.1.8.5 maps to MAC 00:1E:C9:52:EA:F2 (interface eth1)
  IP 10.1.9.1 maps to MAC 00:1F:C9:53:EA:F2 (interface eth1)
```

The following example displays the static entries in the ARP cache.

```
# show arp static
Static ARP entries
  IP 10.1.9.1 maps to MAC 00:1F:C9:53:EA:F2
```

show array

The **show array** command displays information about hardware components on a Violin Array.

Syntax

```
show array [detail]
```

The **detail** option shows additional information about power, LED, and fan status.

Command Mode

Enable mode, Config mode

Examples

The following example displays hardware information about a Violin Array.

```
# show array

Array                              Type       #VIMM  Alrm  Ready  T/Con T/Amb  MGs Status
-------------------------------------------------------------------------------------------
VIOLIN_MEMORY_ARRAY_1S813F00202  XVS 8264   64     no    no     25    18     optimal
```

The following fields are displayed by the command:

`Array`	The name and serial number of the Violin Array.
`Type`	Model number of the Violin Array.
`#VIMM`	The number of VIMMs installed in the Violin Array.
`Alrm`	Whether there are any active alarms on the Violin Array.
`Ready`	The status of the data plane.
`T/Con`	The temperature of the Array Controller.
`T/Amb`	The ambient temperature of the Violin Array.
`MGs Status`	Status of the internal Memory Gateways.

The following example displays additional information about LED status and fan speed.

```
# show array detail
ARRAY: VIOLIN_MEMORY_ARRAY_1S813F00202
  Chassis Type                  : XVS 8264
  Number of VIMMs               : 64
  Ambient Temp                  : 19
  Power A                       : ON
  Power B                       : ON
  Lid Ajar Time                 : 0 secs
  Alarm LED                     : OFF
  Power A LED                   : ON
  Power B LED                   : ON
  MG-a Status                   : running
  MG-b Status                   : running
Fans:
  fan-a1                        : Medium
  fan-b1                        : Slow
  fan-c1                        : Slow
  fan-a0                        : Medium
  fan-b0                        : Slow
  fan-c0                        : Slow
```

show array balance

The **show array balance** command displays the current RAID rebalance settings and the balance status of the VCMs in a Violin Array.

Syntax

```
show array balance
```

Command Mode

Enable mode, Config mode

Example

The following example displays RAID balance information for a Violin Array.

```
# show array balance
auto balance            : yes
one-time balance        : no
weekly balance          : no

vcm-a                   : balance in progress
vcm-b                   : balanced
vcm-c                   : balanced
vcm-d                   : unbalanced
```

The following fields are displayed by the command:

`auto balance`	Whether automatic balancing is enabled, set with the **array balance enable** command. When unbalanced RAID groups are detected, they are automatically rebuilt.
`one-time balance`	Whether a one-time RAID rebuild is scheduled to occur at a specified time and date, set with the **array balance schedule once** command.
`weekly balance`	Whether RAID rebuilds are scheduled to occur on a weekly basis, set with the **array balance schedule weekly** command
`vcm-`*x*	The current status for each VCM in the Violin Array: `balanced`, `unbalanced`, or `balance in progress`.

show array cooling

The **show array cooling** command displays the cooling policy information for the array and at what temperature the environmental monitor takes action. Environmental monitor actions:

Action	Description
Cool	If all devices in the zone are at or below the default or specified temperature, then the fan speed is reduced.
Hot	If only one device in a zone is at or above the default or specified limit, then the fans for that zone are raised to high.
Alarm	When a device reaches the default or specified temperature the ACM raises an alarm. If callhome is configured, then an email is sent.

Syntax

```
show array cooling
```

Command Mode

Enable mode, config mode

Example

```
# show array cooling

Temperature Thresholds
  acm         : cool 65, hot 67, alarm 70
  mg          : cool 65, hot 67, alarm 70
  vcm         : cool 64, hot 67, alarm 70
  vimm        : cool 65, hot 67, alarm 70
  ambient     : cool 30, hot 32, alarm 35
```

show array modules

The **show array modules** command displays status, alarm, and inventory information about individual modules or types of modules installed on a Violin Array.

Syntax

```
show array modules id <module-id> [summary] [detail] [upgrade-log]
show array modules type {<module-type> | all} [summary] [detail] [upgrade-images]
```

where:Command Mode

`<module-id>`	Displays information about a specific VIMM, VCM, ACM, Memory Gateway, or host bus adapter.
`<module-type>`	Displays information about the VIMMs, VCMs, ACMs, Power Controller Modules, Front Panel Module, Memory Gateways, or host bus adapters installed in the Violin Array. Use the **all** option to display information about all modules.
`summary`	Displays basic information about the module or module type. When specified with module type **vimm**, displays statistics for the VIMMs installed in the Violin Array.
`detail`	Displays additional information about the module or module type.
`upgrade-log`	Displays the contents of the upgrade log for the module.
`upgrade-images`	Displays upgrade images available for the specified module type.

Enable mode, Config mode

Examples

The following example displays information about the VIMMs installed in the Violin Array.

```
# show array modules type vimm

ViMM    VCM      RG    Type      OOS    Status      Temp    %-       %-        %-        %-        %-
                                                    (C)     FmtCap   DieFail   BlkFail   BlkEraA   LifeTi
                                                                                         vg        meRem

00      vcm-a    0     2T-       no     Active      40      84       0.00      1.33      0.00      100.00
                       MLC-
                       Flash

01               1     2T-       no     Active      43      84       0.00      1.18      0.00      100.00
                       MLC-
                       Flash

02               0     2T-       no     Active      42      84       0.00      1.42      0.00      100.00
                       MLC-
                       Flash

03               s     2T-       no     Spare(F     42      0        0.00      1.17      0.00      100.00
                       MLC-              )
                       Flash

[more output follows]
```

The following fields are displayed by the command:

`VIMM`	The ID of the VIMM.
`VCM`	The vRAID Controller Module (VCM) that manages the VIMM.
`RG`	The RAID group to which the VIMM belongs.
`Type`	The type of VIMM.
`OOS`	Whether the `out-of-service` flag has been activated for the VIMM.
`Status`	The status of the VIMM: active, spare, or VIMM not present in the slot.
`Temp(C)`	Temperature of the VIMM.
`%-FmtCap`	The formatted storage capacity percentage for the VIMM, which can be set with the **array format capacity** command. A single level cell (SLC) system is formatted to 65%, and a multi-level cell (MLC) system is formatted to 84%.
`%-DieFail`	The percentage of the die on the VIMM that has failed.
`%-BlkFail`	The percentage of blocks on the VIMM that have failed.
`%-BlkEraAvg`	The block erasure percentage for the VIMM.
`%-LifeTimeRem`	The percentage of the VIMM's total lifetime remaining.

The **summary** option displays statistics about all of the VIMMs in the Violin Array. For example:

```
# show array modules type vimm summary
number of vimms:          64
healthy:                  64
type:                     2T-MLC-Flash
active:                   60
spare:                    4
boot:                     0
admin down:               0
failed:                   0
out-of-service:           0
alarmed:                  0
temperature:              50/64 (max temp vimm 44)
```

The following example displays additional information about a specific VIMM, including power consumption, serial number, and FPGA and software versions.

```
# show array modules id vimm20 detail
VIMM20:
   VCM                    : vcm-a
   Type                   : 2T-MLC-Flash
   Status                 : Active
   Present                : yes
   Power                  : yes
   Out-of-Service         : no
   Current (mA)           : 1236.96
   RAID                   : 0
   Spare                  : no
   Health                 : health threshold: (OK)
   Temp (C)               : 48
   Serial                 : 2N812F00639
   Model                  : 410-0344-00_R03
   Date                   : 20180604
   FpgaVersion            : 7.1.2.3
   SwVersion              : 7.2.0.0
   %-Format Capacity      : 84
   %-DieFail              : 0.00
   %-BlkFail              : 1.14
   %-BlkEraseAvg          : 0.00
   %-LifeTimeRemaining    : 100.00
     Bytes read           : 2,581,913,185,280
     Bytes written        : 1,253,355,149,312
     ECC Corrected        : 1,558 (rate: 0.00e+00)
```

The following example displays information about a VCM, including the IDs of the VIMMs managed by the VCM.

```
# show array modules id vcm-a
vcm-a
present                 : yes
power                   : yes
state                   : active
out-of-service          : no
vimm assigned           : 00-04,20-23,25,40-41,43-45,49
temperature             : 48
frequency               : 500
sw version              : [vcm A7.5.0.1 #2
(1d651dd)|modsw:1d651dd,modhw:9775542,tallmaple:743fbf5] Tue Oct 23
18:20:58 PDT 2018 root@ci-ginkgo1

mfg serial              : 2S812F00046
mfg model               : 620-0216-00_R01
mfg version             : 01
mfg date                : 06042018
alarms:
```

The following example displays information about the internal Memory Gateways.

```
# show array modules type mg detail

mg-a:
Present                 : yes
Running                 : yes
Mfg serial              : 1Y813B00264
Mfg model               : 420-0138-01_R01
Mfg version             : 01
Mfg date                : 06042018
out-of-service          : no
Power                   : yes
Current (mA)            : 3760.16
Temperature (C)         : 27
Host type               : Concerto OS
Hw config               : SHB5-768
Internal switch port    : 4
IP address              : 169.254.1.101
HW address              : 00:90:FB:5C:42:71
External IP address     : 10.5.11.200

mg-b:
Present                 : yes
Running                 : yes
Mfg serial              : 1Y813B00263
Mfg model               : 420-0138-01_R01
Mfg version             : 01
Mfg date                : 06042018
out-of-service          : no
Power                   : yes
Current (mA)            : 3637.19
Temperature (C)         : 29
Host type               : Concerto OS
Hw config               : SHB5-768
Internal switch port    : 6
IP address              : 169.254.1.102
HW address              : 00:90:FB:60:16:15
External IP address     : 10.5.11.132
```

show array serial-logging

The **show array serial-logging** command indicates which MGs and VCMs on a Violin Array have serial logging enabled.

Syntax

```
show array serial-logging
```

Command Mode

Enable mode, Config mode

Example

The following example displays the serial-logging settings for a Violin Array.

```
# show array serial-logging
Component    Serial Logging
-------------------------------
  vcm-a        On
  vcm-b        On
  vcm-c        On
  vcm-d        On
  mg-a         On
  mg-b         On
```

show banner

The **show banner** command displays the text configured for the login banner and the message of the day (MOTD) banner.

Syntax

```
show banner
```

Command Mode

Standard mode, Enable mode, Config mode

Example

The following example displays the configured banners.

```
# show banner
Banners:
    MOTD:
Violin Systems Array Controller

    Login:
Unauthorized Access Prohibited.  Usage of the Violin Array is subject
to the Violin Systems License agreement which is included under this product's
Web Interface Help section.
```

show bootvar

The **show bootvar** command displays information about the software image files and boot options on the Violin Array.

Syntax

```
show bootvar
```

Command Mode

Enable mode, Config mode

Example

The following example displays image information for the Violin Array.

```
# show bootvar
Installed images:

  Partition 1:
  supervisor A7.5.0.1 #3-ir 2018-09-25 16:18:08 ppc acm root@ci-
ginkgo1:super:069b27d

  Partition 2:
  supervisor A7.5.0.1 #2 2018-10-23 19:51:09 ppc acm root@ci-
ginkgo1:super:1d651dd

Last boot partition: 1
Next boot partition: 1

Serve image files via HTTP/HTTPS: no

No boot manager password is set.

Image signing: signature validation disabled
Admin require signed images: yes

Settings for next boot only:
   Fallback reboot on configuration failure: yes (default)
```

The following fields are displayed by the command:

`Installed images`	The software images loaded on the two boot partitions.
`Last boot partition`	Which of the boot partitions supplied the software image during the last system reboot.

`Next boot partition`	The boot partition that will supply the software image the next time the system is rebooted.
`No boot manager password is set`	Whether the boot manager is password-protected.
`Image install currently in progress`	Whether a software image was in progress when the **show bootvar** command was run.
`Image signing`	Whether software images are required to be signed with a trusted signature.
`Admin require signed images`	Whether software images installed by the admin user are required to be signed with a trusted signature.
`Settings for next boot only`	Settings that apply to the next time the system is rebooted.
`Fallback reboot on configuration failure`	Whether the system is booted with the previous software image in the event the system configuration cannot be loaded with a new software image.

show chassis info

The **show chassis info** command displays basic information about a Violin Array chassis, including the product series and chassis serial number.

Syntax

```
show chassis info
```

Command Mode

Enable mode, Config mode

Examples

The following example displays information about a Violin Array chassis.

```
# show chassis info
    Product Series : XVS 8264
    Serial Number  : 1S813F00202
```

The following fields are displayed by the command:

`Product Series`	Model number of product series
`Serial Number`	Chassis serial number.

show cli

The **show cli** command displays the settings configured for CLI sessions on the Violin Array.

Syntax

```
show cli
```

Command Mode

Enable mode, Config mode

Example

The following example displays the CLI settings for the Violin Array. These settings are controlled with the **cli default** and **cli session** commands.

```
# show cli
CLI current session settings:
  Maximum line size:        8192
  Terminal width:           189 columns
  Terminal length:          50 rows
  Terminal type:            xterm
  X display setting:        (none)
  Auto-logout:              15 minutes
  Paging:                   enabled
  Progress tracking:        enabled
  Prefix modes:             disabled

CLI defaults for future sessions:
  Auto-logout:              15 minutes
  Paging:                   enabled
  Progress tracking:        enabled
  Prefix modes:             disabled

Settings for both this session and future ones:
  Show hidden config:        yes
  Confirm losing changes:    yes
  Confirm reboot/shutdown:   yes
  Confirm factory reset:     yes
  Prompt on empty password:  yes
```

show clock

The **show clock** command displays the current system time, date, and configured time zone.

Syntax

```
show clock
```

Command Mode

Standard mode, Enable mode, Config mode

Example

The following example displays the current system time.

```
# show clock
Time:       00:55:12
Date:       2018/11/30
Time zone:  America North United_States Pacific
            (US/Pacific)
UTC offset: -0800 (UTC minus 8 hours)
```

show cluster

The **show cluster** command displays cluster status on the Violin Array.

Syntax

```
show cluster global [brief]
show cluster local [error-status]
show cluster master
show cluster standby
```

where:

`global`	Displays the global status of the cluster.
`brief`	Displays a summary of the global cluster status.
`local`	Displays the cluster status for the local node.
`error-status`	Lists cluster errors on the local node.
`master`	Displays the status for the cluster master.
`standby`	Displays the status for the standby node.

Command Mode

Enable mode, Config mode

Example

The following example displays global status of the cluster.

```
# show cluster global
Cluster ID:          15000-0000-0000
Cluster name:        1S813F00202
Management IP:       10.5.12.11/20
Cluster master IF:   eth1
Cluster node count:  2

Node Status:
        Node ID: 6  <--- (local node)
        Host ID: e86a639b6506
        Hostname: lab-fil4140-acm-b
        Node Role: master
        Node State: online
        Node internal address: 169.254.1.11, port: 60102
        Node external address: 10.5.10.34
        Recv. Heartbeats from: 7
        Send Heartbeats to: 7
[more output follows]
```

show cluster configured

The **show cluster configured** command displays global cluster configuration settings on the Violin Array.

Syntax

```
show cluster configured
```

Command Mode

Enable mode, Config mode

Example

The following example displays the global cluster configuration settings.

```
# show cluster configured
Global cluster config:
        Cluster enabled: yes
        Cluster ID: 99999-9999-1014
        Cluster name: Cluster-KYLE
        Cluster interface: eth1
        Cluster port: 60102
        Cluster max nodes: 50
        Cluster expected nodes: 1
        Cluster startup time: 180
        Cluster shared secret: 1234567890123456
        Cluster master auto-discovery enabled: yes
        Cluster master manual IP address: 0.0.0.0
        Cluster master manual port: 60102
        Cluster master virtual IP address: 10.1.9.192/22
        Cluster master interface: eth1
```

The following fields are displayed by the command:

`Cluster enabled`	Whether the cluster has been enabled with the **cluster enable** command.
`Cluster ID`	The identifier for the cluster. All nodes configured with a given cluster ID are part of the same cluster.
`Cluster name`	The name of the cluster. The cluster name has an equivalent function to a hostname.
`Cluster interface`	The interface for the cluster service.
`Cluster port`	The service port for the cluster.
`Cluster max nodes`	The maximum number of nodes that can be configured in a cluster.

`Cluster expected nodes`	The number of nodes the cluster master checks for in the cluster.
`Cluster startup time`	The maximum number of seconds allowed for the startup phase for a cluster.
`Cluster shared secret`	The shared secret used for authenticating messages between nodes in a cluster.
`Cluster master auto-discovery enabled`	Whether auto-discovery of the cluster master is enabled. Auto-discovery is enabled by default.
`Cluster master manual IP address`	The cluster master IP address.
`Cluster master manual port`	The cluster master port number.
`Cluster master virtual IP address`	The cluster master virtual IP address and netmask. The virtual IP address is installed by the cluster master, and all other cluster nodes ensure they do not install the virtual IP address.
`Cluster master interface`	The interface to which the master virtual address is assigned.

show cluster upgrade staged

The **show cluster upgrade staged** command checks whether the Violin Array is ready for staged upgrade.

Syntax

```
show cluster upgrade staged status
```

Command Mode

Enable mode, Config mode

Example

The following example displays the upgrade readiness of the cluster.

```
# show cluster upgrade staged status
Checking readiness for staged upgrade...
  - Array failed scrubber check. Wait for VIMMs to complete scrubbing.
  - Scrubbing progress: 64.42 percent

Array is not ready for staged upgrade
```

show configuration

The **show configuration** command displays the active saved configuration for the Violin Array.

Syntax

```
show configuration [full] [running] [text files]
```

where:

`full`	Displays all of the commands in the active saved configuration, including those that set default values.
`running`	Displays the commands in the running configuration.
`text files`	Lists the configuration text files stored on the Violin Array.

If you enter the **show configuration** command with no options, the commands in the active saved configuration are displayed.

Command Mode

Enable mode, Config mode

Examples

The following example displays the active saved configuration for the Violin Array.

```
# show configuration files initial
##
## Active saved database "initial"
## Generated at 2018/11/04 18:56:29 -0800
## Hostname: hostname-acma
##

##
## License keys
##
   license install LK2-RESTRICTED_CMDS-7X87-4VKJ-8A46-1234-1234
   license install LK2-VSHARE-7X87-4VKJ-E9R8-1234-1234

##
## Network interface configuration
##
   interface eth0 create
   interface eth1 create
   interface eth2 create

[more output follows]
```

show configuration files

The **show configuration files** command lists the configuration files stored on the Violin Array and also can display the contents of a specified configuration file.

Syntax

```
show configuration files [<filename>]
```

Entering the **show configuration files** command without the `<filename>` option lists the configuration files stored on the Violin Array. Specifying a `<filename>` displays the contents of the file.

Command Mode

Enable mode, Config mode

Examples

The following example lists the configuration files on the Violin Array.

```
# show configuration files
initial (active)
initial.bak

Active configuration: initial
Unsaved changes:      no
```

The following example displays the contents of a configuration file.

```
# show configuration files initial
##
## Active saved database "initial"
## Generated at 2018/11/04 18:56:29 -0800
## Hostname: hostname-acma
##

##
## License keys
##
   license install LK2-RESTRICTED_CMDS-7X87-4VKJ-8A46-1234-1234
   license install LK2-VSHARE-7X87-4VKJ-E9R8-1234-1234

##
## Network interface configuration
##
   interface eth0 create
   interface eth1 create
   interface eth2 create

[more output follows]
```

show email

The **show email** command displays the settings for sending notification e-mails when various informational or failure events occur on the Violin Array.

Syntax

```
show email [events]
```

The **events** option lists the informational and failure events for which notification e-mails are sent, as well as which events generate callhome e-mails.

Command Mode

Enable mode, Config mode

Example

The following example displays the e-mail settings on the Violin Array.

```
# show email
Mail hub:          callhome.vmem.com
Mail hub port:     25
Domain override:
Return address:    do-not-reply
Include hostname in return address: yes

Current reply address: do-not-reply@VKYLE.eng.vmem.int

Security mode:         tls-none
SSL min version:       tls1.2
Verify server cert:    yes
Supplemental CA list: default-ca-list

SMTP authentication: disabled

Dead letter settings:
  Save dead.letter files: yes
  Dead letter max age:    14 days

Consolidate email settings:
  Enable:        no
  Period:        60
  Max_events:    5

Email notification recipients:
   callhome@vmem.com (all events, in detail)

Callhome emails
  Enabled:       yes
  AutoUpload:    yes
  Recipient:     callhome@vmem.com
  Mail hub:      callhome.vmem.com
  SMTP authentication: disabled
```

show eventlog

The **show eventlog** command lists the contents of the system event log if it has been enabled.

Syntax

```
show eventlog [id <event-id> [detailed]]
show eventlog recent [<n> [detailed]]
```

where:

`id <event-id>`	Displays all of the commands in the active saved configuration, including those that set default values.
`recent`	Lists the most recent events. To display the *n*th most recent event in the event log, use the `<n>` parameter.
`detailed`	Displays details about the specified event.

Command Mode

Enable mode, Config mode

Example

The following example lists the events in the system event log.

```
# show eventlog
ID    Time                 Description
-------------------------------------------------------------------------------
43931  2018/11/09 07:56:55 Alarm cleared by vcm-c.  Message: Upgrading
43932  2018/11/09 07:57:15  Alarm set by raid 6.  Message: RAID group has no spare
43933  2018/11/09 07:57:15  Alarm set by raid 7.  Message: RAID group has no spare
43934  2018/11/09 07:57:15  Alarm set by raid 8.  Message: RAID group has no spare
43935  2018/11/09 07:57:15  Alarm set by raid 9.  Message: RAID group has no spare
43936  2018/11/09 07:57:15  Alarm set by raid10.  Message: RAID group has no spare
43937  2018/11/09 07:57:15  Alarm set by raid11.  Message: RAID group has no spare
43938  2018/11/09 07:57:18  Alarm cleared by raid 6.  Message: RAID group has no spare
43939  2018/11/09 07:57:18  Alarm cleared by raid 7.  Message: RAID group has no spare
43940  2018/11/09 07:57:18  Alarm cleared by raid 8.  Message: RAID group has no spare
...
43950  2018/11/09 07:59:15  Alarm cleared by vcm-d.  Message: Upgrading - waiting to
reboot.
43951  2018/11/09 07:59:15  Alarm set by vcm-d.  Message: Upgrading - rebooting.
43952  2018/11/09 07:59:19  Alarm set by vcm-d.  Message: Powered off
43953  2018/11/09 07:59:22  Alarm cleared by vcm-d.  Message: Powered off
43954  2018/11/09 07:59:22  Alarm cleared by raid 6.  Message: RAID group has no spare
```

The following example lists the 100th most recent event in the system event log.

```
# show eventlog recent 100
ID     Time                 Description
-------------------------------------------------------------------------------
44831  2018/11/25 15:51:35  admin has logged out (cli) from IP: 10.12.56.25.
```

The following example shows details about a specified event in the system event log.

```
# show eventlog id 2901 detailed
Event ID:     43934
Time:         2018/11/09 07:57:15
Description:  Alarm set by raid 8.  Message: RAID group has no spare
Event Name:   /alarm/events/device/alarm-set
Bindings:
  device:  raid 8
  alarm_id:  10002
  alarm_severity:  1
  alarm_msg:  RAID group has no spare
```

show files

The **show files** command lists the debug dump (sysdump), TCP dump, and statistics report files accumulated on the Violin Array, and can display the contents of a specified file.

Syntax

```
show files {debug-dump | stats | tcpdump} [<filename>]
```

Entering the **show files** command without the `<filename>` option lists the files of the specified type stored on the Violin Array. Specifying a `<filename>` displays the contents of the file.

Command Mode

Enable mode, Config mode

Examples

The following example lists the debug dump files on the Violin Array.

```
# show files debug-dump
sysinfo-sysdump-VMEM01-20120626-185055.txt
sysinfo-sysdump-VMEM01-20120626-185039.txt
sysinfo-sysdump-VMEM01-20120626-185024.tgz
```

The following example displays the contents of a debug dump file.

```
# show files debug-dump sysinfo-sysdump-VMEM01-20120626-185024.tgz
================================================
Event information:

Description:    Unexpected failure of process mgmtd
Binary path:    /opt/tms/bin/mgmtd
Binary size:    13291844
Binary time:    2018-11-05 01:29:00 -0800
Core name:      core.4624
Core size:      40042496
Core time:      2018-11-05 15:16:57 -0800
Process ID:     4624
Fatal signal:   SIGSEGV
Process uptime: 7h 9m 5.410s

Backtrace:

Core was generated by `/opt/tms/bin/mgmtd'.

[more output follows]
```

show files system

The **show files system** command displays information about the used and available space on the `/config` and `/var` filesystems on the Violin Array.

Syntax

```
show files system
```

Command Mode

Standard mode, Enable mode, Config mode

Example

The following example displays filesystem information for the Violin Array.

```
# show files system
Statistics for /config filesystem:
  Space Total          124 MB
  Space Used           5 MB
  Space Free           119 MB
  Space Available      112 MB
  Space Percent Free   95%
  Inodes Percent Free  99%

Statistics for /var filesystem:
  Space Total          26214 MB
  Space Used           965 MB
  Space Free           25249 MB
  Space Available      23918 MB
  Space Percent Free   96%
  Inodes Percent Free  99%
```

show files var

The **show files var** command lists the files and free space percentage for the `/var` filesystem on the Violin Array.

Syntax

```
show files var all | summary
```

The **all** parameter lists all the files in the `/var` filesystem, ordered by size. The **summary** parameter lists the 24 largest files in the `/var` filesystem.

Command Mode

Enable mode, Config mode

Example

The following example lists the files in the `/var` filesystem on the Violin Array.

```
# show files var all
/var filesystem percentage free: 98 percent
Size (K)    File Name
-------------------------------------------------------
  2152    /var/opt/tms/snapshots/snap-V7-acm-b-20141029-161017.tgz
   328    /var/opt/tms/sysdumps/sysdump-V7-acm-b-20141029-161028.tgz
   272    /var/opt/tms/snapshots/.running/chassisd/chassisd.map
   204    /var/log/storylines
    72    /var/log/fru_check
    68    /var/log/wtmp
    40    /var/log/serialmg-b.log.2
    40    /var/log/serialmg-b.log.1
    40    /var/log/serialmg-b.log
    20    /var/log/serialmg-b.log.3
    20    /var/log/dmesg
    12    /var/log/web_access_log
    12    /var/log/serialvcm-d.log.2
    12    /var/log/serialvcm-d.log.1
    12    /var/log/serialvcm-c.log.2
    12    /var/log/serialvcm-c.log.1
     8    /var/log/build_versions/version-first-booted-20141024-.txt
     4    /var/opt/tms/snapshots/.running/wsmd/wsmd.pid
     4    /var/opt/tms/snapshots/.running/vprocmond/vprocmond.pid
     4    /var/opt/tms/snapshots/.running/voffload/voffload.pid
     4    /var/opt/tms/snapshots/.running/vmhubd/vmhubd.pid
     4    /var/opt/tms/snapshots/.running/vmapd/vmapd.pid
     4    /var/opt/tms/snapshots/.running/alarmd/alarmd.pid
     4    /var/opt/tms/snapshots/.running/acud/acud.pid
     4    /var/opt/tms/snapshots/.running/acmd/acmd.pid
     4    /var/log/wtmp.1.gz
     4    /var/log/web_error_log
     4    /var/log/systemlog
     4    /var/log/serialvcm-d.log
     4    /var/log/serialvcm-c.log
     4    /var/log/serialvcm-b.log
     4    /var/log/serialvcm-a.log
     4    /var/log/serialmg-b.log.5
     4    /var/log/serialmg-b.log.4
     4    /var/log/powerevents
     4    /var/log/lastlog
     4    /var/log/inventory
     0    /var/log/wdog_events.dat
     0    /var/log/i2ccheck
     0    /var/log/fpm_maint
```

show ftp-server

The **show ftp-server** command indicates whether the FTP server has been enabled on the Violin Array. By default, the FTP server is disabled.

Syntax

```
show ftp-server
```

Command Mode

Enable mode, Config mode

Example

The following example displays the status of the FTP server.

```
# show ftp-server
FTP server enabled:  no
SSL enabled:         yes
SSL min-version:     tls1.2
```

show hosts

The **show hosts** command shows values configured by host-related commands: hostname, name servers, domain name list, and static host mappings.

Syntax

```
show hosts
```

Command Mode

Enable mode, Config mode

Example

The following example displays the status of the FTP server.

```
# show hosts
Hostname: hostname-acma
Name server: 10.1.8.5 (configured)
Domain name: eng.vmem.int (configured)
IP 127.0.0.1 maps to hostname localhost
Automatically map hostname to loopback address: yes
```

show inventory

The **show inventory** command lists information about all of the ACMs, VCMs, and VIMMs installed in a Violin Array.

Syntax

```
show inventory
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the modules in a Violin Array.

```
# show inventory
 ---------------------------------------------------------------------------------------------
       inventory scan started at 11:54:15 11/29/2018 and is complete
 =============================================================================================
  device            presence      state          serial         part no.
 ---------------------------------------------------------------------------------------------
 chassis             present     verified      1S813F00202        XVS 8264  MIM3 type 8
     fpm             present     verified      38815F00007 620-0085-00_R12
 ---------------------------------------------------------------------------------------------
   acm-a             present     verified      52811F00015 410-0342-01_R02  ACM type 2
                           MAC addresses:    001B97005449    001B970053EA
                                 SD card:        1056E54A   ATP AF SD

   acm-b             present     verified      52811F00005 410-0342-01_R02  ACM type 2
                           MAC addresses:    001B9700544A    001B970053DA
                                 SD card:        1046E0E4   ATP AF SD
 ---------------------------------------------------------------------------------------------
   pcm-a             present     verified      56815F00015 410-0270-00_R05
   pcm-b             present     verified      56815F00006 410-0270-00_R05
 ---------------------------------------------------------------------------------------------
   psu-a             present     verified    K303QG000XAHP    DS2000-3-403
   psu-b             present     verified    K303QG000WAHP    DS2000-3-403
 ---------------------------------------------------------------------------------------------
   hba-a             present     verified    RFE1733U07993 410-0341-00_R01
   hba-b             present     verified    RFE1727T65712 410-0341-00_R01
   hba-c             present     verified    RFE1727T65706 410-0341-00_R01
   hba-d             present     verified    RFE1727T65765 410-0341-00_R01

  device   OOS      presence      state          serial         part no.     OOS reason
 ---------------------------------------------------------------------------------------------
    mg-a             present     verified      1Y813B00264 420-0138-01_R01
                                                HW config:        SHB5-768
    mg-b             present     verified      1Y813B00263 420-0138-01_R01
                                                HW config:        SHB5-768

  device   OOS      presence      state          serial         part no.     OOS reason
 ---------------------------------------------------------------------------------------------
   vcm-a             present     verified      2S812F00046 620-0216-00_R01
   vcm-b             present     verified      2S736F00056 620-0216-00_R01
   vcm-c             present     verified      2S736F00003 620-0216-00_R01
   vcm-d             present     verified      2S736F00050 620-0216-00_R01

  device   OOS      presence      state          serial         part no.        type     vimm set
 ---------------------------------------------------------------------------------------------
  vimm00             present     verified      2N812F01108 410-0344-00_R03    2T-MLC-Flash  .
  vimm01             present     verified      2N812F00390 410-0344-00_R03    2T-MLC-Flash  .
  vimm02             present     verified      2N812F00666 410-0344-00_R03    2T-MLC-Flash  .
  vimm03             present     verified      2N812F00683 410-0344-00_R03    2T-MLC-Flash  0
```

[more output follows]

The following fields are displayed by the command:

`device`	The identifier for the ACM, VCM, or VIMM.
`presence`	Whether the module is present in the slot.
`state`	Whether the reported information is the most current information.
`serial`	The serial number of the module.
`model`	The model number of the module.
`version`	The hardware version of the module.
`OOS`	Whether the `out-of-service` flag is activated for the module.
`fault detail`	Information about current alarms for the module.
`type`	The capacity and system type – single level cell (SLC) or multi-level cell (MLC) – of the VIMM.
`vimm set`	The spare status of the VIMM.

show images

The **show images** command displays information about the software image files and boot options on the Violin Array.

Syntax

```
show images
```

Command Mode

Enable mode, Config mode

Example

The following example displays image information for the Violin Array.

```
# show images
Installed images:

  Partition 1:
  supervisor A7.5.0.1 #3-ir 2018-09-25 16:18:08 ppc acm root@ci-
ginkgo1:super:069b27d

  Partition 2:
  supervisor A7.5.0.1 #2 2018-10-23 19:51:09 ppc acm root@ci-
ginkgo1:super:1d651dd

Last boot partition: 2
Next boot partition: 2

Images available to be installed:

  supervisorA7.5.0.01-dev2018-07-11133246ppcacmckahneng-builds-
ginkgosuperb726d12.img
  supervisor A7.5.0.1 #1-dev 2018-07-11 13:32:46 ppc acm ckahn@eng-
builds-ginkgo:super:b726d12

  supervisorA7.5.0.022018-10-23195109ppcacmrootci-
ginkgo1super1d651dd.img
  supervisor A7.5.0.1 #2 2018-10-23 19:51:09 ppc acm root@ci-
ginkgo1:super:1d651dd

  supervisorA7.5.0.03-ir2018-09-25161808ppcacmrootci-
ginkgo1super069b27d.img
  supervisor A7.5.0.1 #3-ir 2018-09-25 16:18:08 ppc acm root@ci-
ginkgo1:super:069b27d

Serve image files via HTTP/HTTPS: no

No image install currently in progress.

No boot manager password is set.

Image signing: signature validation disabled
Admin require signed images: yes

Settings for next boot only:
   Fallback reboot on configuration failure: yes (default)
```

The following fields are displayed by the command:

`Images available to be installed`	The software image files loaded on the system.
`Installed images`	The software images loaded on the two boot partitions.
`Last boot partition`	Which of the boot partitions supplied the software image during the last system reboot.
`Next boot partition`	The boot partition that will supply the software image the next time the system is rebooted.
`No boot manager password is set`	Whether the boot manager is password-protected.
`Image install currently in progress`	Whether a software image was in progress when the **show images** command was run.
`Image signing`	Whether software images are required to be signed with a trusted signature.
`Admin require signed images`	Whether software images installed by the admin user are required to be signed with a trusted signature.
`Settings for next boot only`	Settings that apply to the next time the system is rebooted.
`Fallback reboot on configuration failure`	Whether the system is booted with the previous software image in the event the system configuration cannot be loaded with a new software image.

show interfaces

The **show interfaces** command displays configuration information and traffic statistics for the Ethernet interfaces on the Violin Array.

Syntax

```
show interfaces [<ifname>] [brief] [configured]
```

You can display information for all interfaces or limit the display to a specified `<ifname>`. The **brief** option omits the TX and RX data from the output. The **configured** option displays the current values of the configurable settings for each displayed interface.

Command Mode

Enable mode, Config mode

Examples

The following example displays information about an interface on the Violin Array.

```
# show interfaces eth1
Interface eth1 status:
   Comment:
   Admin up:           yes
   Link up:            yes
   IP address:         10.1.10.136
   Netmask:            255.255.252.0
   Secondary address:  10.1.9.192/22 (alias: 'eth1:0')
   Speed:              1000Mb/s (auto)
   Duplex:             full (auto)
   Interface type:     ethernet
   Interface ifindex:  3
   Interface source:   physical
   MTU:                1500
   HW address:         00:25:90:01:23:45

   RX bytes:           1423170356          TX bytes:       700258161
   RX packets:         8041570             TX packets:     1131955
   RX mcast packets:   3991236             TX discards:    0
   RX discards:        0                   TX errors:      0
   RX errors:          0                   TX overruns:    0
   RX overruns:        0                   TX carrier:     0
   RX frame:           0                   TX collisions:  0
                                           TX queue len:   1000
```

The following example displays the configured settings for the interface. Each of the listed settings is configurable in the CLI.

```
# show interfaces eth1 configured
Interface eth1 configuration:
   Comment:
   Enabled:            yes
   DHCP:               no
   Zeroconf:           no
   IP address:         10.1.10.136
   Netmask:            255.255.252.0
   Speed:              auto
   Duplex:             auto
   MTU:                1500
```

show ip default-gateway

The **show ip default-gateway** command displays the currently active default route.

Syntax

```
show ip default-gateway [static]
```

The **static** option displays the statically configured default route, if one exists.

Command Mode

Enable mode, Config mode

Example

The following example displays the currently active default route.

```
# show ip default-gateway
Active default gateways:
   10.1.8.1 (interface: eth1)
```

show ip dhcp

The **show ip dhcp** command displays the DHCP configuration settings for the Violin Array.

Syntax

```
show ip dhcp
```

Command Mode

Enable mode, Config mode

Example

The following example displays the DHCP settings on the device.

```
# show ip dhcp
DHCP primary interface:
   Configured: eth1
   Active:     (none)

DHCP: yield default gateway to static configuration: no

DHCP Client Options:
   Send Hostname:    no
   Client Hostname:  VMG01 (using system hostname)
```

The following fields are displayed by the command. The fields reflect the settings configured with the **ip dhcp** command.

`Configured`	The interface on the Violin Array that will accept non-interface-specific configuration via DHCP.
`Active`	The interface currently active as a DHCP primary interface, if any.
`DHCP: yield default gateway to static configuration`	Whether the Violin Array ignores the default gateway assignment from the DHCP server if there is already a default gateway configured with the **ip default gateway** command.
`Send Hostname:`	Whether the Violin Array supplies a hostname to the DHCP server during negotiation.
`Client Hostname:`	The hostname to be supplied to the DHCP server when the **send-hostname** option is configured.

show ip route

The **show ip route** command displays the routing table in the system, including dynamic routes and any active static routes.

Syntax

```
show ip route [static]
```

The **static** option limits the display to statically configured routes.

Command Mode

Enable mode, Config mode

Example

The following example displays the active static and dynamic routes.

```
# show ip route
Destination     Mask            Gateway         Interface   Source
default         0.0.0.0         10.1.8.1        eth1        static
10.1.8.0        255.255.252.0   0.0.0.0         eth1        interface
```

show jobs

The **show jobs** command displays command sequences, schedules, and configuration settings for jobs on the Violin Array.

Syntax

```
show jobs [<job-id>]
```

The option limits the display to information about the specified job.

Command Mode

Enable mode, Config mode

Example

The following example shows information about a job configured on the Violin Array.

```
# show jobs 101
Job 101 (Sample Job):
   Status:              inactive
   Enabled:             yes
   Continue on failure: no
   Comment:             Test job
   Schedule Type:       once
   Time and date:       1970/01/01 00:00:00 -0800
   Last Exec Time:      Wed 2018/28/09 19:44:37 -0800
   Next Exec Time:      N/A
   Commands:
      Command 1: show ver
      Command 2: show services small-servers
   Last Output:
```

show ldap

The **show ldap** command displays current settings for LDAP authentication.

Syntax

```
show ldap
```

Command Mode

Enable mode, Config mode

Example

The following example shows information about a job configured on the Violin Array.

```
# show jobs 101
User base DN      : ou=users,dc=example,dc=com
User search scope : subtree
Login attribute   : sAMAccountName
Bind DN           :
Bind password     :
Group base DN     :
Group attribute   : member
LDAP version      : 3
Referrals         : yes
Server port       : 389
Search Timeout    : 5
Bind Timeout      : 5
SSL mode          : none
Server SSL port   : 636 (not active)
SSL cert verify   : yes

No LDAP servers configured.
```

show licenses

The **show licenses** command displays licenses for features installed on the system, along with associated license keys.

Syntax

```
show licenses
```

Command Mode

Enable mode, Config mode

Example

The following example displays the feature licenses installed on the Violin Array.

```
# show licenses
License 1: LK2-RESTRICTED_CMDS-7X87-SAMPLE-LICENSE-KEY
   Feature: RESTRICTED_CMDS
   Valid: yes
   Tied to cluster ID: 99999 (ok)
   Active: yes

License 2: LK2-VSHARE-7X87-SAMPLE-LICENSE-KEY
   Feature: VSHARE
   Valid: yes
   Tied to cluster ID: 99999 (ok)
   Active: yes
```

The following fields are displayed by the command:

`License n:`	The license key for the feature.
`Feature:`	The name of the licensed feature.
`Valid:`	Whether the license is valid.
`Tied to cluster ID:`	The identifier of the cluster to which the feature license is tied.
`Active:`	Whether the license is currently active. If no is displayed, check to make sure the license key was input correctly.

show locate

The **show locate** command shows the status of the ID LED on the front and rear of the Violin Array. The ID LED can be turned on by using the "`locate`" command and by physically pressing the button the Violin Array.

Syntax

```
show locate
```

Command Mode

Enable mode, Config modes

Example

The following example shows the status of the ID LED.

```
# show locate
System locator: ON
```

show log

The **show log** command displays the contents of the current log file.

Syntax

```
show log [not matching <reg-exp>] [matching <reg-exp>]
```

If you include the **matching** or **not matching** parameter, only log lines matching or not matching the specified regular expression are printed. The `<reg-exp>` is an extended regular expression as defined by the grep man page.

Command Mode

Enable mode, Config mode

Example

The following example displays the current log file on a Violin Array.

```
# show log
Nov 30 01:05:30 vcm-c vcm-c oam_bowos[691]: 3335[017845646] INFO
vimm_mon_stats_send_reply(): Stats for VIMM57. Data length: 1389
Nov 30 01:05:03 lab-fil4140-acm-b vutils[6416]: [vvimms_stats.NOTICE]: Received VIMM 48
statistics from vcm-b
Nov 30 01:05:30 vcm-a vcm-a oam_bowos[691]: 6612[017849791] INFO
vimm_mon_stats_send_reply(): Stats for VIMM45. Data length: 1401
Nov 30 01:05:03 lab-fil4140-acm-b vutils[6416]: [vvimms_stats.NOTICE]: Received VIMM 57
statistics from vcm-c
Nov 30 01:05:03 lab-fil4140-acm-b vutils[6416]: [vvimms_stats.NOTICE]: Received VIMM 45
statistics from vcm-a
Nov 30 01:05:30 vcm-d vcm-d oam_bowos[691]: 1640[017843592] INFO
vimm_mon_stats_send_reply(): Stats for VIMM60. Data length: 1390
Nov 30 01:05:03 lab-fil4140-acm-b vutils[6416]: [vvimms_stats.NOTICE]: Received VIMM 60
statistics from vcm-d
Nov 30 01:05:30 vcm-b vcm-b oam_bowos[691]: 4193[017847731] INFO
vimm_mon_stats_send_reply(): Stats for VIMM50. Data length: 1390
Nov 30 01:05:30 vcm-c vcm-c oam_bowos[691]: 3336[017845649] INFO
vimm_mon_stats_send_reply(): Stats for VIMM61. Data length: 1389
[More output follows]
```

show log continuous

The **show log continuous** command displays the last few lines of the current log file, and then continues to display new lines as they come in, until you press CTRL+C.

Syntax

```
show log continuous [not matching <reg-exp>] [matching <reg-exp>]
```

If you include the **matching** or **not matching** parameter, only log lines matching or not matching the specified regular expression are printed. The `<reg-exp>` is an extended regular expression as defined by the grep man page.

Command Mode

Enable mode, Config mode

Example

The following example displays log messages that include the text “xg”.

```
# show log continuous matching cli
Nov 30 02:40:44 hostname-acmb cli[22542]: [cli.NOTICE]: user admin: CLI launched
Nov 30 02:40:49 hostname-acmb cli[22542]: [cli.NOTICE]: user admin: Executing command:
enable
Nov 30 02:40:49 hostname-acmb cli[22542]: [cli.NOTICE]: user admin: Entering enable mode
Nov 30 02:40:55 hostname-acmb cli[22542]: [cli.NOTICE]: user admin: Executing command:
config terminal
Nov 30 02:40:55 hostname-acmb cli[22542]: [cli.NOTICE]: user admin: Entering configuration
mode
Nov 30 03:00:27 hostname-acmb cli[22542]: [cli.NOTICE]: user admin: Executing command:
show log continuous matching cli
```

show log files

The **show log files** command displays the contents of locally archived log files.

Syntax

```
show log files [<log-file-number> [not matching <reg-exp>] [matching <reg-exp>]]
```

Entering the **show log files** command without specifying a log file number displays a list of the archived log files. Specifying a `<log-file-number>` displays the contents of that log file. If you include the **matching** or **not matching** parameter, only log lines in the file that match or do not match the specified regular expression are printed. The `<reg-exp>` is an extended regular expression as defined by the grep man page.

Command Mode

Enable mode, Config mode

Example

The following example displays log messages in archived log file 1.

```
# show log files 1
Nov  8 21:47:34 lab-fil4140-acm-b mdreq[4551]: [mdreq.NOTICE]: Connecting directly to host
acm-a
Nov  8 21:47:34 lab-fil4140-acm-b mdreq[4551]: [mdreq.NOTICE]: session 1: connected to IP
169.254.1.10
Nov  8 21:50:07 vcm-d vcm-d oam_bowos[691]: 0885[000144018] INFO
vtu_log_write_timeout_cb(): -- MARK --
Nov  8 21:52:00 lab-fil4140-acm-b mdreq[5253]: [mdreq.NOTICE]: Connecting directly to host
acm-a
Nov  8 21:52:00 lab-fil4140-acm-b mdreq[5253]: [mdreq.NOTICE]: session 1: connected to IP
169.254.1.10
Nov  8 21:52:00 lab-fil4140-acm-b acmd[3243]: [acmd.NOTICE]: Received mg-a ACM log forward
master mg-a
Nov  8 21:52:01 lab-fil4140-acm-b mgmtd[3105]: [mgmtd.NOTICE]: Configuration changed by
user i:3243-0-0
[More output follows]
```

show logging

The **show logging** command displays configuration settings for the logging feature on the Violin Array.

Syntax

```
show logging [syslog-facility]
```

The `syslog-facility` option displays the configured logging facility.

Command Mode

Enable mode, Config mode

Example

The following example displays the logging configuration for the Violin Array.

```
# show logging
Local logging level: notice
  Override for class storylines: info
  Override for class mgmt-core: notice
  Override for class mgmt-back: notice
  Override for class mgmt-front: notice
Default remote logging level: notice
Remote syslog receiver: 169.254.1.101 (log level: notice)
  Override for class storylines: info
  Override for class mgmt-core: notice
  Override for class mgmt-back: notice
  Override for class mgmt-front: notice
Allow receiving of messages from remote hosts: yes
Allow relaying of messages from remote hosts: no
Number of archived log files to keep: 5
Log rotation size threshold: 300 megabytes
Log format: standard
Subsecond timestamp field: disabled
Levels at which messages are logged:
  CLI commands: notice
```

The following fields are displayed by the command:

`Local logging level`	The minimum severity of log messages to be saved in log files on local persistent storage, set with the **logging local** command.
`Override for class`	Per-class overrides on the logging level, set with the **logging local override class** command. All classes that do not have an override set use the global logging level set with the **logging local** command; any classes that do have an override will do as the override specifies.

`Default remote logging level`	The minimum severity of log messages sent to syslog servers, set with the **logging trap** command.
`Remote syslog receiver`	Remote syslog receiver that receives syslog messages from the ACM. When Splunk forwarding is enabled, the syslog receiver is an internal MG; when Splunk forwarding is disabled, the syslog receiver will be the other ACM. Remote syslog receivers configured in Violin devices are as follows: • 169.254.1.101 for MG-a • 169.254.1.102 for MG-b • 169.254.1.10 for ACM-a • 169.254.1.11 for ACM-b
`Allow receiving of messages from remote hosts`	Whether this system is enabled to receive log messages from another host, set with the **logging receive** command.
`Allow relaying of messages from remote hosts`	Whether the ACM is enabled to relay messages to MG. The value is set as **yes** when the ACM is relaying the logs to the MG. If the ACM is just sending its own logs, then the value is set as **no**.
`Number of archived log files to keep`	The **logging files rotation max-num** command sets how many log files are kept on the system.
`Log rotation size threshold`	The threshold size, which when reached, log files stored on the Violin array are automatically rotated, set with the **logging files rotation criteria** command.
`Log format`	The format for log messages on the system, either standard or WebTrends Enhanced Log File Format (WELF), set with the **logging format** command.
`Subsecond timestamp field`	Whether the “seconds” field is included in log messages, set with the **logging fields seconds** command.
`Levels at which messages are logged`	The severity level at which CLI commands that the user executes are logged, set with the **logging level cli commands** command; and the severity of log messages that get placed in the audit log, set with the **logging level audit mgmt** command.

show logging files upload-auto

The s**how logging files upload-auto** command displays configuration settings for automatic uploading of log files to a remote host.

Syntax

```
show logging files upload-auto [detailed]
```

Command Mode

Enable mode, Config mode

Examples

The following example displays the configuration for automatic log file uploading.

```
# show logging files upload-auto
Enable:              no
Protocol:            https
Remote URL:          support.vmem.com/support/downloads/incoming
Username:            anonymous
User password:       *
```

The **detailed** keyword displays additional information; for example:.

```
# show logging files upload-auto detailed
```

```
Enable:              yes
Protocol:            email
Remote Site:         support.vmem.com
Remote Directory:    incoming/autosupport
Remote URL:          support.vmem.com/support/downloads/
incoming
Username:            anonymous
User password:       *
Email address:       support-logs@vmem.com
Upload throttle:     3 hour(s)
Upload delay:        15 minute(s)
Email split size:    6 MB

ACM only options:
-----------------
Upload interval:     24 hour(s)
Max file size:       20 MB
Include Sysdump:     yes
Include Cache info:  yes
Include Cores:       no
Include MGs:         yes
```

The following fields are displayed by the command. The settings for these fields are controlled with the **logging files upload-auto** command.

`Enable`	Whether automatic gathering and uploading of syslog files is enabled on the system.
`Protocol`	The protocol used for uploading the log files.
`Upload interval`	How often the log files are uploaded.
`Max file size`	The maximum size for log file uploads, in MB.
`Include Sysdump`	Whether system dump files are included with the uploaded logs.
`Include Cache info`	Whether cache information is included with the uploaded logs.
`Include Cores`	Whether core files are included with the uploaded logs.
`Include MGs`	Whether mg log files are included with the uploaded logs.
`Remote Site`	The name of the remote host where log files are uploaded.
`Remote Directory`	The directory on the remote host where the log files will be uploaded.
`Remote URL`	The full URL of the location on the remote host where the logs will be uploaded.

`Upload throttle`	Specifies an upload throttle window that comes into effect after an upload has occurred. The throttle window suppresses further automatic log uploads within this window so that the support servers are not inundated with uploads. The default throttle window is 3 hours and the range is 0-24 hours.
`Upload delay`	Specifies an upload delay that comes into play when a callhome event has been generated. The logs will be automatically uploaded after the specified upload delay. The default value is 15 minutes. The acceptable range is 0-60 minutes.
`Email split size`	Specifies the split size for an email upload bundle. The default is 6 MB with a range value of 0-20 MB.
`Username`	The username used for uploading files to the remote host.
`User password`	The password used for uploading files to the remote host.
`Email address`	The e-mail address where log files are sent, if configured.

show logins

The **show logins** command the users currently logged into the CLI on the Violin device.

Syntax

```
show logins
```

Command Mode

Enable mode, Config mode

Example

The following example displays user login information.

```
# show logins
Username      Line       Host
--------------------------------------
admin         pts/0      169.254.1.11
admin         pts/1      10.12.56.35
```

show memory

The **show memory** command displays information about memory usage on the Violin Array.

Syntax

```
show memory
```

Command Mode

Enable mode, Config mode

Example

The following example displays memory usage information.

```
# show memory
          Total       Used       Free       Used+B/C  Free-B/C
Physical   1961 MB     385 MB    1576 MB     955 MB    1006 MB
Swap          0 MB       0 MB       0 MB

Physical Memory Borrowed for System Buffers and Cache:
  Buffers:                 140 MB
  Cache:                   429 MB
  Total Buffers/Cache:     570 MB
```

show ntp

The **show ntp** command displays NTP status on the Violin Array, along with information about the configured NTP servers and peers.

Syntax

```
show ntp [configured]
```

The **configured** keyword displays the NTP configuration for the Violin Array.

Command Mode

Standard mode, Enable mode, Config mode

Examples

The following example displays current NTP status information.

```
# show ntp
NTP is enabled.
Clock is synchronized.  Reference: 10.1.8.5.  Offset: 2.358 ms.
Active servers and peers:
                                                                  Poll      Last
                                        Offset  Ref               Interval  Response
Address         Status      Stratum   (msec)  Clock             (sec)     (sec)
================================================================================
10.1.8.5        sys.peer (*)    3      2.358  130.126.24.53     1024       597
```

The output of the command indicates whether NTP is enabled on the Violin Array, whether the clock on the Violin Array is synchronized to an NTP server, the NTP server to which the Violin Array is synchronized, and the offset in milliseconds of the Violin Array clock to the NTP server.

For the active NTP servers and peers, the following fields are displayed:

`Address`	The address of the NTP server or peer.
`Status`	Current status of the NTP server or peer. An asterisk (*) indicates that the Violin Array is synchronized to this NTP server.
`Stratum`	The NTP stratum for this NTP server or peer.
`Offset (msec)`	The offset in milliseconds of the Violin Array clock to the NTP server or peer.
`Ref Clock`	The address of the reference clock for the NTP server or peer.
`Poll Interval (sec)`	The polling interval, in seconds, for this NTP server or peer.
`Last Response (sec)`	The time, in seconds, since the last response was received from this NTP server or peer.

The following example displays the NTP configuration on the Violin Array.

```
# show ntp configured
NTP enabled: yes
No NTP peers configured.
NTP server 10.1.8.5
  Enabled: yes
  NTP version: 4
```

show out-of-service

The **show out-of-service** command lists the components of a Violin Array where the `out-of-service` flag has been activated, which indicates the component has failed or has been powered off. The `out-of-service` flag can be activated by the system, or by an administrator entering the **no array modules id** command.

Syntax

```
show out-of-service
```

Command Mode

Enable mode, Config mode

Examples

The following example lists the components that have the `out-of-service` flag activated, then clears the out-of-service flag for one of them.

```
# show out-of-service
id     out-of-service   description
-------------------------------------
mg-a   user-set         disabled via CLI at 21:36:36 06/24/2013
mg-b   user-set         disabled via CLI at 21:36:36 06/24/2013

Summary: 2 components out-of-service detected.

Use 'array modules id <component> enable' to clear user-set out-of-
service.

# conf t
(config) # array modules id mg-a enable
```

show pcie

The **show pcie** command displays PCIe connection information for a Violin Array.

Syntax

```
show pcie [configs] [ports]
```

where:.

`configs`	Displays the current PCIe configuration.
`ports`	Displays status information for ACM ports on the Violin Array.

If you enter the **show pcie** command without options, both config and ports information is shown.

Command Mode

Enable mode, Config mode

Example

The following example displays PCIe configuration and port information for an XVS.

```
# show pcie
Current PCI-E config: mg

Ports acm-a (acm-a):
  mg_out: optimal
   hba-a: optimal
   vcm-a: optimal
   vcm-c: optimal
  mg_in: optimal
   hba-b: optimal
   vcm-b: optimal
   vcm-d: optimal
  mg_interconnect: optimal
   clustering: optimal
   interconnect: optimal

Ports acm-b (acm-b):
  mg_out: optimal
   hba-c: optimal
   vcm-a: optimal
   vcm-c: optimal
  mg_in: optimal
   hba-d: optimal
   vcm-b: optimal
   vcm-d: optimal
  mg_interconnect: optimal
   clustering: optimal
   interconnect: optimal
```

Example

The following example displays PCIe configuration and port information for an XVS.

```
# show pcie
Current PCI-E config: mg

Ports acm-a (lab-fil4140-acm-a):
  mg_out: optimal
   hba-a: optimal
   vcm-a: optimal
   vcm-c: optimal
  mg_in: optimal
   hba-b: optimal
   vcm-b: optimal
   vcm-d: optimal
  mg_interconnect: optimal
   clustering: optimal
   interconnect: optimal

Ports acm-b (lab-fil4140-acm-b):
  mg_out: optimal
   hba-c: optimal
   vcm-a: optimal
   vcm-c: optimal
  mg_in: optimal
   hba-d: optimal
   vcm-b: optimal
   vcm-d: optimal
  mg_interconnect: optimal
   clustering: optimal
   interconnect: optimal
```

show running-config

The **show running-config** command displays the running configuration on the screen. This command is functionally equivalent to the **write terminal** command.

Syntax

```
show running-config [full]
```

The **full** option includes commands that set default values in the display.

Command Mode

Enable mode, Config mode

Example

The following example displays the running configuration.

```
(config) # show running-config
```

show services small-servers

The **show services small-servers** command displays the status of the TCP/UDP “small server” services: `echo`, `chargen`, `discard`, `daytime`, and `time`.

Syntax

```
show services small-servers
```

Command Mode

Enable mode, Config mode

Example

The following example displays the status of the TCP/UDP “small server” services.

```
# show services small-servers
Service   TCP          UDP
-------   ---          ---
echo      disabled     disabled
chargen   disabled     disabled
discard   disabled     disabled
daytime   disabled     disabled
time      disabled     disabled
```

show snmp

The **show snmp** command displays information about the SNMP configuration on the Violin Array.

Syntax

```
show snmp
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the SNMP settings on the Violin Array.

```
# show snmp
SNMP enabled:          yes
SNMP port:             161
System contact:
System location:        MVII

Read-only communities:
   public

Interface listen enabled: yes
No Listen Interfaces.

Traps enabled:             yes
Default trap community:    public
Default trap port:         162

Trap sinks:
   10.1.4.135
      Enabled: yes
      Type: traps version 2c
      Port: 162 (default)
      Community: public (default)
```

The following fields are displayed by the command:

`SNMP enabled`	Whether SNMP is enabled on the Violin Array, set with the **snmp-server enable** command.
`SNMP port`	The UDP port for the SNMP agent, set with the **snmp-server port** command (default 161).
`System contact`	The contents of the syscontact variable served from the System MIB in MIB-II, set with the **snmp-server contact** command.

`System location`	The contents of the syslocation variable served from the System MIB in MIB-II, set with the **snmp-server location** command.
`Read-only communities`	Communities specified as read-only, allowing management stations to retrieve, but not modify, MIB objects on the Violin Array, set with the **ro** option in the **snmp-server community** command.
`Interface listen enabled`	Whether the Violin Array is configured to accept SNMP connections only on specific interfaces, set with the **snmp-server listen enable** and **snmp-server listen interface** commands. If there are non-DHCP interfaces in the listen list, SNMP connections are only accepted on interfaces in the list. Otherwise, SNMP connections are accepted on any interface.
`Traps enabled`	Whether the Violin Array is configured to send SNMP traps to hosts (trap sinks) specified to receive them, set with the **snmp-server traps** command.
`Default trap community`	The default community string for SNMP traps, set with the **snmp-server traps community** command. This setting applies to SNMP traps sent to hosts that do not have a custom community string set.
`Default trap port`	The default UDP port to which SNMP traps are sent, set with the **snmp-server traps port** command (default 162).
`Trap sinks`	Settings for the hosts that will receive SNMP traps from the Violin Array, set with the **snmp-server host** command.

show snmp engineID

The **show snmp engineID** command displays the value of the local SNMPv3 engine ID. The local engine ID is used in conjunction with SNMPv3 user passwords to generate authentication and encryption keys for SNMPv3 users.

Syntax

```
show snmp engineID
```

Command Mode

Enable mode, Config mode

Example

The following example displays the local SNMPv3 engine ID on the Violin Array.

```
# show snmp engineID
Local SNMP engineID: 0x80008c3123456789306533396464626464
```

show snmp user

The **show snmp user** command displays information about SNMPv3 users configured on the Violin Array.

Syntax

```
show snmp user
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about SNMPv3 users on the Violin Array.

```
# show snmp user
User name: admin
   Enabled overall:          no
   Authentication type:      sha
   Privacy type:             aes-128
   Authentication password:  (NOT SET; user disabled)
   Privacy password:         (NOT SET; user disabled)
```

For each SNMPv3 user, the following fields are displayed by the command. These settings are configured with the **snmp-server user v3** command.

`User name`	The name of the SNMPv3 user.
`Enabled overall`	Whether SNMPv3 access is enabled for the user.
`Authentication type`	The hash algorithm to be used for authentication of the SNMPv3 user.
`Privacy type`	The privacy encryption type.
`Authentication password`	Encrypted version of the SNMPv3 user's password, if configured.
`Privacy password`	Encrypted version of the SNMPv3 user's privacy password, if configured. If not set, then the authentication password is used as the privacy password.

show ssh client

The **show ssh client** command displays information about the configuration for SSH clients and keys for local user accounts.

Syntax

```
show ssh client
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the SSH client configuration.

```
# show ssh client
SSH client Strict Hostkey Checking: ask

SSH Global Known Hosts:
    Entry 1: vmg01
         Finger Print: b0:52:ac:29:ca:b1:2a:14:10:8b:9b:ca:4f:56:e4:e7

User Identities:
  User oboe:
    DSAv2 Public key:
ssh-dss AAAAB3NzaC1kc3MAAACBALDWBXpkmiyl76GD6BdPYizfmXpsZ8qguFKVIAcC2
EVABQILUPWTMkDJw9vNdZ6723jDTFOMhyq8V+peqH1vYmN5rIoySkneOI1nRqnCMlHGJh
3dWI+FtjV78EmyWGAs37lBAzFn7O2YxjziWjxe6N1bpI0MuIKtMDUCBtuItitiP09uSXA
GQ48Bi8weRFuv29E9fEGi5Bw4O1pDhzsnuI5wh39XOuSix3LnqK7dZsb5L2jqkDLhkc=

    DSAv2 Private key:
********
    Passphrase:
********
   RSAv2 Public key:
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA3EuIwjkvsxnS+yDfrJUgppHEp0IgHVe
0FbmpLulD55/qbEyYYOYxat/tmmdv0GK29Bqlnr/92hWoPzo+GLcFMrnph8ZJNZ81Uc

    RSAv2 Private key:
********
    Passphrase:
********
SSH authorized keys:
  User oboe:
  No authorized keys for user oboe.
```

The following fields are displayed by the command:

`SSH client Strict Hostkey Checking`	How the system checks connecting SSH clients against its known hosts file, set with the **ssh client global host-key-check** command. This can be one of the following:f `yes` Clients are permitted to connect only if a matching host key is already in the system's known hosts file. `no` Clients are permitted to connect always, and any new or changed host keys are accepted without checking the system's known hosts file. `ask` The system prompts the user to accept new host keys, but does not permit a connection if there is already an entry in the known hosts file that does not match the one presented by the host. This is the default behavior.
`SSH Global Known Hosts`	The entries in the system's global known hosts file, set with the **ssh client global known-host** command.
`User Identities`	The DSAv2 and RSAv2 keys for each user, if configured, set with the **ssh client user identity** command.
`SSH authorized keys`	the list of authorized public keys for each user, if any, set with the **ssh client user authorized-key** command.

show ssh server

The **show ssh server** command displays information about the configuration for the SSH server and host keys.

Syntax

```
show ssh server [host-keys]
```

The **host-keys** keyword displays the RSAv1, RSAv2, and DSAv2 host keys set or generated for the system.

Command Mode

Enable mode, Config mode

Example

The following example displays information about the SSH server configuration.

```
# show ssh server
SSH server configuration:
   SSH server enabled:       yes
   Minimum protocol version: 2
   X11 forwarding enabled:   no
   SSH server ports:         22

   Interface listen enabled: yes
   No Listen Interfaces.

Host Key Finger Prints:
    RSA v1 host key: 88:9c:56:b2:c4:6a:8d:81:d5:a2:ff:9d:6c:06:93:a5
    RSA v2 host key: 4a:a2:2f:a7:20:59:71:b6:55:74:af:5f:36:eb:85:08
    DSA v2 host key: cf:5d:db:5f:ea:b3:46:cc:9b:51:9f:9b:25:47:75:89
```

The following fields are displayed by the command:

`SSH server enabled`	Whether the SSH server is enabled on the system, set with the **ssh server enable** command.
`Minimum protocol version`	The minimum SSH version for SSH client connections, either 1 or 2, set with the **ssh server min-version** command.
`X11 forwarding enabled`	Whether X11 forwarding is enabled for the SSH server, set with the **ssh server x11-forwarding** command.
`SSH server ports`	The list of ports on which the system accepts SSH client connections, set with the **ssh server ports** command.
`Interface listen enabled`	Whether the system is configured to listen for SSH client connections only on specific interfaces, as well as the list of those interfaces, set with the **ssh server listen** command. If there are no interfaces in the list, then the system listens for SSH connections on all interfaces.
`Host Key Finger Prints`	The fingerprints of the RSAv1, RSAv2, and DSAv2 host keys set or generated for the system. To display the actual host keys, use the **show ssh server host-keys** command.

show stats alarm

The **show stats alarm** command displays the status and configuration settings for statistics-based alarms on the Violin Array.

Syntax

```
show stats alarm [<alarm-id> [rate-limit]]
```

If you specify an `<alarm-id>`, the configuration settings for the alarm are displayed. The **rate-limit** option displays rate limiting settings and statistics for the alarm.

Command Mode

Enable mode, Config mode

Examples

The following example displays the state of alarms configured on the system.

```
# show stats alarm
Alarm chassis_temp:                                          ok
Alarm conntrack_entries:                                     ok
Alarm cpu_util_indiv (Average CPU utilization too high):     ok
Alarm disk_io (Operating System Disk I/O too high):          (disabled)
Alarm fs_mnt (Free filesystem space too low):                ok
Alarm intf_util (Network utilization too high):              (disabled)
Alarm lid_ajar_time:                                         ok
Alarm memory_pct_used (Too much memory in use):              (disabled)
Alarm paging (Paging activity too high):                     ok
Alarm vimm_temp:                                             ok
```

The following example displays the settings and status for the `chassis_temp` alarm.

```
# show stats alarm chassis_temp
Alarm chassis_temp:
    Enabled:                    yes
    Alarm state:                ok
    Rising error threshold:     75
    Rising clear threshold:     70
    Rate limiting:
        Events skipped:         0 (since last event)
        Short window:           1 alarms in 1 hour
        Medium window:          20 alarms in 1 day
        Long window:            50 alarms in 7 days
    Event repetition:           single

    Current time:               2018/11/30 02:01:04
    Last event time:
```

The following fields are displayed by the command:

`Enabled`	Whether the alarm is enabled.
`Alarm state`	The current state of the alarm
`Rising error threshold`	Threshold value that triggers the alarm when the statistic rises above the threshold.
`Rising clear threshold`	Threshold value that clears the alarm when the statistic drops below the threshold.
`Rate limiting`	Alarm rate limiting settings, set with the **stats alarm rate-limit** command.
`Event repetition`	Either `single` (generate an alarm event only when the alarm changes state), or `while-not-cleared` (generate periodic alarm events while the alarm is in an error state)

show stats chd

The **show stats chd** command displays configuration settings for computed historical datapoint (CHD) datasets.

Syntax

```
show stats chd [<chd-id>]
```

If you specify a `<chd-id>`, the configuration settings for the specified CHD dataset are displayed.

Command Mode

Enable mode, Config mode

Example

The following example displays information about the `writes_hour` CHD dataset.

```
# show stats chd writes_hour
CHD "writes_hour" (Number of NFS writes in last interval):
   Enabled:           yes
   Source dataset:    sample "writes"
   Computation basis: time
   Interval:          30 second(s)
   Range:             30 second(s)
```

The following fields are displayed by the command:

`Enabled`	Whether the CHD is enabled.
`Source dataset`	The source sample dataset the CHD uses to perform calculations.
`Computation basis`	The method the CHD uses to determine when to perform a new calculation. The value `data-points` means that the determination is based on appearance of new data points in the source dataset, and the calculation is always done over a fixed number of data points. The value `time` means that the determination is based on the passage of a fixed time interval, and the calculation is always done over a fixed time interval.
`Interval`	How often a new calculation is performed. The CHD performs a new calculation every time the number of seconds specified for the interval elapses.
`Range`	The range of datapoints that are used in the CHD calculation. The inputs for the calculation are the datapoints in the source dataset whose timestamps fall within the most recent range.

show stats cpu

The **show stats cpu** command displays utilization statistics for the CPUs on the Violin Array, including the current utilization level for each CPU, the peak over the past hour, and the average over the last hour. To clear this data, enter the **stats chd cpu_util clear** command.

Syntax

```
show stats cpu
```

Command Mode

Enable mode, Config mode

Example

The following example displays CPU utilization statistics for the Violin Array.

```
# show stats cpu

CPU 0
  Utilization:                14%
  Peak Utilization Last Hour: 71% at 2018/11/30 01:39:53
  Avg. Utilization Last Hour: 15%

CPU 1
  Utilization:                14%
  Peak Utilization Last Hour: 37% at 2018/11/30 01:39:38
  Avg. Utilization Last Hour: 13%
```

show stats sample

The **show stats sample** command displays the enabled/disabled status and sampling interval for sample datasets on the Violin Array.

Syntax

```
show stats sample [<sample-id>]
```

If you specify a `<sample-id>`, the configuration settings for the specified sample dataset are displayed.

Command Mode

Enable mode, Config mode

Example

The following example displays information about the `writes` sample dataset.

```
# show stats sample writes
Sample "writes" (Number of NFS writes in last interval):
   Enabled:           yes
   Sampling interval: 10 seconds
```

show story boot

The **show story boot** command displays system and specific module boot information.

Syntax

```
show story boot <system | acm | acm-a | acm-b |vcm | vcm-a | vcm-b |
vcm-c | vcm-d | vimm | vimmXX | fail>
```

where:

`system`	System boot stories including ACM, VCM, VIMM.
`acm`	All ACM boot stories including VCM and VIMM.
`acm-a`	ACM-a boot stories.
`acm-b`	ACM-b boot stories.
`vcm`	All VCM boot stories including VIMM.
`vcm-a`	VCM-a boot stories.
`vcm-b`	VCM-b boot stories.
`vcm-c`	VCM-c boot stories.
`vcm-d`	VCM-d boot stories.
`vimm`	All VIMM boot stories.
`vimmXX`	vimm*XX* boot stories.
`fail`	Display all failed boot stories.

Command Mode

Enable mode, Config mode

Example

The following is partial output for a boot story for a specific VIMM.

```
# show story boot vimm12
Nov  8 21:56:36  VIMM 12: belongs to RAID Group 6
Nov  8 21:58:00  VIMM 12: belongs to RAID Group 6
Nov  8 22:00:01  VIMM 12: belongs to RAID Group 6
Nov  8 22:01:28  VIMM 12: belongs to RAID Group 6
Nov  8 22:03:21  BEGIN: VIMM 12: is booting
Nov  8 22:03:26  VIMM 12: config EEPROM cache initialized
Nov  8 22:03:27  VIMM 12: finished sw-scan
Nov  8 22:03:28  VIMM 12: is of type 2048GiB, MLC-NAND, formatted
capacity = 1721GiB
Nov  8 22:03:28  VIMM 12: belongs to RAID Group 6
Nov  8 22:04:52  BEGIN: VIMM 12: is booting
Nov  8 22:04:59  VIMM 12: config EEPROM cache initialized
Nov  8 22:05:00  VIMM 12: finished sw-scan
...
```

show story continuous

The **show story continuous** command continuously displays tail of storylines file.

Syntax

```
show story continuous <boot | fault | format | recovery | upgrade>
```

where:

`boot`	Continuously display boot stories.
`fault`	Continuously display fault stories.
`format`	Continuously display format stories.
`recovery`	Continuously display recovery stories.
`upgrade`	Continuously display upgrade stories.

Command Mode

Enable mode, Config mode

show story fail

The **show story fail** command displays the stories for all failures.

Syntax

```
show story fail <summary>
```

where:

`summary`	Display summary of all failed stories.

Command Mode

Enable mode, Config mode

Example

The following is an example of the `show story fail summary` command.

```
# show story fail summary

Failed
  ACM boot          : 1
  VCM boot          : 19
  format            : 14
  RAID rebuild      : 64

Total stories: 98
```

show story fault

The **show story fault** command displays fault information for array modules.

Syntax

```
show story fault <acm | acm-a | acm-b | vcm |vcm-a | vcm-b | vcm-c |
vcm-d | vimm | vimmXX | pcm | pcm-a | pcm-b | fan | fan-a1 | fan-b1 | fan-
c1 | fan-a0 | fan-b0 | fan-c0 | fail>
```

where:

`acm`	All ACM fault stories.
`acm-a`	ACM-a fault stories.
`acm-b`	ACM-b fault stories.
`vcm`	All VCM fault stories.
`vcm-a`	VCM-a fault stories.
`vcm-b`	VCM-b fault stories.
`vcm-c`	VCM-c fault stories.
`vcm-d`	VCM-d fault stories.
`vimm`	All VIMM fault stories.
`vimmXX`	vimm*XX* fault stories.
`pcm`	All PCM fault stories.
`pcm-a`	PCM-a fault stories.
`pcm-b`	PCM-b fault stories.
`fan`	All fan fault stories.
`fan-a1`	Fan-a1 fault stories.
`fan-b1`	Fan-b1 fault stories.
`fan-c1`	Fan-c1 fault stories.
`fan-a0`	Fan-a0 fault stories.
`fan-b0`	Fan-b0 fault stories.
`fan-c0`	Fan-c0 fault stories.
`fail`	Display all failed fault stories.

Command Mode

Enable mode, Config mode

Example

The following is partial output for a `show story fault vimm` command.

```
# show story fault vimm
Nov 21 05:04:36  VIMF:BEGIN: A failure is detected on VIMM  4
Nov 21 05:04:36  VIMF:Major failure: Fatal CPL IRQ [0x0000000590001000
(0x00321d69, 0x00008935)]: CRAM_ERR,
Nov 21 05:04:36  VIMF:Dumping VIMM control registers to log
Nov 21 05:04:36  VIMF:**ALARM** VIMM  4 Fatal CPL IRQ
[0x0000000590001000 (0x00321d69, 0x00008935)]: CRAM_ERR,
Nov 21 05:04:36  VIMF:Preparing to take VIMM  4 out of service
Nov 21 05:04:37  VIMF:Deactivating VIMM  4
Nov 21 05:04:38  VIMF:SUCCESS: Failure handling COMPLETE
```

show story format

The **show story format** command displays the sequence of events that occur when an "`array format`" command is issued from the CLI.

Syntax

```
show story format <fail>
```

where:

`fail`	Display failed format stories.

Command Mode

Enable mode, Config mode

Example

The following is partial output for a `show story format` command.

```
# show story format
Apr 29 15:53:27  FMTO:BEGIN: Starting array format operation
Apr 29 15:53:27  FMTO:Checking user confirmation to proceed with the
format operation
Apr 29 15:53:28  FMTO:Checking if system is ready for formatting
Apr 29 15:53:29  FMTO:FAILURE: Pre-format checks failed because Array
is in use by MG/Host, format request is rejected. Please power off MG
or direct connected Host and try again.
Apr 29 15:53:53  FMTO:BEGIN: Starting array format operation
Apr 29 15:53:53  FMTO:Checking user confirmation to proceed with the
format operation
Apr 29 15:53:54  FMTO:Checking if system is ready for formatting
Apr 29 15:54:09  FMTO:Sending format action to VCM(s)
Apr 29 15:54:19  FMTO:Formatting of array in progress
Apr 29 18:22:11  FMTO:SUCCESS: Array format operation SUCCEEDED
...
```

show story recovery

The **show story recovery** command displays RAID rebuild and recovery information.

Syntax

```
show story recovery <rebuild | fail>
```

where:

`rebuild`	RAID rebuild stories.
`fail`	Display all failed recovery stories.

Command Mode

Enable mode, Config mode

Example

The following is partial output for a `show story recovery rebuild` command.

```
# show story recovery rebuild
Nov 21 05:05:10  BEGIN: Starting full RAID rebuild operation for group
2, VIMM  3
Nov 21 05:08:05  SUCCESS: Canceled full RAID rebuild of group 2 VIMM  3
Nov 21 05:08:06  BEGIN: Starting adv-delta RAID rebuild operation for
group 2, VIMM  4
Nov 21 05:08:06  SUCCESS: adv-delta RAID rebuild of VIMM  4 in group 2
SUCCEEDED
```

show story summary

The **show story summary** command displays a summary of all stories.

Syntax

```
show story summary
```

Command Mode

Enable mode, Config mode

Example

The following is an example of the `show story summary` command.

```
# show story summary
Failed
  ACM boot          : 1
  VCM boot          : 19
  format            : 14
  RAID rebuild      : 64

Successful
  System boot       : 210
  ACM boot          : 121
  VCM boot          : 63
  VIMM boot         : 2667
  format            : 8
  VCM fault         : 56
  VIMM fault        : 288
  RAID rebuild      : 484
  Cluster upgrade   : 74
  ACM upgrade       : 23
  VCM upgrade       : 24

Total stories: 4116
```

show story upgrade

The **show story upgrade** command displays cluster upgrade, ACM and VCM upgrade information.

Syntax

```
show story upgrade <cluster | acm | acm-a | acm-b |vcm | vcm-a | vcm-b |
vcm-c | vcm-d | fail>
```

where:

`cluster`	Cluster upgrade stories including ACM.
`acm`	All ACM upgrade stories including VCM.
`acm-a`	ACM-a upgrade stories.
`acm-b`	ACM-b upgrade stories.
`vcm`	All VCM upgrade stories including VIMM.
`vcm-a`	VCM-a upgrade stories.
`vcm-b`	VCM-b upgrade stories.
`vcm-c`	VCM-c upgrade stories.
`vcm-d`	VCM-d upgrade stories.
`fail`	Display all failed upgrade stories.

Command Mode

Enable mode, Config mode

Example

The following is an example of the `show story upgrade` command for VCM A.

```
# show story upgrade vcm-a
Apr 28 22:15:51  Starting upgrade of VCM(s)
Apr 28 22:15:51  Starting VCM upgrade
Apr 28 22:16:02  Launching VCM upgrade script
Apr 28 22:16:02  Checking pre-upgrade conditions for VCM
Apr 28 22:16:03  Copying upgrade image to VCM
Apr 28 22:16:33  Verifying validity of upgrade image on VCM
Apr 28 17:29:50  VCMU:BEGIN: Starting upgrade of VCM(s)
Apr 28 17:29:50  VCMU:VCM-a: Starting VCM upgrade
Apr 28 17:29:50  VCMU:BEGIN: Starting upgrade of VCM(s)
Apr 28 17:29:50  VCMU:VCM-a: Starting VCM upgrade
Apr 28 17:30:08  VCMU:VCM-a: Launching VCM upgrade script
Apr 28 17:30:08  VCMU:VCM-a: Launching VCM upgrade script
Apr 28 17:30:08  VCMU:VCM-a: Checking pre-upgrade conditions for VCM
Apr 28 17:30:08  VCMU:VCM-a: Checking pre-upgrade conditions for VCM
Apr 28 17:30:09  VCMU:VCM-a: Copying upgrade image to VCM
Apr 28 17:30:09  VCMU:VCM-a: Copying upgrade image to VCM
Apr 28 17:30:43  VCMU:VCM-a: Verifying validity of upgrade image on VCM
Apr 28 17:30:43  VCMU:VCM-a: Verifying validity of upgrade image on VCM
```

show system alarms

The **show system alarms** command displays the active system alarms on a Violin Array.

Syntax

```
show system alarms
```

Command Mode

Enable mode, Config mode

Examples

The following example displays the system alarms for the Violin Array.

```
# show system alarms
--- show alarm for VMA01 at Wed Nov 28 15:54:11 2018

No ACM alarms
No FPM alarms
vcm-a
    No alarms
vcm-b
    No alarms
vcm-c:
Temp unreadable, last at 44 C
System booting (0.0% complete)
Data plane disabled
Scheduler paused
Port 1 (x4) negotiated to 0 lanes
Port 2 (x4) negotiated to 0 lanes
vcm-d
    No alarms
No VIMM alarms
No FPM alarms
No MG alarms.
No HBA alarms
No PCM alarms
No PSU alarms
No FAN alarms
No RAID alarms
```

show telnet-server

The **show telnet-server** command indicates whether the Telnet server has been enabled on the Violin Array.

Syntax

```
show telnet-server
```

Command Mode

Enable mode, Config mode

Example

The following example displays the status of the Telnet server.

```
# show telnet-server
Telnet server enabled:  no
```

show terminal

The **show terminal** command displays information about the terminal settings for the current CLI session.

Syntax

```
show terminal
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the terminal settings for the current CLI session. These settings are controlled with the **cli terminal session** command.

```
# show terminal
CLI current session settings:
  Terminal width:           104 columns
  Terminal length:          24 rows
  Terminal type:            xterm
  X display setting:        (none)
```

show upgrade

The **show upgrade** command displays upgrade configuration status for VIMMs and VCMs. VIMMs can be included or excluded from a staged upgrade process using the `[no] array upgrade staged vimms` command.

Syntax

```
show upgrade
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the upgrade configuration status for VIMMs and VCMs.

```
# show upgrade
Upgrade Parameter Information.
 - Enable staged upgrade of VIMMs: False
 - VIMM upgrade pending: False
 - Skip version check enabled for VCMs and VIMM upgrades.
```

show usernames

The **show usernames** command lists the local user accounts configured on the system and displays information about the capabilities and status for each account.

Syntax

```
show usernames
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the configured local user accounts.

```
# show usernames
USERNAME     FULL NAME                CAPABILITY  ACCOUNT STATUS
aaa                                   monitor     Account disabled
admin        System Administrator     admin       No password required for login
monitor      System Monitor           monitor     No password required for login
oboe                                  admin       Password set
```

The following fields are displayed by the command:

`USERNAME`	The user ID for the account.
`FULL NAME`	The full name for the account, if any; configured with the **username full-name** command.
`CAPABILITY`	The capability for the user account. This can be one of the following: `admin` Can access all data and perform all configuration tasks, including modifying configuration files. User accounts have the admin capability by default. `monitor` Can read all data and perform all actions, but cannot modify the configuration. `unpriv` Has access to Standard command mode only.
`ACCOUNT STATUS`	Whether the account has been administratively disabled and the status of the password for the account.

show users

The **show users** command lists information about the users currently logged into the system.

Syntax

```
show users
```

Command Mode

All modes

Example

The following example displays information about the users currently logged in to the system.

```
# show users
USERNAME    FULL NAME                LINE    HOST              IDLE
admin       System Administrator     pts/0   10.11.31.10       0d  0h  0m  0s
admin       System Administrator     web/1   10.1.4.100        0d 22h 47m  2s
```

The following fields are displayed by the command:

`USERNAME`	The user ID for the account.
`FULL NAME`	The full name for the account, if any; configured with the **username full-name** command.
`LINE`	The console (`tty`), pseudo-terminal (`pts`), or Web interface (`web`) where the user is logged in.
`HOST`	IP address or hostname of the host the user logged in from.
`IDLE`	The amount of time since the user last entered a command.

show users history

The **show users history** command displays the user login history for the Violin Array.

Syntax

```
show users history [username <userid>]
```

where:

`username <userid>`	Displays the login history for the specified username.

Command Mode

Enable mode, Config mode

Example

The following example lists the login history for the system. For each login, the command displays the username, line, originating host, login time, and how long the user was logged in.

```
# show users history
admin    pts/49       10.12.56.31      Fri Nov 30 01:51   still logged in
admin    pts/46       10.12.56.31      Fri Nov 30 01:49   still logged in
admin    pts/46       10.12.56.31      Fri Nov 30 00:15 - 01:49  (01:33)
admin    pts/46       10.12.56.31      Thu Nov 29 22:17 - 23:45  (01:28)
admin    pts/46       10.12.56.31      Thu Nov 29 21:04 - 22:16  (01:11)
admin    pts/49       10.12.145.50     Thu Nov 29 15:04 - 15:04  (00:00)
admin    pts/49       10.12.145.50     Thu Nov 29 11:27 - 11:49  (00:22)
```

show version

The **show version** command displays information about the system software running on the Violin Array.

Syntax

```
show version [concise | detail]
```

The concise option displays a single line of output showing the software version number and build date. By default, the command displays detailed output.

Command Mode

Enable mode, Config mode

Example

The following example displays information about the system software.

```
# show version detail
Product name:       supervisor
Product release:    A7.5.0.1
Build ID:           #2
Build date:         2018-10-23 19:51:09
Target arch:        ppc
Target hw:          acm
Built by:           root@ci-ginkgo1

Uptime:             14d 21h 51m 46.816s

Product model:      acm32gb
Product hw:         acm
Host ID:            cdd6ff9dd5c4
Serial number:      1S520F00020
```

show vimms

The **show vimms** command displays status, alarm, and inventory information about VIMMs installed on a Violin Array.

Syntax

```
show vimms [id <module-id>] [version] [detail] [summary]
```

where:

`id <module-id>`	Displays information about a specific VIMM.
`version`	Displays the software version of the installed VIMMs.
`summary`	Displays statistics for the VIMMs installed in the Violin Array.
`detail`	Displays additional information about VIMM(s), including power consumption, serial number, and FPGA and software versions.

Command Mode

Enable mode, Config mode

Examples

The following example displays information about the VIMMs installed in the Violin Array.

```
# show vimms

ViMM   VCM     RG   Type      OOS   Status      Temp (C)   %-FmtCap   %-DieFail   %-BlkFail   %-BlkEraAvg   %-LifeTimeRem
00     vcm-a   0    2T-MLC-Flash   no    Active      40         84         0.00        1.33        0.00          100.00
01             1    2T-MLC-Flash   no    Active      43         84         0.00        1.18        0.00          100.00
02             0    2T-MLC-Flash   no    Active      42         84         0.00        1.42        0.00          100.00
03             s    2T-MLC-Flash   no    Spare(F)    42         0          0.00        1.17        0.00          100.00

[more output follows]
```

The following fields are displayed by the command:

VIMM	The ID of the VIMM.
VCM	The vRAID Controller Module (VCM) that manages the VIMM.
RG	The RAID group to which the VIMM belongs.
Type	The type of VIMM.
OOS	Whether the out-of-service flag has been activated for the VIMM.
Status	The status of the VIMM: active, spare, or VIMM not present in the slot.
Temp(C)	Temperature of the VIMM
%-FmtCap	The formatted storage capacity percentage for the VIMM, which can be set with the **array format capacity** command. A single level cell (SLC) system is formatted to 65%, and a multi-level cell (MLC) system is formatted to 84%.
%-DieFail	The percentage of the die on the VIMM that has failed.
%-BlkFail	The percentage of blocks on the VIMM that have failed.
%-BlkEraAvg	The block erasure percentage for the VIMM.
%-LifeTimeRem	The percentage of the VIMM's total lifetime remaining.

The **summary** option displays statistics about all of the VIMMs in the Violin Array. For example:

```
# show vimms summary
number of vimms:          64
healthy:                  64
type:                     2T-MLC-Flash
active:                   60
spare:                    4
boot:                     0
admin down:               0
failed:                   0
out-of-service:           0
alarmed:                  0
temperature:              49/61 (max temp vimm 59)
```

The following example displays additional information about a specific VIMM, including power consumption, serial number, and FPGA and software versions.

```
# show vimms id vimm55 detail
VIMM55:
   VCM                    : vcm-c
   Type                   : 2T-MLC-Flash
   Status                 : Active
   Present                : yes
   Power                  : yes
   Out-of-Service         : no
   Current (mA)           : 1210.41
   RAID                   : 6
   Spare                  : no
   Health                 : health threshold: (OK)
   Temp (C)               : 60
   Serial                 : 2N628F00104
   Model                  : 410-0344-00_R01
   Date                   : 20170601
   FpgaVersion            : 7.1.2.3
   SwVersion              : 7.2.0.0
   %-Format Capacity      : 84
   %-DieFail              : 0.00
   %-BlkFail              : 1.23
   %-BlkEraseAvg          : 1.48
   %-LifeTimeRemaining    : 98.52
     Bytes read           : 3,246,697,271,296
     Bytes written        : 3,687,711,604,736
     ECC Corrected        : 116 (rate: 0.00e+00)
```

The following example displays the firmware version for the installed VIMMs.

```
# show vimms version summary
64 VIMMs at version 7.1.2.3: 00 01 02 03 04 05 06 07 08 09 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
62 63
```

The following example displays detailed information about the VIMM firmware.

```
# show vimms version detail
Vimm    Firmware Version ( Date )                    Control Plane Version ( Date )
-------------------------------------------------------------------------------------
00      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
01      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
02      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
03      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
04      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
05      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
06      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
07      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
08      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
09      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
10      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
11      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
12      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
13      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
14      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
15      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
16      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
17      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
18      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
19      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
20      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
21      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
22      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
23      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
24      7.1.2.3     ( Sep  7 14:33:30 2018 )        7.2.0.0     ( Sep  7 14:33:30 2018 )
...
```

show web

The **show web** command displays the settings for the Violin Web interface.

Syntax

```
show web
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the Violin Web interface configuration.

```
# show web
Web User Interface:
   Web interface enabled:  yes
   HTTP enabled:           no
   HTTP port:              80
   HTTP redirect to HTTPS: no
   HTTPS enabled:          yes
   HTTPS port:             443
   HTTPS certificate name: default-cert
   SSL min-version:        tls1.2
   Listen enabled:         yes
   No Listen Interfaces.

   Inactivity timeout:     15 min
   Session timeout:        2 hr 30 min
   Session renewal:        30 min

Web file transfer proxy:
   Proxy enabled: no

Web file transfer certificate authority:
  HTTPS server cert verify: yes
  HTTPS supplemental CA list: default-ca-list
```

The following fields are displayed by the command:

`Web interface enabled:`	Whether the Violin Web Interface is enabled on the device, set with the **web enable** command.
`HTTP enabled:`	Whether HTTP access to the Violin Web Interface is enabled, set with the **web http enable** command.

`HTTP port:`	The TCP port used for HTTP access to the Violin Web Interface, set with the **web http port** command; default is port 80.
`HTTP redirect to HTTPS:`	Whether HTTP requests for the Violin Web interface are redirected to HTTPS, set with the **web http redirect** command.
`HTTPS enabled:`	Whether HTTPS access to the Violin Web Interface is enabled, set with the **web https enable** command.
`HTTPS port:`	The TCP port used for HTTP access to the Violin Web Interface, set with the **web https port** command; default is port 443.
`HTTPS certificate name:`	The name of HTTPS certificate required to access the Violin Web Interface.
`SSL min-version:`	The set version of SSL protocol, set with **web httpd ssl min-version** command.
`Listen enabled:`	Whether the system is configured to listen for HTTP connections only on specific interfaces, as well as the list of those interfaces, set with the **web httpd listen** command. If there are no interfaces in the list, then the system listens for HTTP connections on all interfaces.
`Inactivity timeout:`	The amount of idle time allowed before a user is logged out of the Violin Web Interface, set with the **web auto-logout** command.
`Session timeout:`	The length of time before a Web session expires, set with the **web session timeout** command.
`Session renewal:`	The length of time before Web session cookies are automatically regenerated, set with the **web session renewal** command.
`Proxy enabled:`	Whether a Web file transfer proxy connection has been enabled, as well as the settings for the Web proxy connection, set with the **web proxy** and **web proxy auth** commands.
`HTTPS server cert verify:`	Whether verification of server certificates during HTTPS file transfers is enabled, set with the **web client cert-verify** command.
`HTTPS supplemental CA list:`	Supplemental CA certificates for verification of server certificates during HTTPS file transfers, set with the **web client ca-list** command.

show whoami

The **show whoami** command displays the username of the currently logged-in user, and the capabilities that user has.

Syntax

```
show whoami
```

Command Mode

Enable mode, Config mode

Example

The following example displays information about the currently logged-in user.

```
# show whoami
Current user: admin
Capabilities: admin
```

slogin

The **slogin** command initiates an SSH client connection to a specified host. You can use this command to establish a secure connection from the CLI on one Violin Array to the CLI on another.

Syntax

```
slogin [<options>] [<hostname>]
```

where:

`<options>`	Are one or more options for the **slogin** command. Enter the **slogin** command without any parameters to list the available options. See this link for a description of the options.
`<hostname>`	Is the hostname of the remote system. On a Violin Array, you can specify the hostnames `mg-a`, `mg-b`, or `mg-master` to log into an internal Memory Gateway.

Command Mode

Standard mode, Enable mode, Config mode

Example

The following shows an example of using the **slogin** command to establish a CLI session on a remote Violin Array.

```
> slogin vma01.vmem.com
The authenticity of host 'vma01.vmem.com (10.1.9.212)' can't be
established.
RSA key fingerprint is d8:3c:e0:ec:67:d7:a6:d6:16:92:35:6f:24:5c:59:84.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vma01.vmem.com,10.1.9.212' (RSA) to the
list of known hosts.

Unauthorized Access Prohibited.  Usage of the Violin Array is subject
to the Violin Systems License agreement which is included under this
product's Web Interface Help section.

Last login: Tue Nov 20 06:09:27 2018 from 169.254.1.10

Violin Array Controller

Cluster ID:          15000-0000-0000
Cluster name:        vma01
Management IP:       10.1.9.212/22
Cluster master IF:   eth1
Cluster node count:  2
Local name:          vma01-acmb
Local role:          master
Local state:         online
Master address:      10.1.9.208 (ext) 169.254.1.11 (int)
Master state:        online

NOTICE: mg-a booted/running.
NOTICE: mg-b booted/running.

vma01-acmb > enable
vma01-acmb # exit
Connection to vma01.vmem.com closed.
>
```

snmp-server community

The **snmp-server community** command sets the community name required to be supplied with SNMP requests to the system.

Note: The community name must be strong and difficult to guess. Use the same policy as passwords to set the community name.

Syntax

```
snmp-server community <community> [ro]
no snmp-server community [<community>]
```

where `<community>` is the SNMP community. The **ro** option adds the community as a read-only community, so that management stations can retrieve, but not modify MIB objects on the Violin Array. The **no** form of the command removes all SNMP communities and resets to the default community, or if a `<community>` is specified, removes the community name from the configuration.

Command Mode

Config mode

Example

The following example configures the SNMP community name `Mgmtaccess1` on the Violin Array.

```
(config) # snmp-server community Mgmtaccess1
```

snmp-server contact

The **snmp-server contact** command sets the syscontact variable served from the System MIB in MIB-II.

Syntax

```
snmp-server contact <contact name>
no snmp-server contact
```

The **no** form of the command clears the contents of the syscontact variable.

Command Mode

Config mode

Example

The following example sets the syscontact variable to vmem_admin.

```
(config) # snmp-server contact vmem_admin
```

snmp-server enable

The **snmp-server enable** command activates SNMP or individual SNMP components on the Violin Array.

Syntax

```
[no] snmp-server enable [communities | mult-communities | traps]
```

Entering the **snmp-server enable** command without options enables the SNMP server, including serving of SNMP variables sending of SNMP traps. The **communities** option enables community-based authentication on this system. The **mult-communities** option allows multiple communities to be configured. The **traps** option enables sending of SNMP traps from this system.

The **no** form of the command disables SNMP on the Violin Array entirely, or for a specified option.

Command Mode

Config mode

Examples

The following example enables the SNMP server on the Violin Array.

```
(config) # snmp-server enable
```

The following example enables sending of SNMP traps from the Violin Array.

```
(config) # snmp-server enable traps
```

Traps may only be enabled if the SNMP server overall is enabled. The following traps are sent by the SNMP agent:

- Cold boot (may include SNMP configuration having been changed)
- Link up/down
- CPU load too high
- CPU load no longer too high
- Paging activity too high

Note that traps are only sent if there are trap sinks configured with the **snmp-server host** command, and if these trap sinks are themselves enabled.

snmp-server host

The **snmp-server host** command specifies information about hosts that will receive SNMP traps from the Violin Array.

Syntax

```
[no] snmp-server host <ip-address> disable
[no] snmp-server host <ip-address> traps [<community>] [port <number>]
[version 1 | 2c]
```

where:

`<ip-address>`	Is the IP address of a host to be used as an SNMP trap sink.
`disable`	Disables sending traps to the SNMP trap sink, but does not remove it from the configuration. Use the **no** form of the command to re-enable sending traps to the host.
`traps`	Enables sending SNMP traps to the specified host.
`<community>`	Optionally specifies the community string.
`port <number>`	Overrides the default target port for this trap sink.
`version 1 | 2c`	Sets the SNMP version of traps to send to this host.

Command Mode

Config mode

Example

The following example configures the Violin Array to send SNMP traps to the host at 10.10.10.10.

```
(config) # snmp-server host 10.10.10.10 traps
```

snmp-server listen enable

The **snmp-server listen enable** command enables the interface listen list for SNMP connections. If this command is enabled, and at least one non-DHCP interface is specified in the list, SNMP connections are only accepted on interfaces in the list. When this command is disabled, SNMP connections are accepted on any interface.

Syntax

```
[no] snmp-server listen enable
```

The **no** form of the command allows SNMP connections to be accepted on any interface.

Command Mode

Config mode

Example

The following example enables the interface listen list for SNMP connections.

```
(config) # snmp-server listen enable
```

snmp-server listen interface

The **snmp-server listen interface** command specifies the list of interfaces on which SNMP connections are accepted. If at least one non-DHCP interface is specified in the list, and the **snmp-server listen enable** command is configured, SNMP connections are only accepted on interfaces in the list. Otherwise, SNMP connections are accepted on any interface.

Syntax

```
[no] snmp-server listen interface <ifname>
```

Where `<ifname>` is an interface to add to the SNMP listen list. The interface should be statically configured; that is DHCP and zeroconf should be disabled.

If the interface is also running as a DHCP client, it will be as if the interface was not added to the listen list. If DHCP is later disabled on the interface, it will be as if the interface was then added to the listen list.

Command Mode

Config mode

Example

The following example adds interface `eth0` to the listen list for SNMP connections.

```
(config) # snmp-server listen enable
(config) # snmp-server listen interface eth0
```

snmp-server location

The **snmp-server location** command sets the syslocation variable served from the System MIB in MIB-II.

Syntax

```
snmp-server location <system location>

no snmp-server location
```

The **no** form of the command clears the contents of the syslocation variable.

Command Mode

Config mode

Example

The following example sets the syslocation variable to datacenter_1.

```
(config) # snmp-server location datacenter_1
```

snmp-server port

The **snmp-server port** command sets the UDP port for the SNMP agent.

Syntax

```
[no] snmp-server port <number>
```

The **no** form of the command resets the port used for the SNMP agent to the default of UDP port 161.

Command Mode

Config mode

Example

The following example sets the port for the SNMP agent to UDP port 4310.

```
(config) # snmp-server port 4310
```

snmp-server traps

The **snmp-server traps** command specifies settings for sending traps to hosts configured to receive them from the Violin Array.

Syntax

```
[no] snmp-server traps {community <community> | event <event-name> | port <number>}
```

where:

`community <community>`	Set the default community string for SNMP traps. This setting applies to SNMP traps sent to hosts that do not have a custom community string set.
`event <event-name>`	Specifies the events that are sent as SNMP traps. By default, all notifiable events are sent as SNMP traps to any configured trap sinks. Use the **no** form of the command to disable individual events for conversion to SNMP traps.
`port <number>`	Sets the default port to which SNMP traps are sent. The **no** form of the command resets the port used for sending traps to the default of UDP port 162.

Command Mode

Config mode

Examples

The following example configures the default community string for SNMP traps.

```
(config) # snmp-server traps community mgmtaccess
```

The following example disables sending SNMP traps for the event `user-logout`.

```
(config) # no snmp-server traps event user-logout
```

snmp-server traps send-test

The **snmp-server traps send-test** command sends a test SNMP trap to all configured trap sinks. The trap that is sent is the testTrap notification from the TMS-MIB. This trap is only ever sent on request from the user; it is never triggered automatically. The testTrap notification cannot be enabled or disabled with the **snmp-server traps events** command; it is always enabled, meaning it will always be sent when requested by the user.

Syntax

```
snmp-server traps send-test
```

Command Mode

Enable mode, Config mode

Example

The following example sends a test trap to all configured trap sinks.

```
(config) # snmp-server traps send-test
```

snmp-server user v3

The **snmp-server user v3** command specifies identity and security parameters for an SNMPv3 user on the Violin Array.

Syntax

```
snmp-server user <username> v3 [encrypted | prompt] auth <hash-type>
<password> [priv <encryption-type> [<password>]]
no snmp-server user <username> v3
```

where:

`<username>`	Specifies the name of the SNMPv3 user to be configured.
`encrypted`	Allows you to specify the passwords for the command in encrypted format.
`prompt`	Causes the CLI to prompt you for the passwords. Use this as an alternative to entering the passwords on the command line.
`auth <hash-type> <password>`	Specifies the hash algorithm and hashed password to be used for authentication of the SNMPv3 user.
`priv <encryption-type> [<password>]`	Optionally configures the SNMPv3 privacy settings for this user. You can specify the encryption type and optionally specify a password. If you do not specify a password here, the password specified with the **auth** parameter is used.

The **no** form of the **snmp-server user v3** command deletes the SNMPv3 user from the configuration.

Command Mode

Config mode

Example

The following example creates an SNMPv3 user named “oboe” with password “violin”. The MD5 hash algorithm is used for authentication.

```
(config) # snmp-server user oboe v3 auth md5 violin
```

snmp-server user v3 enable

The **snmp-server user v3** command enables or disables an SNMPv3 user on the Violin Array.

Syntax

```
[no] snmp-server user <username> v3 enable
```

The **no** form of the **snmp-server user v3 enable** disables SNMPv3 access for the specified `<username>`.

Command Mode

Config mode

Examples

The following example enables SNMPv3 access for a user named “oboe”.

```
(config) # snmp-server user oboe v3 enable
```

The following example disables SNMPv3 access for the user “oboe”.

```
(config) # no snmp-server user oboe v3 enable
```

ssh client generate identity

The **ssh client generate identity** command generates a new identity (DSAv2 private and public keys) for a specified user account. When the keys are generated, the private key is written to the user's `ssh` directory in an appropriately named file; for example, `id_dsa`.

This identity can be used when the user uses the **slogin** command to connect from the system to another host.

Syntax

```
ssh client generate identity user <userid>
```

where `<userid>` is the name of the user account for which keys are generated. The user account must already exist in the system. Do not use this command to make changes to the root account.

Command Mode

Config mode

Example

The following example creates an account called "oboe" and generates DSAv2 private and public keys) for the account.

```
(config) # username oboe
(config) # ssh client generate identity user oboe
```

The output of the **show ssh client** command indicates that the keys exist for the account.

```
# show ssh client
SSH client Strict Hostkey Checking: ask

No SSH global known hosts configured.

User Identities:
  User oboe:
    DSAv2 Public key:
ssh-dss AAAAB3NzaC1kc3MAAACBAO/j75rR4PKuiCaL8PM9piYuXtDVt53gNkTzlJVNGFgHPwphd6
djvNwj4EvKcOgkMiCWbWGBLnRwAAAIEAmFVkUzx752OAH2+RoV3VzNAgIWDl9a7u1fJDFiYRuG7wCCE
LZYKGbWBQejeXjc7u07ncI4qCJMc9lSYZ9rwMv40l4I7MgUmZx9jpJoK4YS/hV2HbFwr

    DSAv2 Private key:
********
    Passphrase:
********

No SSH authorized keys configured.
```

ssh client global host-key-check

The **ssh client global host-key-check** command configures how the system checks connecting SSH clients against its known hosts file.

Syntax

```
[no] ssh client global host-key-check {yes | no | ask}
```

where:

`yes`	Permits clients to connect only if a matching host key is already in the system's known hosts file.
`no`	Permits clients to connect always, and accept any new or changed host keys without checking the system's known hosts file.
`ask`	Configures the system to prompt the user to accept new host keys, but does not permit a connection if there is already an entry in the known hosts file that does not match the one presented by the host. This is the default behavior.

Command Mode

Config mode

Example

The following example configures the system to permit SSH clients to connect only if a matching host key for the client is already in the system's known hosts file.

```
(config) # ssh client global host-key-check yes
```

ssh client global known-host

The **ssh client global known-host** command adds or removes entries in the system's global known hosts file.

Syntax

```
ssh client global known-host <host-key>

no ssh client global known-host <known-host-entry>
```

where:

`<host-key>`	Is the host key for the host you want to add to the global known hosts file.
`<known-host-entry>`	Is an entry to be removed from the global known hosts file. You can display the entries in the global known hosts file with the **show ssh client** command.

Command Mode

Config mode

Examples

The following example adds a host key for the host **vmg01.vmem.com** to system's global known hosts file. Note the quotation marks enclosing the host key.

```
(config) # ssh client global known-host "vmg01.vmem.com ssh-rsa AAAAB3NzaC1yc2
EAAABIwAAAIEA4LyiNmJ5KrhGV07mnmP6eNG7hvB/X+df+ans5k+WwlEBb/obYUzr2zOGiC+HCRCy
fLZOGB1ALNWBvyYFC93ZYvsR9dzPoNbPkzlS7+Q+P4rzjgZGW1IHUSEjHsRayUZ2zJ+7f700ictFc5
vfUiq1X5yjk="
```

The following example removes the entry in the system's global known hosts file for the host **vmg01.vmem.com**.

```
(config) # no ssh client global known-host vmg01.vmem.com
```

ssh client identity user

The **ssh client identity user** command sets private and public keys for a specified user account. Use this command as an alternative to have the system generate the keys.

Syntax

```
ssh client identity user <userid> {private-key <key> | public-key <key>}
```

where:

`<userid>`	is the name of the user account for which keys are generated. The user account must already exist in the system. Do not use this command to make changes to the root account.
`<key>`	Is the DSAv2 public or private key for the specified user.

Command Mode

Config mode

Example

The following example sets the public and private keys for the account “oboe”.

```
(config) # ssh client identity user oboe public-key violin
(config) # ssh client identity user oboe private-key piano
```

ssh client user authorized-key

The **ssh client user authorized-key** command adds or removes a key in the list of authorized SSHv2 RSA or DSA public keys for a user account. These keys can be used to log into the user's account.

Syntax

```
ssh client user <userid> authorized-key [rsakey] sshv2 <key>
no ssh client user <userid> authorized-key [rsakey] sshv2 {<key> | <keyid>}
```

where:

`<userid>`	Is the name of the user account for which keys are generated. The user account must already exist in the system. Do not use this command to make changes to the root account.
`rsakey`	Indicates the `<key>` will be an RSA key.
`<key>`	Is the key to be added to the list of authorized public keys for the specified user.
`<keyid>`	Is an identifier for a key that can be used in place of the full key when deleting the key from the configuration. Use the **show ssh client** command to display the key IDs for each user.

Notes

- The **no** form of the **ssh client user authorized-key** command removes the specified key from the configuration.
- If a key is being pasted from a cut buffer, and it was displayed with a paging program, it is likely that newline characters have been inserted, even if the output was not long enough to require paging. Most likely **show** command output will be displayed this way, since paging is enabled by default in the CLI. You can enter the **no cli session paging enable** command prior to entering the **show** command to prevent the newline characters from being inserted.

Command Mode

Config mode

Example

The following example adds a key to the list of authorized public keys for the user "oboe".

```
(config) # ssh client user oboe authorized-key sshv2 AAAAB3NzaC1yc2EAGfN
iSv8ja5/my8a976bydUaZJQ5olC4EPxjs4tfBVjRpNYp1CBhAFks017gvmFjN4T9IASP+d
jaL9KvzIcAoHY3VW4u6C9IHm+3PAROWoNnBzMdzP18SXh5uLx6P//cDHRw/Q7LYNeuRAe1
```

The following example removes the key from the list of authorized public keys for the user “oboe”. In the example, the key ID is entered in place of the actual key.

```
(config) # show ssh client
SSH client Strict Hostkey Checking: ask

No SSH global known hosts configured.

No SSH user identities configured.

SSH authorized keys:
  User oboe:
    Key 1: AAAAB3NzaC1yc2EAGfNiSv8ja5/my8a976bydUaZJQ5olC4EPxjs4tfBVjR
pNYp1CBhAFks017gvmFjN4T9IASP+djaL9KvzIcAoHY3VW4u6C9IHm+3PAROWoNnBzMdzP
18SXh5uLx6P//cDHRw/Q7LYNeuRAe1
(config) # no ssh cli user oboe authorized-key sshv2 ?
<public key ID>
1
(config) # no ssh cli user oboe authorized-key sshv2 1
```

ssh client user identity

The **ssh client user identity** sets or generates RSAv2 and DSAv2 public and private keys for a user account.

Syntax

```
ssh client user <userid> identity <key-type> generate
ssh client user <userid> identity <key-type> private-key [<key>]
ssh client user <userid> identity <key-type> public-key <key>
no ssh client user <userid> identity [<key-type>]
```

where:

`<userid>`	Is the name of the user account for which keys are generated. The user account must already exist in the system. Do not use this command to make changes to the root account.
`<key-type>`	Is the type of key you are generating or adding for the user. Enter a key type or specify **rsa2** for RSAv2 or **dsa2** for DSAv2.
`generate`	Generates public and private keys of the specified `<key-type>` for the user.
`<key>`	Is the public or private key for the specified user. If you enter the **private-key** keyword without specifying the private key, the CLI prompts you for the private key.

The **no** form of the command deletes the SSH client identity keys for the user, either all of them, or the specified `<key-type>`.

Command Mode

Config mode

Examples

The following example generates DSAv2 private and public keys for the account “oboe”.

```
(config) # ssh client user oboe dsa2 identity generate
```

The following example specifies an RSAv2 private key for the account “oboe”. In the example, the CLI prompts you to enter the private key.

```
(config) # ssh client user oboe identity rsa2 private-key
Key: violin
Confirm: violin
```

The following example deletes the DSAv2 keys for the account “oboe”.

```
(config) # no ssh client user oboe dsa2 identity dsa2
```

ssh client user known-host remove

The **ssh client user known-host remove** command removes entries in the known hosts file for a user account.

Syntax

```
ssh client user <userid> known-host <known-host> remove
```

where:

`<userid>`	Is the name of the user account for which known hosts are being removed. Do not use this command to make changes to the root account.
`<known-host>`	Is a host to be removed from the known hosts file for the user.

Command Mode

Enable mode, Config mode

Examples

The following example removes the entry in the system’s global known hosts file for the host **vmg01.vmem.com**.

```
# ssh client user known-host vmg01.vmem.com remove
```

ssh server enable

The **ssh server enable** command enables or disables the SSH server.

Syntax

```
[no] ssh server enable
```

The **no** form of the command disables the SSH server. If the SSH server is disabled, the CLI is accessible over the serial console, or the local console using a keyboard. Note that this does not terminate existing SSH sessions; it only prevents new ones from being established.

Note: The SSH server must be enabled in order for the Violin Array to operate correctly.

Command Mode

Config mode

Example

The following example enables the SSH server.

```
(config) # ssh server enable
```

ssh server host-key

The **ssh server host-key** command sets or generates RSAv1, RSAv2, or DSAv2 public and private host keys for the SSH server.

Syntax

```
ssh server host-key generate
ssh server host-key <key-type> private-key [<key>]
ssh server host-key <key-type> public-key <key>
```

where:

`generate`	Regenerates new host keys for the SSH server. This generates three keys: RSA for SSHv1, RSA for SSHv2, and DSA for SSHv2. Note that the system automatically generates the host keys on its first boot, so this only needs to be done if a security breach is suspected and the keys need to be changed.
`<key-type>`	Is the type of host-key you are setting. Enter a key type or specify **rsa1** for RSAv1, **rsa2** for RSAv2, or **dsa2** for DSAv2.
`<key>`	Manually sets the host key of the specified key type. Use this command to set either the private or public keys. You should set both keys if you are changing them. If you enter the **private-key** keyword without specifying the private key, the CLI prompts you for the private key.

Command Mode

Config mode

Examples

The following example regenerates new host keys for the SSH server.

```
(config) # ssh server host-key generate
```

The following example specifies a DSAv2 private key for the SSH server host key. In the example, the CLI prompts you to enter the private key.

```
(config) # ssh server host-key dsa2 private-key
Key: violin
Confirm: violin
```

ssh server listen

The **ssh server listen** command configures the system to accept SSH connections only on specific interfaces.

Syntax

```
[no] ssh server listen enable
[no] ssh server listen interface <ifname>
```

where:

`enable`	Causes the system to accept SSH connections only from interfaces specified with the **ssh server listen interface** command. When disabled, or if there are no interfaces specified with the **ssh server listen interface** command, the system accepts SSH connections on all interfaces.
`interface <ifname>`	Specifies an interface to add to the list of interfaces on which the system accepts SSH connections. If the interface is also running as a DHCP client, it will be as if the interface was not added to the list. If DHCP is later disabled on this interface, it will be as if the interface was then added to the list.

Command Mode

Config mode

Example

The following example configures the system to accept SSH connections only on interfaces eth0 and eth1.

```
(config) # ssh server listen enable
(config) # ssh server listen interface eth0
(config) # ssh server listen interface eth1
```

ssh server min-version

The **ssh server min-version** command sets the version of SSH used by the SSH server, either version 1, or version 1 and 2.

Syntax

```
[no] ssh server min-version {1 | 2}
```

where **1** allows SSH version 1 or 2 to be used, and **2** allows SSH version 2 only.

Command Mode

Config mode

Example

The following example sets the minimum SSH version used by the SSH server to SSHv2

```
(config) # ssh server min-version 2
```

ssh server ports

The **ssh server ports** command specifies a list of one or more ports on which the SSH server listens for connections. By default, the SSH server listens on TCP port 22.

Syntax

```
ssh server ports <port-list>
```

where `<port-list>` is a list of one or more port numbers (1-65535). Specifying a new port list removes any previously configured port list.

Command Mode

Config mode

Example

The following example configures the SSH server to listen for connections on TCP ports 23, 34, and 45.

```
(config) # ssh server ports 23 34 45
```

ssh server x11-forwarding

The **ssh server x11-forwarding** command enables X forwarding on the SSH server for connecting SSH clients.

Syntax

```
[no] server x11-forwarding enable
```

Command Mode

Config mode

Example

The following example enables X11 forwarding on the SSH server.

```
(config) # ssh server x11-forwarding enable
```

stats alarm

The **stats alarm** command enables and configures thresholds for alarms on the Violin Array.

Syntax

```
[no] stats alarm <alarm-id> enable
[no] stats alarm <alarm-id> event-repeat {single | while-not-cleared}
stats alarm <alarm-id> {falling | rising} {error-threshold | clear-
threshold} <value>
```

where:

`<alarm-id>`	Is the alarm to be configured. Enter **stats alarm ?** to display a list of alarm IDs.
`enable`	Enables the alarm. Use the **no** form of the command to disable the alarm.
`event-repeat single`	Generates an alarm event only when the alarm changes state. This is the default.
`event-repeat while-not-cleared}`	Generates periodic alarm events while the alarm is in an error state, until the alarm is cleared. Use the **no** form of the command to reset to the default of **single**.
`falling`	Configures an alarm event to be generated when the statistic falls below a threshold value.
`rising`	Configures an alarm event to be generated when the statistic rises above a threshold value.
`error-threshold <value>`	Triggers the alarm when the statistic reaches the specified `<value>`.
`clear-threshold <value>`	Clears the alarm when the statistic reaches the specified `<value>`.

Command Mode

Config mode

Example

The following example configures the `chassis_temp` alarm. An alarm event will be generated when the temperature of the chassis rises above 75 degrees Centigrade. The alarm will be automatically cleared when the chassis temperature drops below 70 degrees. A copy of the alarm event will be generated periodically as long as the alarm remains active.

```
(config) # stats alarm chassis_temp enable
(config) # stats alarm chassis_temp rising error-threshold 75
(config) # stats alarm chassis_temp rising clear-threshold 75
(config) # stats alarm chassis_temp event-repeat while-not-cleared
```

stats alarm clear

The **stats alarm clear** command clears a specified alarm on the Violin Array. Clearing an alarm resets it to a non-error state and discards the event history for the alarm.

Syntax

```
stats alarm <alarm-id> clear
```

where `<alarm-id>` is the alarm to be cleared. Enter **stats alarm ?** to display a list of alarm IDs.

Command Mode

Enable mode, Config mode

Example

The following example clears the `chassis_temp` alarm.

```
# stats alarm chassis_temp clear
```

stats alarm rate-limit

The **stats alarm rate-limit** command configures settings for limiting the number of alarm events that are generated over specific time periods. Three time periods are supported for alarm event rate limiting: short, medium, and long. You can set the duration of these time periods and the number of alarm events that can be generated over each time period.

Syntax

```
stats alarm <alarm-id> rate-limit count {long | medium | short} <count>
stats alarm <alarm-id> rate-limit window {long | medium | short}
<duration>
```

where:

`<alarm-id>`	Is the alarm to be configured. Enter **stats alarm ?** to display a list of alarm IDs.
`count {long | medium | short} <count>`	Sets the number of alarm events that can be generated for the alarm over each of the three time periods.
`window {long | medium | short} <duration>`	Sets the duration of the short, medium, and long time periods for rate limiting the alarm. For each time period, the number of alarm events generated by the alarm is limited to no more than the `<count>` over the `<duration>`.

Command Mode

Config mode

Example

The following example configures rate limiting for the `chassis_temp` alarm. The number of alarm events generated by the `chassis_temp` alarm is limited to no more than 5 per hour, 20 per day, and 50 in a week.

```
(config) # stats alarm chassis_temp rate-limit count short 5
(config) # stats alarm chassis_temp rate-limit window short 1
(config) # stats alarm chassis_temp rate-limit count medium 20
(config) # stats alarm chassis_temp rate-limit window medium 1
(config) # stats alarm chassis_temp rate-limit count long 50
(config) # stats alarm chassis_temp rate-limit window long 7
```

stats alarm rate-limit reset

The **stats alarm rate-limit reset** command resets the rate-limiting counters and time for the specified alarm.

Syntax

```
stats alarm <alarm-id> rate-limit reset
```

where `<alarm-id>` is the alarm to be reset. Enter **stats alarm ?** to display a list of alarm IDs.

Command Mode

Enable mode, Config mode

Example

The following example resets the rate-limiting count and time values for the `chassis_temp` alarm.

```
# stats alarm chassis_temp rate-limit reset
```

stats chd

The **stats chd** command enables a specified computed historical datapoint (CHD) dataset and configures how often a new calculation is performed, as well as the range of datapoints from its source sample dataset that is used in the calculation.

Syntax

```
[no] stats chd <chd-id> enable
stats chd <chd-id> compute time {inteval <seconds> | range <seconds>}
```

where:

`<chd-id>`	Is the CHD dataset to be configured. Enter **stats chd ?** to display a list of CHD datasets.
`enable`	Enables the specified CHD dataset. Use the **no** form of the command to disable the CHD dataset. When the CHD dataset is disabled, the previously gathered data is retained, but new data is not added.
`interval <seconds>`	Specifies how often a new calculation is performed using the collected data. The CHD performs a new calculation every time the number of seconds specified for the interval elapses.
`range <seconds>`	Specifies the range of datapoints that are used in the CHD calculation. The inputs for the calculation are the datapoints in the source dataset whose timestamps fall within the last `<seconds>` seconds.

Command Mode

Config mode

Example

The following example configures the `writes_hour` CHD dataset. The `writes_hour` CHD dataset uses datapoints in the `writes` sample dataset for its source. In the example, a calculation is performed at 30-second intervals using datapoints from the `writes` sample dataset that have a timestamp from the last 20 seconds.

```
(config) # stats chd writes_hour enable
(config) # stats chd writes_hour compute time interval 30
(config) # stats chd writes_hour compute time range 20
```

stats chd clear

The **stats chd clear** command clears all of the data collected for a specified computed historical datapoint (CHD) dataset.

Syntax

```
stats chd <chd-id> clear
```

where `<chd-id>` is the CHD dataset to be cleared. Enter **stats chd ?** to display a list of CHD datasets.

Command Mode

Enable mode, Config mode

Example

The following example deletes all of the gathered data for the CHD dataset `writes_hour`.

```
# stats chd writes_hour clear
```

stats clear-all

The **stats clear-all** command clears data for all sample datasets, CHD datasets, and resets status for all alarms.

Syntax

```
stats clear-all
```

Command Mode

Enable mode, Config mode

Example

The following example clears sample, CHD, and alarm status data.

```
# stats clear-all
```

stats export

The **stats export** command exports collected statistics to a CSV (comma-separated value) file. The file can be viewed on the Violin Array or copied to another location.

Syntax

```
stats export csv <report-name> [filename <name>] [before <date> <time>]
[after <date> <time>]
```

where:

`<report-name>`	Specifies the name of the sample dataset from which the statistics are exported. You can specify one of the following: `memory` Memory utilization `paging` Paging I/O `cpu_util` CPU utilization
`filename <name>`	Is the name of the exported file. If you do not specify the filename, the file is named based on the report name and date/time the file is generated; for example, `memory-20130129-150654.csv`.
`before <date> <time>`	Limits the datapoints contained in the exported file to those with timestamps before a specified date (`<yyyy>/<mm>/<dd>` format) and time (`<hh>:<mm>:<ss>`, 24-hour format).
`after <date> <time>`	Limits the datapoints contained in the exported file to those with timestamps after a specified date (`<yyyy>/<mm>/<dd>` format) and time (`<hh>:<mm>:<ss>`, 24-hour format).

Command Mode

Enable mode, Config mode

Example

The following example exports data from the `memory` sample dataset to a file and displays the contents of the file.

```
# stats export csv memory after 2014/11/11 00:00:00
Generated report file: memory-20141111-162816.csv

# show files stats memory-20141111-162816.csv
#
# Hostname:          Trng-V7-acm-b
# Report:            Memory utilization
# Time lower bound: 2014/11/11 00:00:00
# Time upper bound: none
# Export time:       2014/11/11 16:28:16 -0800
# System version:   supervisor A7.0.0 #17-ir 2014-10-21 18:12:55 ppc acm
root@ci-fir06:super:0cf4f99
#

#
# Statistic group: Physical memory utilization
# Column 0: Timestamp
# Column 1: Free physical memory (kB)
# Column 2: Used physical memory (kB)
# Column 3: Total physical memory (kB)
#
Timestamp,Free,Used,Total
2014/11/11 15:27:50,1541088,111528,2008704
2014/11/11 15:28:10,1539600,113016,2008704
2014/11/11 15:28:30,1541088,111528,2008704
```

stats sample

The **stats sample** command enables data collection for a specified sample dataset and configures its sampling interval.

Syntax

```
[no] stats sample <sample-id> enable
stats sample <sample-id> interval <seconds>
```

where:

`<sample-id>`	Is the sample dataset to be configured. Enter **stats sample ?** to display a list of sample datasets.
`enable`	Enables data collection for the specified sample dataset. Use the **no** form of the command to disable data collection. When data collection is disabled, the previously gathered data is retained in the dataset, but new data is not added.
`interval <seconds>`	Specifies how often data is sampled and added to the dataset.

Command Mode

Config mode

Example

The following example configures the `writes` sample dataset. The `writes` sample dataset contains datapoints recording the number of NFS writes over the sampling interval. In the example, data collection is enabled for the sample dataset, and the sampling interval is set to 10 seconds.

```
(config) # stats sample writes enable
(config) # stats sample writes interval 10
```

stats sample clear

The **stats sample clear** command clears all of the data collected for a specified sample dataset.

Syntax

```
stats sample <sample-id> clear
```

where `<sample-id>` is the sample dataset to be cleared. Enter **stats sample ?** to display a list of sample datasets.

Command Mode

Enable mode, Config mode

Example

The following example deletes all of the gathered data for the sample dataset `writes`.

```
# stats sample writes clear
```

telnet

The **telnet** command invokes the Telnet client on the Violin Array. Use this command to log into a remote system that is running a Telnet server.

Syntax

```
telnet [<options>] [<hostname>]
```

where:

`<options>`	Are one or more options for the **telnet** command. See this link for information about the Telnet options.
`<hostname>`	Is the hostname of the remote system. If you are connecting to another Violin Array, make sure its Telnet server has been enabled with the **telnet-server enable** command.

Enter the **telnet** command without any parameters to enter Telnet command mode. In Telnet command mode, enter **?** to list the available options.

Command Mode

Standard mode, Enable mode, Config mode

Example

The following shows an example of using the **telnet** command to log into a Violin Systems Gateway.

```
# telnet vmg02.vmem.com
Trying 10.1.10.13...
Connected to vmg02.vmem.com.
Escape character is '^]'.

Unauthorized Access Prohibited.  Usage of the Violin Systems Gateway is
subject to the Violin Systems License agreement which is included under
this product's Web Interface Help section.

login: admin
Last login: Wed Nov 21 14:48:56 from 10.11.56.13

Violin Systems Gateway

Cluster ID:          99999-9999-1014
Cluster name:        Cluster-1
Management IP:       10.1.10.13/22
Cluster master IF:   eth1
Cluster node count:  1
Local name:          vmg02
Local role:          master
Local state:         online
Master address:      10.1.10.13 (ext) 10.1.10.13 (int)
Master state:        online

vmg02 [Cluster-1: master] # exit
Connection closed by foreign host.
```

terminal

The **terminal** command configures terminal settings for the current CLI session. The settings configured with this command override the settings auto-detected for the terminal by the Violin Array.

Syntax

```
terminal {length <lines> | resize | type <terminal-type> | width
<characters>}
no terminal type
```

where:

`length <lines>`	Sets the number of lines per page for the terminal.
`resize`	Detects the size of the terminal and adjusts to the appropriate settings.
`type <terminal-type>`	Sets the terminal type. Enter **terminal type ?** for a list of supported terminal types.
`width <characters>`	Sets the maximum number of characters per line for the terminal.

The **no** form of the command clears the terminal type setting, causing the terminal configuration for the session to be equivalent to a “dumb terminal”.

Command Mode

Enable mode, Config mode

Example

The following example configures the number of lines per page for the current CLI session.

```
# terminal length 128
```

traceroute

The **traceroute** command displays the list of hops a packet takes to reach a specified host.

Syntax

```
traceroute [<options>] [<hostname>]
```

where:

`<options>`	Are one or more options for the **traceroute** command. Enter the **traceroute** command without any parameters to list the available options. See this link for additional information about the options.
`<hostname>`	Is the hostname of the remote system. Press CTRL+C to stop the traceroute operation.

Command Mode

Standard mode, Enable mode, Config mode

Example

The following shows an example of using the **traceroute** command to show the number of hops to a remote Violin Array.

```
> traceroute vma01.vmem.com
traceroute to vma01.vmem.com (10.11.75.10), 30 hops max, 40 byte packets
 1  * * *
 2  lab-core.vmem.com (10.1.8.2)  2.622 ms  2.642 ms  2.645 ms
 3  vma01.vmem.com (10.11.75.10) 2.544 ms  2.556 ms  2.558 ms
```

usb eject

The **usb eject** command unmounts and ejects a USB mass storage device used for upgrading a Violin Array.

Syntax

```
usb eject
```

Command Mode

Config mode

Example

The following example ejects a USB mass storage device from a Violin Array. Enter this command while logged into the same ACM from which the upgrade was done.

```
(config) # usb eject
```

usb mount

The **usb eject** command mounts a USB mass storage device used for upgrading a Violin Array. Once mounted, the USB device can be specified as the location for a software image upgrade.

Syntax

```
usb eject
```

Command Mode

Config mode

Example

The following example mounts a USB mass storage device connected to the Violin Array, then upgrades the Violin Array using a software image on the USB device.

```
(config) # usb mount
(config) # system upgrade all usb://images/vma-A5-5-2.img immediate
```

username

The **username** command creates or removes a local user account. New user accounts are created initially with admin privileges and are disabled by default. To enable a user account, you must set a password for it with the **username password** command (or use the **username nopassword** command to enable the account with no password required for login).

Use the **no** form of the command to remove the user account from the system. Removing a user account does not terminate any current sessions that user has open; it just prevents new sessions from being established.

The "`show usernames`" command displays any accounts created by the system administrator and the following standard accounts, which are enabled by default:

- **admin**: System Administrator (default password is "admin")
- **monitor**: System Monitor (read only)

A root account also exists but is not listed when the "`show usernames`" command is issued. This account is disabled by default. See username password on page 313 if you need to set a password.

Account States

There are five different states for the **admin** and **monitor** accounts. By default, both accounts are enabled. For security reasons, it is recommended that a password be set for the admin account.

1. **No authentication required**

   Any user can log in to this account without having to provide authentication. The admin and monitor accounts are in this state by default and should be changed for security purposes. See username password on page 313.

2. **Local password set**

   The user can log in by entering the password whose hashed version is stored. This is unnecessary if an ssh-authorized key is installed, or if a remote authenticated server comes earlier in the authentication order. See username password on page 313.

3. **Local password login disabled**

   In this state, there is no locally-configured password to allow the user to log in. The user may still log in using an ssh authorized key if one is installed, or remote authentication (for example, RADIUS or TACACS+). The admin account cannot be in this state. See username disable on page 310.

4. **Locked out**

   This account is inaccessible using a password, ssh keys or remote authentication. However, the account still exists on the system.

   To deny a remotely-authenticated user from logging in, the user could be mapped to a locked out account. This user could not be mapped to an account that is disabled because such an account is completely hidden from the rest of the system. See username disable on page 310. The admin account may not be in this state.

5. **Completely disabled**

   The admin account cannot be in this state. See username disable on page 310.

Syntax

```
[no] username <userid>
```

where `<userid>` is the name of the user account to be created.

Command Mode

Config mode

Examples

The following example creates a user account called “oboe”.

```
(config) # username oboe
```

The following example removes the “oboe” user account from the system.

```
(config) # no username oboe
```

username capability

The **username capability** command sets the access privileges for a local user account.

Syntax

```
username <userid> capability <capability>

[no] username <userid> capability
```

where:

`<userid>`	Is the name of a local user account. If the account does not already exist on the system, it is created.
`<capability>`	Sets the capability for this user account. Specify one of the following: `admin` Can access all data and perform all configuration tasks, including modifying configuration files. User accounts have the admin capability by default. `monitor` Can read all data and perform all actions, but cannot modify the configuration. `security` Can display show commands as well as several basic configuration commands. `unpriv` Has access to Standard command mode only.

The **no** form of the command resets the account to the default capability of admin.

Command Mode

Config mode

Examples

The following example sets the capability for the user account “oboe” to monitor.

```
(config) # username oboe capability monitor
```

The following example resets the capability for the “oboe” user account to the default of admin.

```
(config) # no username oboe capability
```

username disable

The **username disable** command administratively disables a local user account or modifies its login privileges.

Syntax

```
[no] username <userid> disable [login | password]
```

where:

`<userid>`	Is the name of a local user account. Entering the **username disable** command without the **login** or **password** keywords administratively disables the specified account.
`login`	Prevents anyone from logging into this account.
`password`	Disables logging into the account with a local password. It is assumed that SSH key access will be used instead.

Notes

- After you disable login privileges for an account with the **login** or **password** keywords, if you want to re-enable the account, you must either assign the account a password with the **username password** command, or use the **username nopassword** command to enable the account with no password required for login.
- Disabling password login to an account re-enables other login methods (SSH keys or remote authentication).

Command Mode

Config mode

Examples

The following example disables the account “oboe”.

```
(config) # username oboe disable
```

The following example re-enables the account “oboe”.

```
(config) # no username oboe disable
```

The following example prevents anyone from logging into the account “oboe”.

```
(config) # username oboe disable login
```

The following example re-enables login for the account “oboe”. The password for the account is set to “string”.

```
(config) # username oboe password string
```

username full-name

The **username full-name** command allows you to apply a descriptive name to a local user account. The name appears in the output of the **show usernames** and **show users** commands.

Syntax

```
[no] username <userid> full-name <name>
```

where:

`<userid>`	Is the name of a local user account.
`<name>`	Is the descriptive name for the account. To specify more than one word in the name, enclose the words in quotes.

Command Mode

Config mode

Examples

The following example sets a descriptive name for the account “oboe”.

```
(config) # username oboe full-name "Oboe Bono"
```

The descriptive name appears in the output of the **show usernames** command. For example:

```
# show usernames
USERNAME   FULL NAME              CAPABILITY    ACCOUNT STATUS
admin      System Administrator   admin         Password set
monitor    System Monitor         monitor       Password set
oboe       Oboe Bono              admin         Password set
```

username nopassword

The **username nopassword** command enables access to a local user account without having to enter a password. For a local user account to be enabled, you must either use the **username nopassword** command to enable the account with no password required for login, or set a password for the account with the **username password** command.

Syntax

```
username <userid> nopassword
```

where:

`<userid>`	Is the name of a local user account.

Command Mode

Config mode

Example

The following example removes the password requirement for the account “oboe”.

```
(config) # username oboe nopassword
```

username password

The **username password** command configures the password for a local user account.

For a local user account to be enabled, you must either set a password for the account with the **username password** command, or use the **username nopassword** command to enable the account with no password required for login.

Syntax

```
username <userid> password [<cleartext-password>]
username <userid> password 0 [<cleartext-password>]
username <userid> password 7 <encrypted-password>
```

where:

`<userid>`	Is the name of a local user account.
`<cleartext-password>`	Specifies a password in clear text.
`0`	Indicates the password will be specified in cleartext.
`7`	Indicates the password will specified in encrypted form.
`<encrypted-password>`	Specifies a password in encrypted form. Two types of password encryption are supported, SHA-1 and MD5.

If you enter the **username password** or **username password 0** commands and press the Return key, the CLI prompts you for the cleartext password. Use this as an alternative to entering the password on the command line.

Command Mode

Config mode

Examples

The following examples set the password for the account “oboe” to “string”.

```
(config) # username oboe password string
```

```
(config) # username oboe 0 password
Password: string
 Confirm: string
```

The following example specifies the password for the account “oboe” in encrypted format.

```
(config) # username oboe password 7 $6$DOFZf9UV$zKRV0WvjA09.VjO9SV7p22M
```

varray

The **varray** command displays status information for a Violin Array.

Syntax

```
varray [<index>]
```

The `<index>` option displays information for the specified virtual device index number.

Command Mode

Enable mode, Config mode

Example

The following example displays information about a Violin Array.

```
# varray
Violin Systems LLC
Version: A7.5.0.1

Device:       /dev/vtmsa
Index:        0

-- Memory Array --
Chassis Type:            XVS 8264
Number of VIMMs:         64
Ambient Temperature:     18
Controller Temperature:  26
Power A:                 ON
Power B:                 ON
Uptime:                  2543090 secs
Lid Ajar Time:           0 secs
Fan 0:                   Medium
Fan 1:                   Slow
Fan 2:                   Slow
Fan 3:                   Medium
Fan 4:                   Slow
Fan 5:                   Slow
Alarm   LED:             OFF
Status  LED:             n/a
Power A LED:             ON
Power B LED:             ON
```

vcounts

The **vcounts** command displays data transfer counters for a Violin Array.

Syntax

```
vcounts [<index>]
```

The `<index>` option displays information for the specified virtual device index number.

Command Mode

Enable mode, Config mode

Example

The following example displays output from the **vcounts** command.

```
# vcounts
Violin Systems LLC
Version: vtms-linux-utils-D5.5.2.2, 12/27/2015

Device:        /dev/vtmsa
Index:         0

-- Target Counts --
IO   > 128K      :            0
64K  < IO <=128K :            0
32K  < IO <=64K  :            1
16K  < IO <=32K  :            1
8K   < IO <=16K  :            2642
4K   < IO <=8K   :            871
IO   =   4K      :            2129564
IO   <   4K      :            2942

IO requests      :            2759079
IO requests completed  :      2136021
active io sent to VCMs :      0
IO requests failed     :      0
IO zero-size requests  :      623058


-- Counts from all VCMs --
IRQ calls:                    3407717
IRQ calls for Violin:         3389704
IRQ calls for errors:         0
Completed I/O bytes:          8763945984
Completed read bytes:         6211884032
Completed write bytes:        2552061952
Completed I/O's:              2142204
Completed read I/O's:         1519142
Completed write I/O's:        623062
Failed read I/O's:            0
Failed write I/O's:           0
Average read bytes:           4089
Average write bytes:          4096
Unaligned host buf reads:     2942
Unaligned host bounce reads:  0
Unaligned host buf writes:    0
Unaligned host bounce writes: 0
Requested DMA reads:          1519184
Requested DMA writes:         623062
Flash partial page reads:     3017
Flash partial page writes:    0
```

The following fields are displayed by the command:

`IRQ calls`	The total interrupt request handler calls to the Violin Array device driver.
`IRQ calls for Violin`	The total calls to the Violin Array device driver where work was done.
`IRQ calls for errors`	The total of DMA errors returned as well as PCIe link loss errors.
`Completed I/O bytes`	The total bytes read/written from/to a Violin Array.
`Completed read bytes`	The total bytes read from the Violin Array.
`Completed write bytes`	The total bytes written to a Violin Array.
`Completed I/O's`	The total I/O read / write requests from and to Violin Array. This is not the individual DMA descriptors completed, but for each of the user requested I/Os.
`Completed read I/O's`	The total I/O read requests from a Violin Array. This is not the individual DMA descriptors completed, but for each of the user requested I/Os.
`Completed write I/O's`	The total I/O write requests to a Violin Array. This is not the individual DMA descriptors completed, but for each of the user requested I/Os.
`Failed read I/O's`	The total failed I/O read requests from a Violin Array. This is not the individual DMA descriptors failed, but for each of the user requested I/Os.
`Failed write I/O's`	The total failed I/O write requests to a Violin Array. This is not the individual DMA descriptors failed, but for each of the user requested I/Os.
`Average read bytes`	The rough average of read I/O request sizes.
`Average write bytes`	The rough average of write I/O request sizes.
`Unaligned host buf reads`	The total I/O read requests from a Violin Array, but only incremented when an unaligned host address required special buffer byte copying to service the DMA request.
`Unaligned host buf writes`	The total I/O write requests to a Violin Array, but only incremented when an unaligned host address required special buffer byte copying to service the DMA request.
`Requested DMA reads`	Incremented for each read DMA descriptor added to the descriptor ring. A single I/O may result in multiple DMA descriptors to complete a single I/O request.
`Requested DMA writes`	Incremented for each write DMA descriptor added to the descriptor ring. Note that a single I/O may result in multiple DMA descriptors to complete a single I/O request.

`Flash partial page reads`	Incremented when a DMA descriptor for read is less than a flash page (4kB) in size. On a DRAM-based system, this will always be 0.
`Flash partial page writes`	Incremented when a DMA descriptor for write is less than a flash page (4kB) in size which leads to a hardware Read-Modify-Write operation.

vdiag

The **vdiag** command runs a series of diagnostic tests on a Violin Array. The diagnostic tests report information about the network connectivity, link status, configuration state, temperature, and active alarms for the modules installed in the Violin Array.

Syntax

```
vdiag
```

Command Mode

Enable mode, Config mode

Example

The following example runs diagnostic tests on a Violin Array.

```
# vdiag
--- Diagnostics for lab-stein115-acmb at Wed Feb 20 19:40:14 PST 2013
--- uptime 06:37:51
--- checking this is the master
--- checking ACM liveness
    - checking connectivity for acm-a: 169.254.1.10... ok
    - checking connectivity for acm-b: 169.254.1.11... ok
    - checking connectivity for ACM-MASTER-VIP: 169.254.1.1... ok
--- checking VCM liveness
    - checking connectivity for vcm-a: 169.254.1.20... ok
    - checking connectivity for vcm-b: 169.254.1.30... ok
    - checking connectivity for vcm-c: 169.254.1.40... ok
    - checking connectivity for vcm-d: 169.254.1.50... ok
--- checking for connected VCMs
tcp        0      0 master:14567               vcm-a:2536                  ESTABLISHED
tcp        0      0 master:14567               vcm-b:4242                  ESTABLISHED
tcp        0      0 master:14567               vcm-c:2322                  ESTABLISHED
tcp        0      0 master:14567               vcm-d:4466                  ESTABLISHED
--- all VCMS connected to Array Controller
--- Versions:
    - acm-a: A6.0.0
    - acm-b: A6.0.0
    - vcm-a: A6.0.0 #2-EA (07738e0)
    - vcm-b: A6.0.0 #2-EA (07738e0)
    - vcm-c: A6.0.0 #2-EA (07738e0)
    - vcm-d: A6.0.0 #2-EA (07738e0)
--- reading VCM states
    - state for VCM: vcm-a... active       uptime 19:40:18 up 6:35, load average: 0.17, 0.10, 0.10
    - state for VCM: vcm-b... active       uptime 19:40:09 up 6:35, load average: 0.19, 0.17, 0.17
    - state for VCM: vcm-c... active       uptime 19:40:15 up 6:35, load average: 0.11, 0.11, 0.12
    - state for VCM: vcm-d... active       uptime 19:40:20 up 6:35, load average: 0.95, 0.31, 0.19
--- Vimms assigned to vcm-a (7)
00,01,03,06,43-45
--- Vimms assigned to vcm-b (5)
08,09,46-48
--- Vimms assigned to vcm-c (7)
10,11,13,16,55-57
--- Vimms assigned to vcm-d (5)
18,19,58-60
```

```
--- Vimms assigned to vcm-c (7)
10,11,13,16,55-57
--- Vimms assigned to vcm-d (5)
18,19,58-60
--- all 24 VIMMs are powered
--- reading ACM temperatures
    - ambient temp... 23
    - temp-controller for ACM: acm-a... 29
    - temp-controller for ACM: acm-b... 28
--- reading VCM temperatures
    - temp for VCM: vcm-a... 38
    - temp for VCM: vcm-b... 39
    - temp for VCM: vcm-c... 43
    - temp for VCM: vcm-d... 45
--- reading acm-a internal pcie links
    - pcie for VCM: vcm-a... gen2x4 (optimal)
    - pcie for VCM: vcm-b... gen2x4 (optimal)
    - pcie for VCM: vcm-c... gen2x4 (optimal)
    - pcie for VCM: vcm-d... gen2x4 (optimal)
    - pcie for External cable: a... gen2x8 (optimal)
--- reading acm-b internal pcie links
    - pcie for VCM: vcm-a... gen2x4 (optimal)
    - pcie for VCM: vcm-b... gen2x4 (optimal)
    - pcie for VCM: vcm-c... gen2x4 (optimal)
    - pcie for VCM: vcm-d... gen2x4 (optimal)
    - pcie for External cable: a... gen2x8 (optimal)
--- reading HBA pcie links
    - pcie for HBA: hba-a... gen2x4
    - pcie for HBA: hba-b... gen2x4
    - pcie for HBA: hba-c... gen2x4
    - pcie for HBA: hba-d... gen2x4
--- reading MG pcie links
    - pcie-in  for MG: mg-a... gen2x8 (optimal)
    - pcie-in  for MG: mg-b... gen2x8 (optimal)
    - pcie-out for MG: mg-a... gen2x8 (optimal)
    - pcie-out for MG: mg-b... gen2x8 (optimal)
--- reading VIMM Config for all VCMs
    - VIMM Config for VCM: vcm-a... Optimal
    - VIMM Config for VCM: vcm-b... Optimal
    - VIMM Config for VCM: vcm-c... Optimal
    - VIMM Config for VCM: vcm-d... Optimal
--- checking alarms
    - No ACM alarms
    - No VCM alarms
    - No VIMM alarms
    - No FPM alarms
    - No MG alarms
    - No HBA alarms
    - No FAN alarms
    - No PCM alarms
    - No PSU alarms
Summary: 0 error(s) detected on lab-stein115-acmb
```

veeprom

The **veeprom** command displays hardware information about the Violin Array, such as the main board serial number, MAC address of the management interface, and manufacturing date of the main board.

Syntax

```
veeprom [<index>]
```

The `<index>` option displays information for the specified virtual device index number.

Command Mode

Enable mode, Config mode

Example

The following example displays output from the **veeprom** command.

```
# veeprom
Violin Systems LLC
Version: vtms-linux-utils-D5.5.2.2, 12/27/2012

Device:       /dev/vtmsa
Proc device:  /proc/driver/vtms/strad0
Index:        0

-- EEPROM info --
ee_version:   1
ee_partnum:   620-0071-00_R19
ee_serialnum: 27212F00013
ee_boardver:  19
ee_mfgdate:   0327201
ee_mgmtmac:   Unknown
```

The following fields are displayed by the command:

`ee_version`	The EEPROM data format version.
`ee_partnum`	The part number of the main board.
`ee_serialnum`	The serial number of the main board.
`ee_boardver`	The version of the main board.
`ee_mfgdate`	The manufacturing date of the main board.
`ee_mgmtmac`	The MAC address of the management interface.

vincident

The **vincident** command collects useful information from the Violin Array, such as version/timestamp of the current kernel, CPU information, partition information, device configuration, and logs. Once collected, this information can be sent to Violin Customer Support for analysis to determine the source of performance issues, such as ECC errors.

Syntax

```
vincident {-a | <tty-device> | <ip-address>} [--max-timeout <seconds>]
```

where:

`-a`	Saves the output to a file on the local system.
`<tty-device>`	Sends the output to a host connected to the Violin Array with a serial cable.
`<ip-address>`	Sends the output to a Violin Array located at the specified IP address.
`--max-timeout <seconds>`	Limits the amount of time spent gathering data to the specified number of seconds.

Command Mode

Enable mode, Config mode

Example

The following shows an example of running the **vincident** command and uploading the resulting file to a destination using SCP.

```
# vincident -a
Gathering information from host...
Gathering information from target...
Target IP = 10.1.10.212
This could take a couple of minutes, please be patient...
Incident report created in vincident.20120927T162608

# file debug-dump upload vincident.20120927T162608 scp://username@hostname/path/
vincident.20120927T162608
```

vinfo

The **vinfo** command displays device and target information for the Violin Array.

Syntax

```
vinfo [<index>]
```

The `<index>` option displays information for the specified virtual device index number.

Command Mode

Enable mode, Config mode

Example

The following example displays output from the **vinfo** command.

```
# vinfo
Violin Systems LLC
Version: A7.5.0.1

Device:       /dev/vtmsa
Index:        0

-- Target Info --
Host Driver:         supervisor A7.5.0.1 #2 2018-10-23 19:51:09 ppc acm root@ci-
ginkgo1:super:1d651dd
Driver Date:          2018-10-23 19:51:09
Target S/W:           A7.5.0.1
Target Port:          1
Stripe number:        n/a
Stripe length:        n/a
Memory:               2008688 bytes
Memory Type:          2T-MLC-Flash VIMMs
RAID groups:          n/a
Granularity:          n/a
ring_size:            n/a
iotimeout:            n/a
LBA sector:           512 bytes
nomsi:                0
nomsix:               0
Serial #:             1S813F00202
Mgmt. MAC:            00:1B:97:00:54:4A
Target Ethernet Link: Up
Target IP Address:    10.5.10.34
Target Netmask:       255.255.240.0
Target Gateway:       10.5.0.1

VCM Devices:
--------------------------------------------------------------------------------
                  VCM-A            VCM-B            VCM-C            VCM-D
Device:          true             true             true             true
Memory (bytes):  true             true             true             true
RAID groups:      n/a              n/a              n/a              n/a
irqtune:          n/a              n/a              n/a              n/a
debug:            0x0              0x0              0x0              0x0
Serial #:        2S812F00046      2S736F00056      2S736F00003      2S736F00050
```

The following fields are displayed by the command:

`Host Driver`	The host system vtms device driver version.
`Driver Date`	Date the driver was compiled.
`Target S/W`	The software / firmware version running on the Violin Array.
`Memory`	The size in bytes of usable system capacity. For flash VIMMs, this value changes based on formatted capacity.
`Memory Type`	The size and type of populated VIMMs.

`RAID groups`	The number of 5-VIMM RAID groups. Spare VIMMs are not counted.
`Granularity`	The smallest access granularity for I/O request in bytes.
`RingSize`	The size of driver DMA descriptor ring per Violin Array. Must be power of 2 with range of `2 - 4096`.
`IrqTune`	The Interrupt combining tunable with `0` = disabled and `4095` being the highest value.
`IoTimeout`	Time in seconds before the device driver declares an I/O as stuck and disables Violin Array I/O access. `0` = disables timeout.
`NoMSI`	When set to `1`, specifies that the driver will not attempt to allocate a PCIe MSI-based interrupt vector.
`Debug`	The current value of driver debug mask. `0` = no debug messages.
`Serial #`	The Violin Array serial number stored on its EEPROM and also shown on the label on the back of the unit.
`Mgmt. MAC`	The Violin Array Ethernet port MAC address, useful for adding into a DHCP server configuration file.

vinventory

The **vinventory** command lists information about all of the ACMs, VCMs, and VIMMs installed in a Violin Array.

Syntax

```
vinventory [rescan]
```

The **rescan** option updates the inventory information for the Array modules.

Command Mode

Enable mode, Config mode

Example

The following example displays information about the modules in a Violin Array.

```
# vinventory


Chassis Serial           Model             Date        System Version  MIM3  layout
        1S813F00202      XVS 8264          06/04/2018  01              0x08  64_r1 with 64 vimms
-------------------------------------------------------------------------------

ACMs are type 2 in constant frequency mode
ACM     Serial           Model             Date        Version   S/W Version
acm-a   52811F00015      410-0342-01_R02   <not set>   <not set> A7.5.0.1
acm-b   52811F00005      410-0342-01_R02   <not set>   <not set> A7.5.0.1
-------------------------------------------------------------------------------

VCM     Serial           Model             Date        Version   S/W Version
vcm-a   2S812F00046      620-0216-00_R01   06/04/2018  01        A7.5.0.1 #2 (1d651dd)
vcm-b   2S736F00056      620-0216-00_R01   06/04/2018  01        A7.5.0.1 #2 (1d651dd)
vcm-c   2S736F00003      620-0216-00_R01   06/04/2018  01        A7.5.0.1 #2 (1d651dd)
vcm-d   2S736F00050      620-0216-00_R01   06/04/2018  01        A7.5.0.1 #2 (1d651dd)
-------------------------------------------------------------------------------

MG      Serial           Model             Date        Version   S/W Version
mg-a    1Y813B00264      420-0138-01_R01   06/04/2018  01
mg-b    1Y813B00263      420-0138-01_R01   06/04/2018  01
-------------------------------------------------------------------------------

HBA     Serial           Model             Date        Version
hba-a   RFE1733U07993    410-0341-00_R01   06/04/2018  01
hba-b   RFE1727T65712    410-0341-00_R01   06/04/2018  01
hba-c   RFE1727T65706    410-0341-00_R01   06/04/2018  01
hba-d   RFE1727T65765    410-0341-00_R01   06/04/2018  01
-------------------------------------------------------------------------------

PCM     Serial           Model             Date        Version
pcm-a   56815F00015      410-0270-00_R05   06/04/2018  05
pcm-b   56815F00006      410-0270-00_R05   06/04/2018  05
-------------------------------------------------------------------------------

64 Vimms 2T-MLC-Flash @ 84% capacity:
VIMM    Serial           Model             Date        FpgaVer   S/W Version
vimm00  2N812F01108      410-0344-00_R03   06/04/2018  7.1.2.3   7.2.0.0
vimm01  2N812F00390      410-0344-00_R03   06/04/2018  7.1.2.3   7.2.0.0
vimm02  2N812F00666      410-0344-00_R03   06/04/2018  7.1.2.3   7.2.0.0
vimm03  2N812F00683      410-0344-00_R03   06/04/2018  7.1.2.3   7.2.0.0
vimm04  2N812F01102      410-0344-00_R03   06/04/2018  7.1.2.3   7.2.0.0
[more output follows]
```

The following fields are displayed by the command:

`device`	The identifier for the ACM, VCM, or VIMM.
`presence`	Whether the module is present in the slot.
`state`	Whether the reported information is the most current information.
`serial`	The serial number of the module.
`model`	The model number of the module.
`version`	The hardware version of the module.

`OOS`	Whether the `out-of-service` flag is activated for the module.
`fault detail`	Information about current alarms for the module.
`type`	The capacity and system type – single level cell (SLC) or multi-level cell (MLC) – of the VIMM.
`vimm set`	The spare status of the VIMM.

vpartial

The **vpartial** command displays the number of read/write I/O requests processed and the number of partial 4kB flash pages.

Syntax

```
vpartial [<index>]
```

The `<index>` option displays information for the specified virtual device index number.

Command Mode

Enable mode, Config mode

Example

The following example displays output from the **vpartial** command.

```
# vpartial
Violin Systems LLC
Version: vtms-linux-utils-D5.5.2.2, 12/27/2015

Device:       /dev/vtmsa
Index:        0

-- Target Unaligned / Partial Counts --
Completed read I/O's:              1065825783
Unaligned host buf reads:          1964
Unaligned host buf bounce reads:   3
Requested DMA page reads:          1065829198
Flash partial page reads:          1989

Completed write I/O's:             493176708
Unaligned host buf writes:         3
Unaligned host buf bounce writes:  0
Requested DMA page writes:         493176757
Flash partial page writes:         3
```

The following fields are displayed by the command:

`Completed read I/O's`	Total I/O read requests from a Violin Array. This is not the individual DMA descriptors completed, but for each of the user requested I/Os.
`Unaligned host buf reads`	Total I/O read requests from a Violin Array, but only incremented when an unaligned host address required special buffer byte copying to service the DMA request.

`Flash partial page reads`	Incremented when a DMA descriptor for read is less than a flash page (4kB) in size.
`Completed write I/O's`	Total I/O write requests to a Violin Array. This is not the individual DMA descriptors completed, but for each of the user requested I/Os.
`Unaligned host buf writes`	Total I/O write requests to a Violin Array, but only incremented when an unaligned host address required special buffer byte copying to service the DMA request.
`Flash partial page writes`	Increments when a DMA descriptor for write is less than a flash page (4kB) in size, which leads to a hardware read-modify-write operation.

vring

The **vring** command displays the Violin Array DMA ring buffer. You can use the command to debug the internal Violin Array I/O request ring at a low level, and check for unaligned flash device access.

Syntax

```
vring [<index>] [-c]
```

The `<index>` option displays information for the specified virtual device index number. The **-c** option displays output continuously without pagination breaks.

Command Mode

Enable mode, Config mode

Example

The following example displays output from the **vring** command. Look for the transfer sizes under the column labeled `SIZE`. If most of the lines show `4096`, full 4kB accesses are being done to the Violin Array hardware, which is optimal. In an unaligned access case, you will see lines alternate between `512` and `3584` for transfer size, since two read-modify-write operations occur for each 4kB of data.

```
# vring
Violin Systems LLC
Version: A7.5.0.1

Device:       /dev/vtmsa
Proc device:  /proc/driver/vtms/strad0
Index:        0

-- DMA ring info --
INDX CMD/FLAGS  TRGT_ADDR           HOST_ADDR           SIZE
1712 0x00020000 0x000000000000a000  0x0000000636561000  4096
1713 0x00020000 0x000000000000b000  0x0000000636565000  4096
1714 0x01020000 0x000000000000a000  0x0000000636561000  4096
[more output follows]
```

vstat

The **vstat** command displays the status of the connection and the ready status of a Violin Array.

Syntax

```
vstat [<index>]
```

The `<index>` option displays information for the specified virtual device index number.

Command Mode

Enable mode, Config mode

Example

The following example displays output from the **vstat** command.

```
# vstat
Violin Systems LLC
Version: A7.5.0.1

Device:       /dev/vtmsa
Index:        0

-- Target Status --
vdev_online:  1
vdev_link:    1
reconfigure:  0
Status LED:   OFF
Alarm  LED:   OFF
PWR_A  LED:   OFF
PWR_B  LED:   OFF
lid_ajar:     0

VCM Devices:  VCM-A         VCM-B         VCM-C         VCM-D
alarm:        OFF           OFF           OFF           OFF
ready:        1             1             1             1
formatting:   0             0             0             0
format_done:  0             0             0             0
paused:       0             0             0             0
link:         1             1             1             1
raid_rebuild: 0             0             0             0
write_buffer: 0             0             0             0
linkwidth:    4             4             4             4
maxlinkwidth: 4             4             4             4
cur_payload:  256           256           256           256
max_payload:  1024          1024          1024          1024
cur_read_req: 4096          4096          4096          4096
dma_active:   0             0             0             0
io_pend:      0             0             0             0
cleanup:      0             0             0             0
resubmit:     0             0             0             0
```

The following fields are displayed by the command:

`Status LED`	Whether the Status LED is on or not.
`Alarm LED`	Whether the Alarm LED is on or not. If it is on, it indicates the status of the LED flashing.
`PWR_A LED`	Whether the Power A LED is on or not.
`PWR_B LED`	Whether the Power B LED is on or not.
`ready`	Whether the data plane is online and ready or offline.

`formatting`	Whether formatting of the VIMMs is in progress or not. This is only applicable to flash VIMM systems.
`format_done`	The progress percentage done during formatting of the VIMMs.
`paused`	The pause interval for I/Os.
`link`	Whether the PCIe connection is online or offline.
`lid_ajar`	Whether the lid is closed or not.
`raid_rebuild`	Status of a RAID group rebuild.
`write_buffer`	Whether flash write buffering is enabled or disabled.
`linkwidth`	How many active PCIe lanes are available.
`maxlinkwidth`	Maximum number of active PCIe lanes.
`cur_payload`	The size of the PCIe payload.
`max_payload`	Maximum size of the PCIe payload.
`cur_read_req`	The size of the PCIe read requests.
`dma_active`	The number of 4kB DMA descriptors actively being processed by Violin 6000 Series Memory Array hardware.
`io_pend`	The number of I/O requests in the queue for a Violin Array. A single I/O request may involve more than one 4kB DMA descriptor.

vtkermit

The **vtkermit** command opens a Kermit connection to a VCM or MG in a Violin Array. Establishing a Kermit connection to a VCM or MG automatically disables serial logging for the module. Serial logging is re-enabled after the Kermit connection is established. You can see the output on the console during the Kermit session.

Syntax

```
vtkermit <mg-id> | <vcm-id>
```

where:

`<mg-id>`	Establishes a Kermit connection to the internal Memory Gateway (MG) with the specified `<mg-id>`.
`<vcm-id>`	Establishes a Kermit connection to the vRAID Controller Module (VCM) with the specified `<vcm-id>`.

Command Mode

Config mode

Example

The following example opens a Kermit connection to an internal MG in a Violin Array.

```
(config) # vtkermit mg-a
Connecting to /dev/ttyS6, speed 9600
 Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
----------------------------------------------------


Unauthorized Access Prohibited.  Usage of the Violin Systems Gateway is subject
to the Violin Systems License agreement which is included under this product's
Web Interface Help section.

mga login: admin
```

vvimms

The **vvimms** command displays status information for Violin Array VIMMs.

Syntax

```
vvimms [<index>]
```

The `<index>` option displays information for the specified virtual device index number.

Command Mode

Enable mode, Config mode

Example

The following example displays VIMM status information.

```
# vvimms
Violin Systems LLC
Version: A7.5.0.1

Device:       /dev/vtmsa
Index:        0

VIMM RG  Type         Status  Temp(C) %-FmtCap %-DieFail  %-BlkFail  %-BlkEraseAvg
VCM
 7    1  64G-SLC-Flash   Active  33       50        0         0          0          A
11    2  64G-SLC-Flash   Active  34       50        0         0          0          A
12    3  64G-SLC-Flash   Active  33       50        0         0          0          A
13    2  64G-SLC-Flash   Active  33       50        0         0          0          A
14    0  64G-SLC-Flash   Active  33       50        0         0          0          A
15    0  64G-SLC-Flash   Active  33       50        0         0          0          A
16    0  64G-SLC-Flash   Active  34       50        0         0          0          A
17    3  64G-SLC-Flash   Active  34       50        0         0          0          A
18    3  64G-SLC-Flash   Active  33       50        0         0          0          A
39    1  64G-SLC-Flash   Active  34       50        0         0          0          A
60    1  64G-SLC-Flash   Active  37       50        0         0          0          A
64    2  64G-SLC-Flash   Active  43       50        0         0          0          A
66    3  64G-SLC-Flash   Active  39       50        0         0          0          A
67    2  64G-SLC-Flash   Active  41       50        0         0          0          A
68    0  64G-SLC-Flash   Active  40       50        0         0          0          A
69    2  64G-SLC-Flash   Active  38       50        0         0          0          A
...
```

```
# vvimms
Violin Systems LLC
Version: A7.5.0.1

Device:       /dev/vtmsa
Index:        0

VIMM    RG    Type       Status    Temp(c    %-        %-        %-        %-        VCM    Bytes-      Bytes-     ECC
                                   )         FmtCap    DieFail   BlkFail   BlkEraseAvg        read        written    Corrected

        0     2T-MLC-    Active    38        84.03     0.00      1.33      0.00      A      2,582,125,661,184   1,253,355,215,872   73 (rate: 0.00e+00)
              Flash

        1     2T-MLC-    Active    41        84.03               1.18                A      2,578,302,496,768   1,253,212,123,136   451 (rate: 0.00e+00)
              Flash

        0     2T-MLC-    Active    41        84.03               1.42                A      2,582,125,661,184   1,253,355,215,872   64 (rate: 0.00e+00)
              Flash

        s     2T-MLC-    Active    41        0                   1.17                A      13,955,697,664      0                   2,244 (rate: 0.00e+00)
              Flash

        2     2T-MLC-    Active    40        84.03               1.28                A      9,664,936,960       1,025,024           116 (rate: 0.00e+00)
              Flash

        5     2T-MLC-    Active    43        84.03               1.28                B      2,578,251,365,376   1,251,248,436,224   1,638 (rate: 0.00e+00)
              Flash
```

[More output follows]

web auto-logout

The **web auto-logout** command specifies the amount of idle time allowed for the Violin Web Interface. After the specified amount of time has elapsed, the user is automatically logged out.

Syntax

```
web auto-logout <minutes>
no web auto-logout
```

where:

`<minutes>`	Is the number of minutes of inactivity before the user is logged out of the Violin Web interface. You can specify from 0 – 525,600 minutes (one year). Specifying 0 disables auto-logout. The default is 150 minutes.

The **no** form of the command disables auto-logout.

Command Mode

Config mode

Example

The following example sets the timeout value for the Violin Web Interface to 60 minutes.

```
(config) # web auto-logout 60
```

web enable

The **web enable** command enables or disables the Violin Web Interface on the device.

Syntax

```
[no] web enable
```

The **no** form of the command disables the Violin Web Interface. By default, the Violin Web Interface is enabled.

Command Mode

Config mode

Example

The following example disables the Violin Web Interface.

```
(config) # no web enable
```

web http enable

The **web http enable** command enables or disables HTTP access to the Violin Web Interface. The command is valid only if the Violin Web Interface is enabled. The Violin Web Interface is enabled by default, but can be disabled or re-enabled with the **web enable** command.

Syntax

```
[no] web http enable
```

The **no** form of the command disables HTTP access to the Violin Web Interface. By default, HTTP access to the Violin Web Interface is enabled, as long as the Violin Web Interface itself is enabled.

Command Mode

Config mode

Example

The following example disables HTTP access to the Violin Web Interface.

```
(config) # no web http enable
```

web http port

The **web http port** command specifies the TCP port used for HTTP access to the Violin Web Interface.

Syntax

```
[no] web http port <port-number>
```

where `<port-number>` is the TCP port number used for HTTP access to the Violin Web Interface. The default is port 80. The **no** form of the command resets the port to the default of port 80; it does not disable HTTP access to the Violin Web Interface.

Command Mode

Config mode

Example

The following example sets the TCP port used for HTTP access to the Violin Web Interface to port 8080.

```
(config) # web http port 8080
```

web http redirect

The **web http redirect** command causes HTTP requests made to the Violin Array to be redirected to HTTPS. For example, when this command is configured, requests made to http://vmg01.vmem.com will instead go to https://vmg01.vmem.com.

Syntax

```
[no] web http redirect
```

By default, HTTP redirect is disabled. In order for HTTP redirect to work, HTTPS must be configured on the Violin Array.

Command Mode

Config mode

Example

The following example enables HTTP redirect on the Violin Array.

```
(config) # web http redirect
```

web httpd listen

The **web httpd listen** command configures the system to accept HTTP connections only on specific interfaces.

Syntax

```
[no] web httpd listen enable

[no] web httpd listen interface <ifname>
```

where:

`enable`	Causes the system to accept HTTP connections only from interfaces specified with the **web httpd listen interface** command. When disabled, or if there are no interfaces specified with the **web httpd listen interface** command, the system accepts HTTP connections on all interfaces.
`interface <ifname>`	Specifies an interface to add to the list of interfaces on which the system accepts HTTP connections. If the interface is also running as a DHCP client, it will be as if the interface was not added to the list. If DHCP is later disabled on this interface, it will be as if the interface was then added to the list.

Command Mode

Config mode

Example

The following example configures the system to accept HTTP connections only on interfaces eth0 and eth1.

```
(config) # web httpd listen enable
(config) # web httpd listen interface eth0
(config) # web httpd listen interface eth1
```

web httpd ssl min-version

The **web httpd ssl min-version** command sets the minimum version of TLS protocol used by the web server either to version TLS 1 or TLS 1.2. The default TLS protocol is TLS 1.2 and it can be changed if your environment does not support TLS 1.2.

Syntax

```
web httpd ssl min-version <tls1 | tls1.2>
```

Command Mode

Config mode

Example

The following example sets the minimum TLS protocol to version TLS 1.2.

```
(config) # web httpd ssl min-version tls1.2
```

web https certificate regenerate

The **web https certificate regenerate** command regenerates the certificate used for HTTPS connections. Note that the system automatically generates the HTTPS certificate when HTTPS is enabled, so the certificate needs to be regenerated only when you want to change it.

Syntax

```
web https certificate regenerate
```

Command Mode

Enable mode, Config mode

Example

The following example regenerates the certificate used for HTTPS connections to the Violin Array.

```
# web https certificate regenerate
```

web https enable

The **web https enable** command enables or disables HTTPS access to the Violin Web Interface. The command is valid only if the Violin Web Interface is enabled. The Violin Web Interface is enabled by default, but can be disabled or re-enabled with the **web enable** command.

Syntax

```
[no] web https enable
```

The **no** form of the command disables HTTPS access to the Violin Web Interface. By default, HTTPS access to the Violin Web Interface is enabled, as long as the Violin Web Interface itself is enabled.

Command Mode

Config mode

Example

The following example disables HTTPS access to the Violin Web Interface.

```
(config) # no web https enable
```

web https port

The **web https port** command specifies the TCP port used for HTTPS access to the Violin Web Interface.

Syntax

```
[no] web https port <port-number>
```

where `<port-number>` is the TCP port number used for HTTP access to the Violin Web Interface. The default is port 443. The **no** form of the command resets the port to the default of port 443; it does not disable HTTPS access to the Violin Web Interface.

Command Mode

Config mode

Example

The following example sets the TCP port used for HTTPS access to the Violin Web Interface to port 444.

```
(config) # web http port 444
```

web proxy

The **web proxy** command specifies settings for a Web proxy connection.

Syntax

```
web proxy host <hostname> | <ip-address> [port <port>]

no web proxy
```

where:

`<hostname> | <ip-address>`	Specifies the hostname or IP address of the Web proxy host.
`<port>`	Sets the TCP port used for communication with the Web proxy. The default is TCP port 1080.

The **no** form of the command disables the Web proxy function.

Command Mode

Config mode

Example

The following example configures a Web proxy.

```
(config) # web proxy host 11.10.10.1 port 8080
```

web proxy auth

The **web proxy auth** command specifies authentication settings used for connecting to a Web proxy.

Syntax

```
[no] web proxy auth authtype {<auth-type> | none | basic}

[no] web proxy auth basic {username <name> | password <password>}
```

where:

`<auth-type>`	Specifies the authentication type to be used with the Web proxy.
`none`	Specifies that no authentication is used with the Web proxy.
`basic`	Specifies that basic (username and password) authentication is used with the Web proxy.
`<name>`	If **basic** is specified as the auth-type, this is the username that is sent to the server.
`<password>`	If **basic** is specified as the auth-type, this is the password that is sent to the server.

Command Mode

Config mode

Example

The following example sets the authentication type used with the Web proxy to basic, with a username of “violin” and password “oboe”

```
(config) # web proxy auth authtype basic
(config) # web proxy auth basic username violin
(config) # web proxy auth basic password oboe
```

web session renewal

The **web session renewal** command specifies the length of time before Web session cookies are automatically regenerated.

Syntax

```
[no] web session renewal <minutes>
```

where `<minutes>` is the number of minutes until the system regenerates Web session cookies. The **no** form of the command resets this value to the default of 30 minutes.

Command Mode

Config mode

Example

The following example sets the renewal time for Web session cookies to 60 minutes.

```
(config) # web session renewal 60
```

web session timeout

The **web session timeout** command specifies the length of time before a Web session expires.

Syntax

```
[no] web session timeout <minutes>
```

where `<minutes>` is the number of minutes until a Web session expires. The **no** form of the command resets this value to the default of 150 minutes.

Command Mode

Config mode

Example

The following example sets Web session timeout value to 60 minutes.

```
(config) # web session timeout 60
```

write memory

The **write memory** command saves the running configuration to the active configuration file. This command is functionally equivalent to the **configuration write** command.

Syntax

```
write memory [local]
```

The **local** option saves the configuration on the local node only, rather than saving it on all of the nodes in the cluster.

Command Mode

Enable mode, Config mode

Example

The following example saves the running configuration to the active configuration file.

```
(config) # write memory
```

write terminal

The **write terminal** displays the running configuration on the screen. This command is functionally equivalent to the **show running-config** command.

Syntax

```
write terminal
```

Command Mode

Enable mode, Config mode

Example

The following example displays the running configuration.

```
(config) # write terminal
```