

The logo features the word "ION" in a large, bold, white sans-serif font, followed by a vertical line and the word "ACCELERATOR" in a smaller, white sans-serif font with a trademark symbol. The background is a blue abstract graphic with diagonal lines and a stylized starburst shape on the left.

**ION** | ACCELERATOR™

ION Accelerator™ 2.4.1

Command-Line Interface (CLI) Reference

7.16.2014

FUSION-iO®



## Copyright Notice

The information contained in this document is subject to change without notice.

Fusion-io MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Except to correct same after receipt of reasonable notice, Fusion-io shall not be liable for errors contained herein or for incidental and/or consequential damages in connection with the furnishing, performance, or use of this material.

The information contained in this document is protected by copyright.

© 2014, Fusion-io, Inc. All rights reserved.

Fusion-io, the Fusion-io logo, ioMemory, VSL, Virtual Storage Layer and ioDrive are registered trademarks of Fusion-io in the United States and other countries.

The names of other organizations and products referenced herein are the trademarks or service marks (as applicable) of their respective owners. Unless otherwise stated herein, no association with any other organization or product referenced herein is intended or should be inferred.

Fusion-io: 2855 E. Cottonwood Parkway, Ste. 100 Salt Lake City, UT 84121 USA

(801) 424-5500

Document No. D0006225-001\_2



## CONTENTS

About the Command-Line Interface (CLI) .....	11
Command Groups .....	11
CLI LOGIN .....	12
BASIC CLI SYNTAX .....	13
CLI Command Example .....	13
Shortening Commands .....	13
COMMANDS UNIQUE TO THE CLI .....	13
COMMON OPTIONS .....	14
TROUBLESHOOTING .....	16
Error Checking .....	16
Command Validation .....	17
OTHER FUNCTIONALITY .....	17
Combining Commands .....	17
Creating Aliases .....	17
Customizing the CLI Environment .....	17
Piping Output .....	18
Filtering Output .....	18
Using Closures and Subcommands .....	18
Logging Off, Shutting Down, or Restarting the Server .....	19
Quick-Start Tasks .....	20
MANAGEMENT TASKS .....	20
Creating and Deleting Multiple Volumes or LUNs .....	21
Sample Command Set .....	21
SOFTWARE UPDATE .....	22
Software Update Flow .....	23
Command-Line Reference .....	24
HELP, HISTORY, VERSION .....	24
help .....	24
history .....	25
version .....	26



BUS COMMANDS .....	27
buses or bus:list .....	27
bus:get .....	28
CHASSIS COMMANDS .....	29
chassis or chassis:list .....	29
chassis:get .....	30
CLUSTER COMMANDS .....	31
clusters or cluster:list .....	31
cluster:get .....	32
CNA COMMANDS .....	33
cnas or cna:list .....	33
cna:get .....	35
CONFIG COMMANDS .....	36
config:alter .....	36
config:backup .....	37
config:config .....	39
config:restore .....	40
config:verify .....	42
config:wipe .....	43
CPU COMMANDS .....	43
cpus or cpu:list .....	43
cpu:get .....	44
DRIVE COMMANDS .....	45
drives or drive:list .....	45
drive:get .....	46
FAN COMMANDS .....	47
fans or fan:list .....	47
fan:get .....	49
FIO COMMANDS .....	49
fio:beacon .....	49
fio:status .....	50
FORMAT COMMAND .....	52
format:format .....	52



INIGROUP COMMANDS .....	52
inigroup:create .....	52
inigroup:delete .....	53
inigroup:get .....	54
inigroups or inigroup:list .....	54
inigroup:update .....	55
INITIATOR COMMANDS .....	56
initiator:create .....	56
initiator:delete .....	57
initiator:get .....	58
initiators or initiator:list .....	58
initiator:update .....	59
KDUMP COMMANDS .....	60
kdumps or kdump:list .....	60
kdump:get .....	60
kdump:delete .....	61
LOG COMMAND .....	61
log:servicereport .....	61
LUN COMMANDS .....	63
lun:create .....	63
lun:delete .....	64
lun:get .....	65
luns or lun:list .....	65
Viewing LUNs by Volume .....	67
MANAGE COMMAND .....	68
manage:oem .....	68
NETWORK COMMANDS .....	69
network:addr .....	69
network:ping .....	69
NODE COMMANDS .....	70
node:get .....	70
nodes or node:list .....	71
node:local .....	73



POOL COMMANDS .....	73
pool:create .....	73
pool:delete .....	74
pool:get .....	75
pools or pool:list .....	76
pool:update .....	77
PORT COMMANDS .....	78
port:get .....	78
ports or port:list .....	78
port:update .....	80
PROFILE COMMANDS .....	81
profile:create .....	81
profile:delete .....	82
profiles or profile:list .....	83
PSU COMMANDS .....	84
psu or psu:list .....	84
psu:get .....	85
RAID COMMANDS .....	86
raid:create .....	86
raid:delete .....	86
raid:get .....	87
raids or raid:list .....	88
raid:update .....	90
RULES COMMANDS .....	91
rules:compile .....	91
rules:delete .....	91
rules:facts .....	91
rules:insert .....	92
rules:reset .....	92
rules or rules:rules .....	92
rules:run .....	93
SAFT COMMANDS .....	94
saft:list .....	94



saft:url .....	94
SERVICE COMMAND .....	95
service:services .....	95
SHELL COMMANDS .....	95
SNMP COMMANDS .....	95
snmp:get .....	95
snmp:mibs .....	96
snmp:update .....	96
SOFTWARE COMMANDS .....	97
soft:apply .....	98
soft:dropbox .....	98
soft:history .....	99
soft:revert .....	100
soft:update .....	101
soft:upload .....	101
soft:version .....	102
soft:versions .....	103
SSH COMMANDS .....	103
ssh:close .....	103
ssh:exec .....	104
ssh:scpput .....	104
ssh:sftp .....	105
ssh:tunnels .....	107
SYSTEM COMMANDS .....	108
system:keys .....	108
system:maintenance .....	109
system:messages .....	109
system:restart .....	110
system:setup .....	110
system:shutdown .....	111
system:status .....	111
TARGET COMMANDS .....	112
target:create .....	112



target:delete .....	113
target:get .....	113
targets or target:list .....	114
target:update.....	115
TEMP (TEMPERATURE) COMMANDS .....	117
temp:get.....	117
temps or temps:list .....	117
VIEW COMMAND .....	118
view:graph.....	118
VOLUME COMMANDS.....	120
volume:create .....	120
volume:delete .....	121
volume:get .....	122
volumes or volume:list .....	123
volume:update.....	124
Appendix A: Shell Commands for Scripting .....	126
shell:auth.....	126
shell:cat .....	127
shell:cd .....	127
shell:clear.....	127
shell:compare .....	128
shell:cp .....	128
shell:display .....	129
shell:each.....	129
shell:echo .....	130
shell:eval.....	131
shell:exit .....	131
shell:explain .....	132
shell:filter.....	132
shell:find.....	132
shell:fold.....	133
shell:grep.....	134
shell:head .....	135





shell:if .....	135
shell:join .....	136
shell:load .....	137
shell:ls.....	138
shell:man .....	138
shell:markdown .....	138
shell:mkdir .....	139
shell:more .....	139
shell:mv .....	139
shell:printf .....	140
shell:pwd .....	140
shell:quit.....	140
shell:rm.....	141
shell:rmdir.....	141
shell:save .....	141
shell:seq.....	142
shell:set .....	142
shell:sleep .....	145
shell:sort .....	145
shell:source.....	146
shell:tac .....	147
shell:tail .....	147
shell:tee .....	148
shell:test .....	148
shell:throw.....	149
shell:types.....	149
shell:unset .....	149
Appendix B: Common CLI Tasks.....	150
COPYING TO/FROM ION ACCELERATOR .....	150
Routing Output.....	150
Routing Input.....	152
WORKING WITH THE CLI ENVIRONMENT (TREE).....	153
Working with Tree Settings.....	154



ATTACHING TO A REMOTE ION ACCELERATOR APPLIANCE .....	155
Appendix C: About the ION Accelerator Guides .....	156



## About the Command-Line Interface (CLI)

---

With the Command-Line Interface (CLI) you can perform basic configuration tasks, as well as fine-tune and manage your ION Accelerator system.



For an introduction to ION Accelerator, as well as First Boot instructions and a variety of best practices and configuration information, refer to the *ION Accelerator Configuration Guide*.



Many of the CLI commands can affect data or configurations on a wide variety of devices. Be sure to use the commands with caution, or try them on a test system if you are unsure of their potential effects. Use the `--help` option with any command to see its command syntax and usage.

### Command Groups

The commands are arranged in the following groups:

- Help, etc.
- CNA
- Fan
- Initiator
- Manage
- Port
- Rules
- SNMP
- Target
- Bus
- Config
- FIO
- Kdump
- Network
- Profile
- SAFT
- Software
- Temp (Temperature)
- Chassis
- CPU
- Format
- Log
- Node
- PSU
- Service
- SSH
- View
- Cluster
- Drive
- Inigroup
- LUN
- Pool
- RAID
- Shell
- System
- Volume



Included in many of these command groups are a several basic types of commands:

- Create – creates a specific object
- Delete – deletes a specific object
- Get – gets information about a specific object
- List – provides a list of objects of a certain type
- Update – changes or sets the information for an object

## CLI LOGIN

To begin using the Command-Line Interface, log in to ION Accelerator at the command line, using the management URL, “Admin” username, and password you chose during installation (assuming those have not been changed since). For example:

```
# ssh admin@10.10.10.99
```



At login, the chassis serial number (circled below) appears in the console text.

Below is a sample login screen for the CLI:

```
Copyright (C) 2012-2014 Fusion-io, Inc.
-----
WARNING: This is a private system. Do not attempt to login unless you are an
authorized user. Any authorized or unauthorized access or use may be monitored
and can result in criminal or civil prosecution under applicable law.
-----
Welcome to Fusion-io ION Accelerator 2.4.0-119

System Serial Number: "2M232406FW"

To further administer go to:

https://10.60.33.141
Password:
Last login: Mon Jun  9 15:35:59 2014 from ns.int.fusionio.com
Fusion-IO FIKON (1.0.1-SNAPSHOT)

Hit '<tab>' for a list of available commands.
and '[cmd] --help' for help on a specific command.
and 'man' for detailed help.

admin@ion-48nnzer0/> █
```

To display a complete list of CLI commands from the console, press **Tab**.



## BASIC CLI SYNTAX

The basic syntax for the CLI commands is:

```
commandgroup:command --option1 <item> --option2 <item> ... arg1 arg2 ...
```

Some options require specified items – these are explained in the next section.

The Help and History commands use the syntax of \*:help and \*:history, respectively. They are explained in detail in the next section.



For Fibre Channel, the UUIDs in commands represent WWPNs.

### CLI Command Example

```
pool:create --pesize 512 mainpool fioa fiob fioc fiod fioc fioc fioc fioc fioc
```

This creates a new storage pool called mainpool, from the device IDs specified, with a physical extent size (pesize) of 512KB.

### Shortening Commands

Using the format `commandgroup:command` you can omit the `commandgroup` part *only when the result is unique to the CLI*. For example, `system:setup` can be shortened to `setup` and `config:verify` to `verify`, etc. However, `port:get` cannot be shortened to `get` because there are other `get` commands (`pool:get`, etc.) that are used by the CLI.

When shortening list commands, use the plural form. For example, `pool:list` becomes `pools`.

### Commands that Cannot Be Shortened

All CLI commands can be shortened to omit their `commandgroup` name, except the following:

- create
- delete
- get
- update

## COMMANDS UNIQUE TO THE CLI

Although most CLI commands have GUI counterparts, there are some functions that are available only via the CLI. These unique commands include:

- help
- history



- `config -db` (lists all the current settings for storage information)
- `node:update` (available in the Setup process)
- `pool:create` (when creating more than one storage pool)
- `pool: delete` (pools can be deleted only in the CLI, not the GUI)
- `pool:update`
- `raid:create` (creating multiple RAIDs)
- `raid:delete, raid:get, raids, raid:update`
- `soft:dropbox, soft:history, soft:revert, soft:version`
- `volume:create` (if the storage profile is Direct Access/JBOD)

## COMMON OPTIONS

The options that are shared by each command in the CLI are described below. They are mentioned throughout the *Command Reference* section, but the descriptions are not repeated.

`--display` (`brief|wide|list|table|xml|json|any`)

Type of display/formatting. Also: `-dt`, `-dl`, `-dx`, `-dw`, `-db`, `-dj`

`--display-table` Display as a table. For example:

```
admin@url> nodes -dt
```

ID	Status	Cluster	IP	Number
fiona	MEMBER		192.168.1.1	1
			192.168.2.1	
fionb	MEMBER		192.168.1.2	2
			192.168.2.2	



`--display-list`

Display as a list. For members that are also lists, contents are displayed as arrays. For example:

```
admin@url> nodes -dl
      id fiona
statusEnum MEMBER
      cluster null
ipaddr [192.168.1.1, 192.168.2.1]
      number 1
      id fionb
statusEnum MEMBER
      cluster null
      ipaddr [192.168.1.2, 192.168.2.2]
      number 2
```

`--display-xml`

Display as XML. (Performance statistics are printed as bytes, not in GB.) For example:

```
admin@url> nodes -dx
<node id="fiona">
  <uuid>16885952</uuid>
  <ipaddrs>
    <ipaddr>192.168.1.1</ipaddr>
    <ipaddr>192.168.2.1</ipaddr>
  </ipaddrs>
  <local>true</local>
  <num slots>6</num slots>
  <number>1</number>
  <status>MEMBER</status>
</node>
<node id="fionb">
  <uuid>33663168</uuid>
  <ipaddrs>
    <ipaddr>192.168.1.2</ipaddr>
    <ipaddr>192.168.2.2</ipaddr>
  </ipaddrs>
  <local>false</local>
  <num slots>0</num slots>
  <number>2</number>
  <status>MEMBER</status>
</node>
```

`--display-json`

Display as JSON. (Performance statistics are printed as bytes, not GB.)

`--display-wide`

Display in wide format

`--display-brief`

Display in brief format

`--display-csv`

Display as comma-separated values



<code>--display-flavor &lt;string&gt;</code>	Flavor of display/formatting. Current values are <code>vmware</code> and <code>detailed</code> (for RAID tables).
<code>--output-file &lt;filename&gt;</code>	Save the command output to a file.
<code>--output-scp &lt;user@host&gt;</code>	Save the command output via SCP to a user's home directory on a host.
<code>--output-share &lt;domain/user@host/share&gt;</code>	Save the command output to a CIFS share.
<code>--output-usb</code>	Save the command output to an attached USB drive.
<code>--wiki</code>	Table form, for cut and paste to a wiki
<code>--window</code>	Display the results in a window, if the GUI is available.

## TROUBLESHOOTING

The CLI has a number of commands to help you track and diagnose errors.

### Error Checking

When errors occur during interactive sessions, the CLI displays a short message describing the error. For example:

```
admin@url> drive:get no_disk  
  
Error executing command:  
  
com.fusionio.fikon.rest.saft.SAFTNotFound: Object not found
```

You can also use the `shell:explain` command to get more information about an error condition:

```
admin@url> explain  
  
The object you've requested doesn't exist.  
  
Try using a listing command (like drives, or volumes) to find the  
identifiers of available objects.
```





## Command Validation

When command validation is enabled, a variety of preconditions are tested on the commands you execute. Any failure of a precondition prints a descriptive message to the console, and the command is not executed.

To check whether validation is on, run `shell:validate --get`

To toggle validation, run `shell:set validate on (or off)`

Here's what a validation error might look like:

```
pool:create newpool bogus_drive  Error executing command:
Can't create pool over bogus_drive, which doesn't exist
```

## OTHER FUNCTIONALITY

### Combining Commands

You can combine multiple commands into a single one by using a semicolon to separate each command. For example, `drives;volumes` will list all the drives and then list all the volumes.

### Creating Aliases

You can create a short alias that will run a longer command. The `{ }` syntax is used to form a closure (a first-class function that can be invoked). When a variable is assigned a closure as its value, typing the name of the variable at the command line executes the closure.

The following example creates an alias named `vc` that runs the `volumes -dt --cluster` command:

```
vc={volumes -dt --cluster}
```



To use aliases in later CLI sessions, you must save the CLI environment tree (`shell:save`).

### Customizing the CLI Environment

The `shell:set` command has a wide variety of options that can be used to customize the way the CLI operates. For example, `CONFIRMATION` prompts the user before command execution; `SUPPRESS_EXECUTION` parses and validates commands but suppresses their execution; `TIME_SAFT` displays the execution times for CLI commands; etc.

For more information on the `shell:set` command, see [Appendix A: Shell Commands for Scripting](#). For more examples that help you customize your CLI environment, see [Working with the CLI Environment \(Tree\)](#) in *Appendix B: Common CLI Tasks*.



## Piping Output

You can also pipe the text output of one command into another, using the piping symbol ("|"). For example, `config:config | more` will page through the configuration one screen at a time.

A convenient pipe command is `grep`, which allows searching for values. For example:

```
luns -dt | grep some_volume
```

## Filtering Output

The CLI enables a number of useful forms of filtering. Here are some sample expressions that can guide your use:

- Get port objects and store them:  

```
> p=(ports -o) // grab port objects and store
```
- Return a list of the modes of the ports:  

```
> each $p {$1 mode} // get mode property
```
- Return a list of booleans indicating which ports are *not* management ports:  

```
> each $p {$1 . mode . neq Management}
```
- Filter the ports, returning the ones that are management ports:  

```
> each $p -w { $1 . mode . eq Management}
```
- Filter the ports, returning a list of the IDs of the ones that are management ports:  

```
> each $p -w { $1 . mode . eq Management } {$1 id}
```

## Using Closures and Subcommands

A *closure* is created by surrounding statement(s) with braces. This forms a function, which can be used directly or assigned to a variable. Within a closure you can refer to any positional argument by `$n`, where `n` is the number of the argument, starting with 1. `$args` refers to all the arguments passed to the function.

```
admin@url> each (volumes) {volume:get $1}
vol1
vol2
...
```

You can use closures to create functions, by assigning the closure to a variable name. Once created you can refer to the closure value by using the `$` symbol, or can invoke the closure by referring to the variable without the `$` sign.



```
admin@url> getall = {each (volumes) {volume:get $1}}
admin@url> getall
vol1
vol2
...
```

*Subcommands* are surrounded by parentheses. They are particularly useful with the `each` command:

```
each (volumes) {volume:get $1}
```

### Logging Off, Shutting Down, or Restarting the Server

To log off the console, use the **exit** or **quit** command. Using **exit** allows a script to specify a numeric exit code, while **quit** always returns 0.

To restart the server, use the `system:restart` command.

To shut down the server from the command line, use the `system:shutdown` command.



## Quick-Start Tasks

---

This section outlines a variety of basic but important tasks you can perform with the CLI. For details on command usage, refer to the *Command-Line Reference* that follows.

Other common but less-critical tasks are outlined in [Appendix B: Common CLI Tasks](#).

### MANAGEMENT TASKS

By running several CLI commands, you can create a basic storage configuration for your ION Accelerator appliance. For more information on each of these commands and others, refer to the [Command-Line Reference](#) section that follows, including the Help commands.

Here are some basic tasks you can complete:

1. Create a **Profile**, based on the type of performance and reliability you need. For example:

```
profile:create maximum_performance
```

This creates a storage pool with a RAID 0 array. (See [Profile Commands](#) in the *Command-Line Reference* section for more information.)

2. Create **volumes** in the storage pool that can be exported later as LUNs. For example:

```
volume:create newvolume 8 pool_md
```

This creates a volume called `newvolume`. It has a capacity of 8GB (the second parameter), using the `pool_md` storage pool.

3. Create **initiator groups**, so you can manage access to LUNs. For example:

```
inigroup:create mygroup <ini1 WWN> <ini2 WWN> <etc.>
```

This creates an initiator group named `mygroup`, with initiators optionally assigned to the group by WWN.

4. Populate each initiator group with the desired initiators. See *Sample Command Set* below for more information.



5. Create **LUNs** (export volumes) to share logical storage with initiators. For example:

```
lun:create myVolume newgroup 21:00:00:24:ff:67:5f:60
21:00:00:24:ff:67:5f:61
```

This creates a LUN by exporting `myVolume` to the initiator group `newgroup` using the specified target port WWPNs.

6. Enter the **Setup** screen after the **First Boot** process has completed, so you can change values as needed:

```
system:maintenance on (do this for both nodes if in HA mode)
```

```
system:setup <screen> (where <screen> is one of the following Setup screens to
display: lan, cluster, timezone, password, or resetios). For details, see system:setup.
```

```
system:maintenance off (do this for both nodes if in HA mode)
```

7. Use the **plural** of various commands (`raids`, `initiators`, `volumes`, etc.) to display information about the objects in the ION Accelerator system.

### Creating and Deleting Multiple Volumes or LUNs

The following commands illustrate how to use the `shell:each` and `shell:seq` commands to create loops that automate common, repetitive tasks. For complete syntax on these commands, refer to [shell:each](#) and [shell:seq](#) in *Appendix A: Shell Commands for Scripting*.

- Create 16 unique volumes of 100GB each, in `RAID10_POOL_1`, where each volume name begins with “vol” followed by a number:

```
each (seq 16) {volume:create vol$1 100 RAID10_POOL_1}
```

- Delete volumes “vol9” through “vol16”:

```
each (seq --first 9 16) {volume:delete vol$1}
```

- Create 16 unique LUNs in the `win` initiator group, using all available targets, where each volume name begins with “vol” followed by a number:

```
each (seq 16) {lun:create vol$1 win -a}
```

### Sample Command Set

The set of commands listed below shows how CLI commands can be used to do the following tasks:

- Create a Reliable Performance storage pool profile.
- Create a `Test2` volume on an HA cluster, with a size of 595GB, for `pool_md3`.
- Create an initiator group `BLUE2` for the volume.



- Assign initiators to the BLUE2 group.
- Create a LUN for the BLUE2 initiators to access the Test2 volume.

Here is the script that does the tasks:

```
profile:create reliable_performance

volume:create --cluster Test2 595 pool_md3

inigroup:create BLUE2

initiator:create --assign BLUE2 21:00:00:24:ff:69:d4:ca IONb2_1
initiator:create --assign BLUE2 21:00:00:24:ff:69:d4:cb IONb2_2
initiator:create --assign BLUE2 21:00:00:24:ff:69:d4:c8 IONb2_3
initiator:create --assign BLUE2 21:00:00:24:ff:69:d4:c9 IONb2_4

initiator:update --assign BLUE2 21:00:00:24:ff:69:d4:ca --id IONb2_1
initiator:update --assign BLUE2 21:00:00:24:ff:69:d4:cb --id IONb2_2
initiator:update --assign BLUE2 21:00:00:24:ff:69:d4:c8 --id IONb2_3
initiator:update --assign BLUE2 21:00:00:24:ff:69:d4:c9 --id IONb2_4

lun:create --all-targets --blocksize 512 Test2 BLUE2
```

## SOFTWARE UPDATE



For more information on the software update process, see [Software Commands](#) in the *Command-Line Reference* section.



If your current software version is earlier than 2.2.0, do not use the steps in either this guide or the *ION Accelerator GUI Guide* to update the software. Instead, refer to the current *ION Accelerator Release Notes* for the two-part update procedure.

To do a *non-disruptive* software update, follow the steps below (refer to the instructions for each command for more details).

1. Obtain the ION Accelerator build file (.iop) from Fusion-io Customer Support.
2. Log in to each node that is to be updated, using the physical IP address of the node.
3. Copy the .iop file onto the local ION Accelerator node. To do this, run the following CLI command (assuming an update to version 2.4.0):

```
soft:upload
```

This will place the .iop file under /home/admin.

4. To perform the update for the first node, run these commands:

```
soft:apply
```

5. Wait until the update is complete.

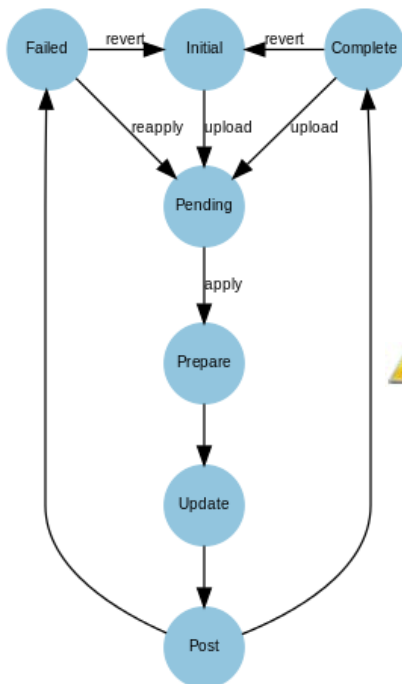


6. Run `soft:history -dt` or `soft:history -dt --cluster` to verify that the node is updated with the latest ION Accelerator software.
7. If you are using HA mode, repeat this procedure on the second node.
8. When you are finished with the update(s), log out of the CLI. The system will automatically reboot.
9. Log back in to the CLI to use the updated software,

To revert to a previous version of the software, run `soft:version`, then `soft:revert`, then `soft:version`.

### Software Update Flow

The basic flow of the software update process is illustrated in the diagram below.



The ION Accelerator software starts out in the *initial* state. After the `software:upload` command is executed, an update is present in ION Accelerator's dropbox (if verification passes). The patch can be applied with the `software:apply` command, which either completes or fails. If it fails (failed state in the diagram), the user can issue the `software:revert` command to discard the patch, or issue the `software:apply` command again, if the issue preventing the patch from succeeding has been fixed.



If you are using HA mode and want to wipe the existing configuration on one or more nodes, you must upgrade both nodes simultaneously. If you upgrade one node at a time, then one node will propagate the data to the other, and data won't be wiped out.



# Command-Line Reference

---

## HELP, HISTORY, VERSION

### help

Displays help for a command.



To display a list of all commands in the CLI, press **Tab** at the command prompt, or type `help #` and press **Enter**.

### Syntax

`help command`

Or `<command> --help` or `<command> -h`. Examples of the command usage, if available, are displayed with the help.

Or `<command> --help-all` to include common options

### Options

- |  |   |
|--|---|
| <code>--bare</code> or <code>-b</code>     | Output the help content as simplified, plain text.  |
| <code>--markdown</code> or <code>-m</code> | Output in markdown format.  |
| <code>--toc</code> or <code>-t</code>      | Generate markdown for table of contents entries.  |
| <code>--lyx</code> or <code>-l</code>      | Generate the Lyx format.  |
| <code>--all</code> or <code>-a</code>      | Display information on the following common options (see <a href="#">Common Options</a> for more details):<br><br><code>--wiki</code> , <code>--output-file</code> , <code>--output-scp</code> , <code>--output-share</code> ,<br><code>--output-usb</code> , <code>--display</code> , <code>display-brief</code> , <code>display-csv</code> ,<br><code>display-flavor</code> , <code>display-list</code> , <code>--display-json</code> , <code>--display-table</code> , <code>--display-wide</code> , <code>--display-xml</code> |





## Arguments

*command* Name of the command to get help for

## Using Auto-Completion

Pressing **Tab** after beginning to type a CLI command displays the possibilities for completing the command, listed alphabetically. Commands (partial or complete) and options can be auto-completed. Below are a few examples.

	Type this ...	See this ...
(Partial command)	<code>u&lt;tab&gt;</code>	<code>unset</code> <code>upload</code> <code>url</code>
(Full command)	<code>raid&lt;tab&gt;</code>	<code>raid:create</code> <code>raid:delete</code> <code>raid:get</code> <code>raid:list</code> <code>raid:raids</code> <code>raid:update</code> <code>raids</code>
(Partial option)	<code>raid:create --&lt;tab&gt;</code>	<code>--chunksize</code> <code>--help</code> <code>--raidtype</code>
	<code>lun:create target&lt;tab&gt;</code>	<code>21:00:00:24:ff:60:03:10</code> <code>21:00:00:24:ff:60:03:11</code> <code>21:00:00:24:ff:60:03:12</code> <code>21:00:00:24:ff:60:03:13</code>

## history

Displays recent commands that have been run. To scroll through recent commands, use the Up and Down arrows.

## Syntax

```
history [options]
```

Or ...

```
<command> --history [options]
```

## Options

`--window` or `-w` Show the history in a window, if possible.

## Notes

The `history` command also enables you to select and repeat a previous command by its prefix, by using “!” and the prefix as the command. For example:

```
> drives
fioa
> !dr
fioa
```



You can also substitute into a previous command by using “^” and the parts you want to substitute. This can be useful for correcting errors in long command strings. For example:

```
> drive:get fioa
... info A
> ^fioa^fiob^
... info B
```

After viewing history, you can recall a command to run by typing ! followed by the number of the command you want to run. For example:

```
> history
0 pool:create pool1 md0
1 lun:create -a rjvol pool1
> !1
```

## version

Shows the current CLI version, and adds ION Accelerator system version information if the `--all` option is used.

### Syntax

```
version [options]
```

### Options

`--all` or `-a` Show all available version information, including the ION Accelerator version.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all instances in the cluster.

`--parallel` Execute the command in parallel against the targets.

(See [Common Options](#) and `help --all` for more display choices.)



## Example

Below is sample information obtained by running the `version --all` command:

```
Version 2.4.0
Build Number 119
Hotfix Id ""
Update Applied
Release Date "Tue Jun 3 20:06:07 MST 2014"
Description "ION Accelerator"
Update State COMPLETE
Estimated Update Time 0
Reboot Required false
Reason
```

## BUS COMMANDS

The Bus commands get information about available buses.

`buses` or `bus:list`

Lists the IDs of the available buses.

### Syntax

```
buses [options]
```

### Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--property` or `-p` *<list>* Properties to display:

- `id` – ID of the cluster
- `uuid` – Machine-readable ID

`--objects` or `-o` Return objects (similar to the `bus:get` command).

`--separator` or `-s` *<type>*

Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.



`--cluster` Issue this command to all instances in the cluster.

`--sort <property>` Sort the output, using the specified Property name to sort on.

`--no-sort` or `-ns` Do not sort the output.

`--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{%1 method}`

`--where` or `-w <function>`  
Filter by a function, if the function is true.

`--where-not` or `-wn <function>`  
Filter by a function, if the function is false.

`--used` Show only objects that are in use.

`--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Examples

This lists available buses:

```
> buses
pci0000:00
pci0000:01
pci0000:02
...
```

### [bus:get](#)

Gets details about a bus, including UUID, bus type, and NUMA node.

### Syntax

```
bus:get [options] id
```

### Options

`--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all instances in the cluster.



(See `help --all` for details on all other options.)

### Arguments

`id` The ID, UUID, or WWPN of the bus to get information for

### Example

This gets details about the bus with the ID `pci0000:35` (from `bus:list`):

```
> bus:get pci0000:35
    Id pci0000:35
    UUID pci0000:35
    Bus Type pci
    NUMA Nodes [1]
```

## CHASSIS COMMANDS

The Chassis commands get information about available chassis.

`chassis` or `chassis:list`

Lists the available chassis.

### Syntax

```
chassis [options]
```

### Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--node` or `-n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--property` or `-p <list>` Properties to display:

- `id` – ID of the cluster
- `uuid` – Machine-readable ID

`--objects` or `-o` Return objects (similar to the `chassis:get` command).

`--separator` or `-s <type>`

Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.



- `--cluster` Issue this command to all instances in the cluster.
- `--sort <property>` Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns` Do not sort the output.
- `--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{%1 method}`
- `--where` or `-w <function>`  
Filter by a function, if the function is true.
- `--where-not` or `-wn <function>`  
Filter by a function, if the function is false.
- `--used` Show only objects that are in use.
- `--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Example

This lists the available chassis:

```
> chassis  
bda5e8f9-a3f6-5daf-bf25-ceeeef562a6
```

### chassis:get

Gets details about a chassis, including serial number, UUID, BIOS version, BIOS release date, chassis type, SKU, manufacturer, and error and warning messages (if any).

### Syntax

```
chassis:get [options] id
```

### Options

- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all instances in the cluster.

(See `help --all` for details on all other options.)



## Arguments

*id* The ID, UUID, or WWPN of the chassis to get information for

## Example

This gets details about the specified chassis (from `chassis:list`):

```
> chassis:get bda5e8f9-a3f6-5daf-bf25-ceeeef562a6
  System Serial  2M232406FW
    System UUID  36353332-3030-324D-3233-323430364657
      ID         bda5e8f9-a3f6-5daf-bf25-ceeeef562a6
      UUID       bda5e8f9-a3f6-5daf-bf25-ceeeef562a6
  BIOS Version   P70
BIOS Release Date 12/20/2013
  Chassis Type   Rack Mount Chassis
    SKU
  Manufacturer
    Errors
    Warnings
```

## CLUSTER COMMANDS

The Cluster commands return information about clusters used in HA mode.

`clusters` or `cluster:list`

Lists the cluster IDs.

### Syntax

```
clusters [options]
```

### Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--property` or `-p` *<list>* Properties to display:

- `id` – ID of the cluster
- `ipaddr` – Cluster IP address
- `uuid` – Machine-readable ID



- `--objects` or `-o` Return objects.
  - `--separator` or `-s <type>`  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
  - `--cluster` Issue this command to all instances in the cluster.
  - `--sort <property>` Sort the output, using the specified Property name to sort on.
  - `--no-sort` or `-ns` Do not sort the output.
  - `--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`
  - `--where` or `-w <function>`  
Filter by a function, if the function is true.
  - `--where-not` or `-wn <function>`  
Filter by a function, if the function is false.
  - `--used` Show only objects that are in use.
  - `--not-used` or `-nu` Show only objects that are not in use.
- (See `help --all` for details on all other options.)

### Example

This returns a list of the cluster IDs, also showing the IP addresses.

```
> clusters --property ipaddr
ionr8i47      10.60.34.47
```

### cluster:get

Gets details about a cluster, including error and warning messages (if any) and IP address.

### Syntax

```
cluster:get [options] id
```

### Options

- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.





`--cluster` Issue this command to all instances in the cluster.

(See `help --all` for details on all other options.)

### Arguments

`id` The ID, UUID, or WWPN of the cluster to get information for

### Example

This gets details about the cluster with the ID `ionr8i47` (from `cluster:list`):

```
> cluster:get mycluster
  Id  ionr8i47
  IP  10.60.34.47
Errors
Warnings
  UUID  ionr8i47
```

## CNA COMMANDS

The CNA commands return information about Converged Networking Adapters.

`cnas` or `cna:list`

Lists available CNAs (Converged Networking Adapters).

### Syntax

```
cnas [options]
```

### Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--property` or `-p <list>` One or more properties to display:

- `id` – ID of each CNA
- `uuid` – Machine-readable IDs
- `vendor` – CNA vendor name(s)
- `product` – Product name for each CNA

`--objects` or `-o` Return objects.



`--separator` or `-s` *<type>*  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.

`--node` or `-n` *<address(es)>*  
Issue this command to one or more nodes in the cluster.

`--cluster`  
Issue this command to all instances in the cluster.

`--sort` *<property>*  
Sort the output, using the specified Property name to sort on.

`--no-sort` or `-ns`  
Do not sort the output.

`--order-with` *<function>*  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`

`--where` or `-w` *<function>*  
Filter by a function, if the function is true.

`--where-not` or `-wn` *<function>*  
Filter by a function, if the function is false.

`--used`  
Show only objects that are in use.

`--not-used` or `-nu`  
Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Examples

This lists available CNAs, showing the ID and vendor for each CAN:

```
> cnas --property id --property vendor
MT27500 Family [ConnectX-3]      Mellanox Technologies
82576 Gigabit Network Connection Intel Corporation
OneConnect 10Gb NIC (be3)      Emulex Corporation
```

This lists available CNAs by UUID:

```
> cnas --uuid
00:02:c9:fc:31:a0
00:1b:21:3a:a5:f0
00:9c:02:3c:a2:a8
```



## `cna:get`

Gets information about a CNA, including fabric type, interconnect, slot #, product name, and vendor.

### **Syntax**

```
cna:get [options] id
```

### **Options**

```
--node or -n <address(es)>
```

Issue this command to one or more nodes in the cluster.

```
--cluster See --urlList above.
```

(See `help --all` for details on all other options.)

### **Arguments**

*id* The ID, UUID, or WWPN of the CNA to get information for

### **Example**

This gets details about the CNA for the UUID `00:02:c9:fc:31:a0` (from `cna:list`):

```
> cna:get 00:02:c9:fc:31:a0

  Id  MT27500  Family [ConnectX-3]
      UUID   00:02:c9:fc:31:a0
      Fabric (fabricType) not found
Interconnect (interconnect) not found
      Slot  0
      Product Mellanox MT27500 Family [ConnectX-3]
      Vendor Mellanox Technologies
```



## CONFIG COMMANDS

The Config commands provide the ability to backup and restore the configuration of an ION appliance, and to apply that configuration when provisioning other appliances.

### `config:alter`

Alters an existing configuration.

#### **Syntax**

```
config:alter [options] configuration
```

#### **Options**

- `--no-auto` or `-na` Do not automatically repair the configuration problems.
- `--from-node` or `-fn <names>`  
List of node identifiers to be changed
- `--to-node` or `-tn <names>`  
List of targets to change to
- `--from-target` or `-ft <names>`  
List of target identifiers to be changed
- `--to-target` or `-tt <names>`  
List of targets to change to
- `--input-last` Load the last known configuration from the CLI.
- `--input-file` or `-if <filename>`  
Load a file with a configuration.
- `--input-pipe` or `-ip` Use `stdin` as input to the command line (non-interactive only).
- `--input-scp` or `-is <string>`  
Use SCP input. For example, `user[:password]@host:filename`
- `--input-share` or `-ic <string>`  
Use CIFS/Windows input. For example,  
`domain/user[:password]@host/share/filename`



`--input-ssh` or `-ih` *<string>*

Use Unix shell file input, such as `user[:password]@host:filename`

`--input-url` or `-ir` *<URL>*

Load a configuration from a URL. For example,

`http://somehost/filename` Or

`ftp://[username[:password]@]host/path/file`

`--input-usb` or `-iu` *<file>*

Load a configuration from the USB drive.

`--current-file` *<filename>*

Read the current configuration from a file, instead of from ION Accelerator.

`--current-url` *<URL>*

Read the current configuration from a URL, instead of from ION Accelerator.

(See `help --all` for details on all other options.)

## Arguments

*configuration* Configuration object to modify, or variable containing the configuration

## config:backup

Backs up the current configuration to a provided destination. For configuration backup files, the CLI forms a generated filename by combining the name of the node with a timestamp, using `.xml` as an extension.

## Syntax

`config:backup [options] outputFilename`

## Options

`--message` or `-m` *<string>*

Message describing the configuration scenario

`--id` or `-i` *<string>* Identifier for this system, which will be embedded into a filename

`--input-file` or `-f` *<filename>*

Upload the configuration from a file.

`--host` *<name>* Host to load configuration from



`--share <string>` Windows (CIFS) share to load configuration from

`--domain <string>` Domain for Windows (CIFS) share user

`--user` or `-u <string>` User name

`--password` or `-p <string>`  
Password for the user

`--output-file` or `-of <file>`  
Save command output to a file or directory

`--output-scp` or `-os <string>`  
Save command output to an SCP destination  
(`user[:pass]@host[:dest]`)

`--output-share` or `-oc <string>`  
Save command output to a CIFS share  
(`domain/user[:pass]@host/share[/dest]`)

`--output-usb` or `-ou` Save command output to the USB drive mounted on the ION Accelerator system

(See `help --all` for details on all other options.)

## Arguments

`outputFilename` Optional filename for output. This is useful with `--host`, `--share`, etc.

## Examples

- `backup --host <server IP> --share <share name> --user <username> --domain <domain name> filename.xml`  
This backs up configuration to a Windows server (CIFS) share, prompting for a password.
- `backup user@host:destdir/filename.xml`  
This backs up the configuration using the `scp` protocol.
- `backup localfile.xml`  
This backs up the configuration to a local file.
- `backup user@192.168.1.1`  
This backs up the configuration to the specified `scp` target.



## config:config

Retrieves all or part of a configuration, depending on the options. If you provide the `--include` option, the set of elements to include starts empty. If you provide the `--exclude` option, the set starts with everything.

### Syntax

```
config:config [options]
```

### Options

- `--flatten` Flatten the resulting configuration into a simple list of objects.
- `--include` or `-i <DomainType>`  
Include only this type of result, starting with the empty set.
- DomainType* is one of the following: `boot_drives`, `boot_raids`, `bus`, `chassis`, `cluster`, `cna`, `cpu`, `drive`, `fan`, `inigroup`, `initiator`, `lun`, `node`, `numa`, `pool`, `port`, `profile`, `psu`, `raid`, `snmp`, `software`, `target`, `temp`, `volume`
- `--exclude` or `-x <DomainType>`  
Exclude this type of result. See the above list of domain types.
- `--objects` Do not format returned objects.
- `--uuid` Show UUIDs instead of readable IDs.
- `--input-last` Retrieve last known configuration.
- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Sample Output

```
CLUSTER: []
NODE: [ionr1sm1]
DRIVE: [fioa, fiob]
RAID: [md0]
POOL: [pool_md0]
VOLUME: [volume0, volume1, volume2, volume3, volume4]
LUN: [327a41ce-489e-11e2-9500-0025900fefc2-LUN0, 327a41ce-489e-11e2-9500-0025900fefc2-LUN1, 33587368-489e-11e2-9500-0025900fefc2-LUN0, 33587368-489e-11e2-9500-0025900fefc2-LUN1]
```



```
TARGET: [tgt, tgt]
CNA: [QLogic Corporation-QLE2562-LFD1014B42206]
PORT: [21:00:00:24:ff:21:23:4c, 21:00:00:24:ff:21:23:4d]
INITIATOR_GROUP: [ini]
INITIATOR: [21:00:00:1b:32:8b:49:77, 21:01:00:1b:32:ab:49:77,
50:01:43:80:04:25:ce:6c]
SOFTWARE: SoftwareVersion [version=2.0.1, patchLevel="", hotfixId="",
releaseDate="Mon Dec 17 09:51:55 MST 2012", buildNumber=253, description="ION
Accelerator", updating=false, updateState=INITIAL, estimatedUpdateTimeMins=0,
rebootRequired=false]
SNMP: SNMPDetail [trapAddress=null, trapCommunity=null, serviceEnabled=true]
```

## config:restore

Applies (restores) a configuration to the current node.

### Syntax

```
config:restore [options] configuration
```

### Options

- `--local` Make changes only on the local node.
- `--dry-run` Show what will be done, but don't do it.
- `--no-auto` or `-na` Do not automatically repair configuration problems.
- `--from-node` or `-fn <names>`  
List of node identifiers to be changed
- `--to-node` or `-tn <names>`  
List of targets to change to
- `--from-target` or `-ft <names>`  
List of target identifiers to be changed
- `--to-target` or `-tt <names>`  
List of targets to change to
- `--input-last` Load the last known configuration.
- `--input-file` or `-if <filename>`  
Use file input.
- `--input-pipe` or `-ip` Use stdin as input to the command line (non-interactive only).





`--input-scp` or `-is` *<string>*

Use SCP input. For example, `user[:password]@host:filename`

`--input-share` or `-iu` *<string>*

Use CIFS/Windows input. For example,  
`domain/user[:password]@host/share/filename`

`--input-ssh` or `-ih` *<string>*

Use Unix shell file input, such as `user[:password]@host:filename`

`--input-url` or `-ir` *<URL>*

Use URL input. For example, `http://somehost/filename` or  
`ftp://[username[:password]@]host/path/file`

`--input-usb` or `-iu` *<file>*

Use content retrieved from the USB drive.

`--current-file` *<filename>*

Read the current configuration from a file, instead of from ION Accelerator.

`--current-url` *<URL>*

Read the current configuration from a URL, instead of from ION Accelerator.

(See `help --all` for details on all other options.)

## Arguments

*configuration* Configuration object to modify, or variable containing the configuration

## Examples

- `restore --input-file cfg.xml`  
This restores the configuration in `cfg.xml`.
- `restore --input-last`  
This attempts to restore the last known configuration.
- `restore --from-target TGA --to-target TGB --input-file cfg.xml`  
This restores from `cfg.xml`, changing references to target TGA into references to TGB.
- `restore --input-url http://backup.server/config.xml`



This restores the configuration from an http URL.

- `restore --input-share adomain/auser@myhost/ashare/cfg.xml`

This restores a configuration from a Windows (CIFS) share.

### config:verify

Returns `TRUE` if the configuration can be applied to the current node.

#### Syntax

```
config:verify [options] configuration
```

#### Options

- `--input-last` Load the last known configuration.
- `--input-file` or `-if <filename>`  
Use file input.
- `--input-url` or `-ir <URL>`  
Use URL input. For example, `http://somehost/filename` or `ftp://[username[:password]@]host/path/file`
- `--input-usb` or `-iu <file>`  
Use content retrieved from the USB drive.
- `--input-share` or `-ic <string>`  
Use CIFS/Windows input. For example, `domain/user[:password]@host/share/filename`
- `--input-scp` or `-is <string>`  
Use SCP input. For example, `user[:password]@host:filename`
- `--input-pipe` or `-ip` Use `stdin` as input to the command line (non-interactive only).
- `--current-file <filename>`  
Read the current configuration from a file, instead of from ION Accelerator.
- `--current-url <URL>`  
Read the current configuration from a URL, instead of from ION Accelerator.

(See `help --all` for details on all other options.)



## Arguments

*configuration* Configuration object to modify, or variable containing the configuration

## `config:wipe`

Wipes (deletes) the specified resources.

## Syntax

```
config:wipe [options]
```

## Option (required)

`--wipe` or `-w <types>` Types of resources to wipe (delete all of). Types include:

`lun, volume, pool, raid, target, initiator, inigroup, all`

(See `help --all` for details on all other options.)

## CPU COMMANDS

The CPU commands get information about CPUs in the ION Accelerator host.

## `cpus` or `cpu:list`

Lists the available CPUs in the host by ID.

## Syntax

```
cpu:list [options]
```

## Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--property` or `-p <list>`

One or more properties to display

`--objects` Return objects.

`--separator` or `-s <type>`

Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.



- `--node` or `-n` *<address(es)>*  
Issue this command to one or more nodes in the cluster.
- `--cluster`  
Issue this command to all instances in the cluster.
- `--sort` *<property>*  
Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns`  
Do not sort the output.
- `--order-with` *<function>*  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`
- `--where` or `-w` *<function>*  
Filter by a function, if the function is true.
- `--where-not` or `-wn` *<function>*  
Filter by a function, if the function is false.
- `--used`  
Show only objects that are in use.
- `--not-used` or `-nu`  
Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Example

This lists all available CPUs in the host, separated by spaces:

```
> cpus -s  
0 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22 23 3 4 5 6 7 8 9
```

### [cpu:get](#)

Gets information about a CPU, including core ID, vendor, family, model, Uarch, Mhz, thread siblings, and NUMA node.

### Syntax

```
cpu:get [options] id
```

### Options

- `--node` or `-n` *<address(es)>*  
Issue this command to one or more nodes in the cluster.



`--cluster` Issue this command to all instances in the cluster.

(See `help --all` for details on all other options.)

### Arguments

`id` The ID or UUID of the CPU to get information for

### Example

This gets information about CPU 11 (from `cpu:list`):

```
> cpu:get 11
  Id 11
  UUID 11
Core Id 5
Vendor GenuineIntel
Family 6
Model 45
Uarch sandybridge-e
Mhz 2493.812
Thread Siblings 11,23
NUMA Node 1
```

## DRIVE COMMANDS

The Drive commands manipulate physical disk structures in the ION Accelerator host.

`drives` or `drive:list`

Lists available drives.

### Syntax

```
drives [options]
```

### Options

`--boot` or `-b` Include *only* boot devices in the list of drives.

`--rescan` or `-r` Force rescan of boot devices.

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--property` or `-p <list>`

One or more properties to display



- `--objects` Return objects.
- `--separator` or `-s <type>`  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all instances in the cluster.
- `--sort <property>` Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns` Do not sort the output.
- `--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{$1 method}`
- `--where` or `-w <function>`  
Filter by a function, if the function is true.
- `--where-not` or `-wn <function>`  
Filter by a function, if the function is false.
- `--used` Show only objects that are in use.
- `--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Example

This lists all available drives in the system:

```
> drives -s
fioa fiob fioc fiod fioe fiof
```

### drive:get

Gets information about a drive, including capacity, device path, slot #, adapter ID, board name, UUID, and total errors and warnings.

### Syntax



```
drive:get [options] id
```

### Options

- `--boot` or `-b` Specify that the drive is a boot device.
- `--rescan` or `-r` Force rescan of boot devices.
- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all instances in the cluster.

(See `help --all` for details on all other options.)

### Arguments

`id` The ID, UUID, or WWPN of the drive to get information for

### Example

This gets information about the drive named `fioa` (from `drive:list`):

```
> drive:get fioa
      Id  fioa
Capacity 1,205.00 GB
Device   /dev/fioa
Slot     3
Adapter  1150D0032
Board Name ioDrive2 Adapter Controller
      UUID 1150D0032-1121
      Err/Warn []
```

## FAN COMMANDS

The Fan commands get information about available fans.



Fan speed may be reported either as a percentage or in RPM. Check the units that apply to your particular platform.

`fans` or `fan:list`

Lists the available fans.

### Syntax

```
fan:list [options]
```



## Options

- `--uuid` or `-u` Show UUIDs instead of readable IDs.
- `--node` or `-n` *<address(es)>*  
Issue this command to one or more nodes in the cluster.
- `--property` or `-p` *<list>* Properties to display:
- `id` – ID of the cluster
  - `uuid` – Machine-readable ID
  - `ipaddr` – Cluster IP address
- `--objects` or `-o` Return objects.
- `--separator` or `-s` *<type>*  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--cluster` Issue this command to all instances in the cluster.
- `--sort` *<property>* Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns` Do not sort the output.
- `--order-with` *<function>*  
Sort the output, extracting key with this function.  
Example: `{$1 method}`
- `--where` or `-w` *<function>*  
Filter by a function, if the function is true.
- `--where-not` or `-wn` *<function>*  
Filter by a function, if the function is false.
- `--used` Show only objects that are in use.
- `--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)





## fan:get

Gets details about a fan.

### Syntax

```
fan:get [options] id
```

### Options

```
--node or -n <address(es)>
```

Issue this command to one or more nodes in the cluster.

```
--cluster
```

Issue this command to all instances in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* The ID, UUID, or WWPN of the fan to get information for

### Example

This gets details about the fan with the ID `fan1`.

```
> fan:get fan1
```

## FIO COMMANDS

The `fio` commands provide information about the ioDrive devices used in the ION Accelerator appliance. These commands are similar to the Command-Line Utilities available with the VSL software.

## fio:beacon

Enable or disable the beacon for an ioDrive attached to a device control node (using the `--on` or `--off` options), or return its status (using neither `--on` nor `--off`).

### Syntax

```
fio:beacon [options] device-node
```

### Options

```
--on Enable the beacon.
```

```
--off Disable the beacon.
```



`--ppci` Print the PCI bus ID of the device node.

(See `help --all` for details on all other options.)

### Arguments

`device-node` ioDrive device control node, such as `/dev/fct1`

### Example

This turns on the beacon for the `fct1` device and prints its PCI bus ID:

```
> fio:beacon --on --ppci /dev/fct1
PCI address: f:0.0
/dev/fct1 beacon ON
```

### [fio:status](#)

Determines the status of Fusion-io devices by displaying a variety of information fields.

### Syntax

`fio:status` [`options`] `device`

### Options

- `--all` or `-a` Report all available information.
- `--err-warn` or `-e` Report only errors and warnings, with minimal device info.
- `--data-volume` or `-d` Report the volume of data read and written.
- `--unavailable-detail` or `-U`  
Show unavailable fields and details for why they are unavailable.
- `--unavailable` Show unavailable fields.
- `--list-fields` or `-l` List fields that can be individually accessed with `--field`.
- `--count` or `-c` Report the number of Fusion-IO boards installed.
- `-fs` Use the standard output format.
- `-fx` Use XML output format.
- `-fj` Use JSON output format.
- `--field <string>` Request a particular field. This option is repeatable.

(See `help --all` for details on all other options.)



## Arguments

*device* Pathname to the control device

## Example

This displays the status for the `/dev/fct1` device:

```
> fio:status /dev/fct1
Found 1 ioMemory device in this system with 1 ioDrive Duo as device
'/dev/fct1'
Driver version: 3.2.6 build 1219

Adapter: Dual Controller Adapter
      Fusion-io ioDrive2 Duo 2.41TB, Product Number:F01-001-2T41-CS-
0001, SN:1150D0032, FIO SN:1150D0032
      External Power: NOT connected
      PCIe Power limit threshold: 55.00W
      Connected ioMemory modules:
        fct1: SN:1150D0032-1111

fct1   Attached
      SN:1150D0032-1111
      Located in slot 0 Upper of ioDrive2 Adapter Controller
SN:1150D0032
      PCI:0f:00.0, Slot Number:3
      Firmware v7.1.13, rev 109322 Public
      1205.00 GBytes device size
      Internal temperature: 38.39 degC, max 42.82 degC
      Reserve space status: Healthy; Reserves: 100.00%, warn at 10.00%
      Contained VSUs:
        fiob: ID:0, UUID:c6c0e0b9-79e9-43bf-8482-b9ef29b7d656

fiob   State: Online, Type: block device
      ID:0, UUID:c6c0e0b9-79e9-43bf-8482-b9ef29b7d656
      1205.00 GBytes device size
```



## FORMAT COMMAND

The `format` command formats objects.

### `format:format`

Formats objects.

#### Syntax

```
format [options] item(s)
```

#### Options

`--flatten` or `-f`      Flattens a collection of arguments into a single one

`--maxdepth` or `-m` *<depth>*

Maximum depth for flattening arguments; default is 4

#### Arguments

*item*                      Objects to flatten. This argument can be used multiple times.

## INIGROUP COMMANDS

The `inigroup` commands enable you to manipulate named groups of initiators. Initiator groups can be organized into a tree, where the leaves of the tree are the initiators. Each initiator or initiator group can have one parent initiator group. Setting the parent of an initiator group to a non-existent group implicitly creates that group.

### `inigroup:create`

Creates an initiator group. If HA is enabled, the group is created across a cluster.

#### Syntax

```
inigroup:create [options] id initiator(s)
```

#### Options

`--uuid` or `-u` *<string>*    UUID for the group (generated if not provided)

`--parent_uuid` or `-p` *<string>*

Optional parent group UUID



`--type` or `-t` *<InitiatorGroupType>*

Optional type of the initiator group: `default` or `aix`. The blocksize for creating AIX groups must be 512B.

`--if_not_exists` or `-ne` If an object with the given identifier already exists, skip creation.

(See `help --all` for details on all other options.)

### Arguments

*id* Human-readable id for the initiator group

*initiator* Optional identifier of initiator to add to this group. This option can be included multiple times.

### Example

This creates an initiator group named `mygroup` that belongs to the parent `0c8e4659-855c-4f86-9712-ed7ba476b1eb`:

```
> inigroup:create --parent_uuid 0c8e4659-855c-4f86-9712-ed7ba476b1eb
mygroup
```

### `inigroup:delete`

Deletes one or more initiator groups (across a cluster if in HA mode).

### Syntax

```
inigroup:delete [options] id(s)
```

### Options

(See `help --all` for details on all other options.)

### Arguments

*id* The ID or UUID of the initiator group to delete. This option may be used multiple times.

### Example

This deletes the initiator group named `tempgroup`:

```
> inigroup:delete tempgroup
```



## `inigroup:get`

Gets details about an initiator group, including type and parent (if any), the IDs for the initiators in the group, and the group UUID.

### Syntax

```
inigroup:get [options] id
```

### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* The ID, UUID, or WWPN of the initiator group to get information for

### Example

This gets information for the `w2k12` initiator group (from `inigroup:list`):

```
> inigroup:get W2K12
      Id  W2K12
      Type
      Parent
      Initiators  21:00:00:24:ff:69:d1:40
                  21:00:00:24:ff:69:9a:bc
      UUID  80550156-b229-447f-9995-8ba9b87c8fbf_win
```

## `inigroups` or `inigroup:list`

Lists initiator groups.

### Syntax

```
inigroups [options]
```

### Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--property` or `-p` *<list>*

One or more initiator group properties to display

`--objects` or `-o` Return objects.



`--separator` or `-s` *<type>*

Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.

`--node` or `--n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

`--sort` *<property>*

Sort the output, using the specified Property name to sort on.

`--no-sort` or `-ns`

Do not sort the output.

`--order-with` *<function>*

Sort the output, extracting key with this function.

Example: `{ $1 method }`

`--where` or `-w` *<function>*

Filter by a function, if the function is true.

`--where-not` or `-wn` *<function>*

Filter by a function, if the function is false.

`--used`

Show only objects that are in use.

`--not-used` or `-nu`

Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Example

This lists all the initiator groups in the system:

```
> inigroups
W2K12
```

### [inigroup:update](#)

Updates (re-keys) an initiator group, by assigning it to a different parent group or giving it a new ID.

### Syntax

```
inigroup:update [options] id
```



## Options

`--parent_uuid` or `-p` *<string>*

New parent group UUID

`--rename` or `--id` or `-i` *<string>*

Rename this initiator group to the specified string.

(See `help --all` for details on all other options.)

## Arguments

*id* The ID or UUID of the initiator group to update

## Example

This updates (renames) the `oldgroup` initiator group to `newgroup`:

```
> inigroup:update --id newgroup
```

## INITIATOR COMMANDS

The Initiator commands enable you to create, delete, list, get information for, and update remote SCSI initiators.

### `initiator:create`

Manually creates an initiator, directly specifying a unique identifier for port, as well as an optional name. If performed in an HA cluster, the initiator definition is created across each machine in the cluster. This command accepts WWPNs (such as `£8:e9:d2:c3:b4:a5:£6:e7`), IQNs (such as `iqn.1992-01.com.example.storage.disk2.sys1.xyz`) or GUID identifiers (such as `0002:c903:004c:7535`) for the initiator.

## Syntax

```
initiator:create [options] UUID id
```

## Options

`--assign` or `-a` *<string>* Assign the newly created initiator to a group.

`--if_not_exists` or `-ne`

If an object with the given identifier already exists, skip creation.

(See `help --all` for details on all other options.)

## Arguments

*UUID* WWPN, IQN, or GUID for the initiator. For example:





WWPN:f8:e9:d2:c3:b4:a5:f6:e7

IQN: iqn.1992-01.com.exempl:dsk.sys1.xy[3]

GID: 0002:c903:004c:7535

*id* Human-readable identifier for the initiator

### Example

This creates the initiator `init22` at WWPN `21:00:00:24:ff:67:5f:60 ...`

```
> initiator:create --assign init22 21:00:00:24:ff:67:5f:60
```

### [initiator:delete](#)

Deletes an initiator.

### Syntax

```
initiator:delete [options] id(s)
```

### Options

`--node` or `--n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* The ID, UUID, or WWPN of the initiator to delete. This argument may be used multiple times.

### Deleting an Initiator

To delete an initiator from an existing group, run the following commands:

```
inigroup:create trashcan
```

```
initiator:update --assign trashcan initiator
```

```
inigroup:delete trashcan
```

This process a) creates a temporary group (`trashcan` in this case) to hold the unwanted initiator; b) assigns that initiator to the temporary group; and c) deletes the temporary group.



## `initiator:get`

Gets information about an initiator, including UUID, protocol, discovery status, and initiator group ID.

### Syntax

```
initiator:get [options] id
```

### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

### Arguments

*id* The ID, UUID, or WWPN to get information for

### Example

This gets information about the `win_1` initiator (from `initiator:list`):

```
> initiator:get win_1
      id  win_1
      UUID  iqn.1991-05.com.microsoft:win-pq45oau7hi9#192.168.20.48
      Protocol  iSCSI
      Discovered  false
Initiator Group  9482affc-fd81-11e3-892b-0015178fbc10
```

## `initiators` or `initiator:list`

Lists available initiators.

### Syntax

```
initiators [options]
```

### Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--property` or `-p` *<list>*

One or more initiator group properties to display

`--objects` or `-o` Return objects.



`--separator` or `-s` *<type>*  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.

`--node` or `-n` *<address(es)>*  
Issue this command to one or more nodes in the cluster.

`--cluster`  
Issue this command to all nodes in the cluster.

`--sort` *<property>*  
Sort the output, using the specified Property name to sort on.

`--no-sort` or `-ns`  
Do not sort the output.

`--order-with` *<function>*  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`

`--where` or `-w` *<function>*  
Filter by a function, if the function is true.

`--where-not` or `-wn` *<function>*  
Filter by a function, if the function is false.

`--used`  
Show only objects that are in use.

`--not-used` or `-nu`  
Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Example

This lists the initiators, separated by spaces:

```
> initiators -s  
win_1 win_2 win_3 win_4
```

### [initiator:update](#)

Updates an existing initiator, by renaming it or assigning it to a group. See also *Deleting an Initiator* previously.

### Syntax

```
initiator:update [options] initiator
```



## Options

`--rename` or `-id` or `-i` *<string>*

Rename the initiator to the specified string.

`--assign` or `-a` *<string>* Assign the initiator to a group.

(See `help --all` for details on all other options.)

## Arguments

*initiator* ID or UUID of the initiator to update

## Example

This assigns the initiator to a group named `init22`.

```
> initiator:update --assign init22
```

## KDUMP COMMANDS

The `kdump` commands get information about kernel dumps.



To clear `kdump` messages, use the `system:messages -clear` command.

`kdumps` or `kdump:list`

Lists the available `kdumps`.

## Syntax

```
kdump:list [options]
```

## Options

(See `help --all` for details on all other options.)

`kdump:get`

Retrieves and stores a `kdump` (kernel dump) at a designated location, which is specified with the various output options. Use the `kdump:list` command to see the available kernel dumps.

## Syntax

```
kdump:get [options] dumpName
```



### Options

`--verbose` or `-v` Show additional information while processing.

(See `help --all` for details on all other options.)

### Arguments

*dumpName* Name of the kdump to get information for

### Example

This gets details about the kdump named `kdump1`:

```
> kdump:get kdump1
```

[kdump:delete](#)

Deletes a kdump.

### Syntax

```
kdump:delete [options] dumpName
```

### Options

(See `help --all` for details on all other options.)

### Arguments

*dumpName* Name of the kdump to delete.

### Example

This deletes `kdump1`:

```
> kdump:delete kdump1
```

## LOG COMMAND

The Log command enables you to gather log files and service report information across an ION Accelerator configuration.

### [log:servicereport](#)

Gathers files and other information into a report package for the `fio-bugreport` utility. The report package can then be sent to Fusion-io Support. You can use the `--output` options to choose the destination for the report, as shown in the examples below.



## Syntax

```
log:servicereport [options] show
```

## Options

```
--include or -I <part(s)>
```

Part(s) of the service report to include:

```
clusters, cnas, config, crm_resource_list, fio_agent_log,
fio_msrv_log, fio_saft_log, fio_scst_conf, fio_status,
inigroups, initiators, ion_default, ion_out, ionservice,
lib_fio, luns, lvdisplay, lvs, messages, nodes, pools,
ports, processes, pvdisplay, pvs, raids, scst_groups,
scst_sessions, scst_tmp, suse_studio_custom, targets,
updatectrl_log, vgdisplay, vgs, volumes
```

```
--exclude or -X <part(s)>
```

Report part to exclude (same items as listed for the `--include` option)

```
--all or -a
```

Include all report parts.

```
--detailed
```

Collect additional detailed information, if available.

```
--limit or -l <size>
```

Limit the gathered log file size to the specified amount, in KiB.

```
--browse
```

Open a view of the `bugreport` directory, if in a GUI environment.

(See `help --all` for details on all other options.)

## Arguments

```
show <part>
```

Part to include in the report. This option can be included multiple times.  
See the `--include` option for details.

## Examples

- `servicereport`  
Creates a standard service report in the user's home directory
- `servicereport -detailed`  
Creates a detailed service report in the user's home directory
- `servicereport lvdisplay pvdisplay vgdisplay`  
Reports LVM information only in the user's home directory
- `servicereport --output-usb`



Creates a standard service report and place it on the USB drive (if available)

- `servicereport --output-share domain/user@host/share`

Sends the report to a CIFS share

- `servicereport --output-scp user@host`

Sends the report through `scp` to user's home directory on the host

## LUN COMMANDS

The LUN commands enable you to create, delete, list, get information for, and update LUNs.

A LUN represents the presentation of a block device (IoMemory, RAID, or volume) by a target, which can be queried by remote initiators. Each LUN has a unique serial number that initiators use for multipath I/O discovery. An initiator group should be given if access control is required; only those initiators in the given group will be allowed access to the block device presented by the target.



ION Accelerator does not auto-discover LUNs that you create. In order to view the LUNs, you need to run `rescan-scsi-bus.sh` (OL or RHEL) or Rescan Volumes (Windows host). For SLES, run `echo - - - > /sys/class/scsi_host/host#/scan <#>`, where “#” indicates the host number based on the current configuration.

### `lun:create`

Creates a LUN.

#### Syntax

```
lun:create [options] volume initiatorGroup target(s)
```

#### Options

`--repair` Indicates repair, after servicing

`--blocksize` or `-b <integer>`

Block size for the LUN. The default is 512B.



Using a random write access pattern with 512B blocks may significantly impact available system RAM.

`--optimized-targets` or `-o`

Create the LUN with NUMA-optimized targets for the specified volume.



`--all-targets` or `-a` Create the LUN with all available targets.

(See `help --all` for details on all other options.)

### Arguments

*volume*                    Volume to export as a LUN

*initiatorGroup*        Name of the initiator group to assign the LUN to

*target*                    Target for the created LUNs. This argument may be used multiple times.

### Examples

This creates LUNs exported to `initiator_group` from all known targets, for `vol1`:

```
> lun:create -a vol1 initiator_group
```

This creates a LUN by exporting `myVolume` to the target port WWPNs  
(`21:00:00:24:ff:67:5f:60` and `21:00:00:24:ff:67:5f:61`):

```
> lun:create myVolume newgroup 21:00:00:24:ff:67:5f:60  
21:00:00:24:ff:67:5f:61
```

This creates LUNs exported to `initiator_group` from targets that are NUMA-optimized for  
`vol1`:

```
> lun:create -o vol1 initiator_group
```

### `lun:delete`

Deletes a LUN.

### Syntax

```
lun:delete [options] id(s)
```

### Options

`--volume` or `-v` *<name>* Delete all LUNs associated with a volume.

`--dry-run`                List deletions to be performed, but do not execute them.

`--group` or `-g` *<name(s)>*

                            Delete all LUNs that are members of the specified group(s).

`--node` or `--n` *<address(es)>*

                            Issue this command to one or more nodes in the cluster.

`--cluster`                Issue this command to all nodes in the cluster.





(See `help --all` for details on all other options.)

### Arguments

*id* ID, UUID, or WWPN of the LUN to delete. This argument may be used multiple times.

### Example

This deletes `testLUN`:

```
> lun:delete testLUN
```

### `lun:get`

Gets details about a LUN.

### Syntax

```
lun:get [options] id
```

### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* The ID, UUID, or WWPN of the LUN to get information for

### Example

This gets information about a LUN (from `lun:list`):

```
> lun:get
0827a41e-dac6-11e3-9679-001b213aa590-LUN0
0827a41e-dac6-11e3-9679-001b213aa590-LUN1
0827a41e-dac6-11e3-9679-001b213aa590-LUN2
...
```

### `luns` or `lun:list`

Lists the LUN IDs. To view LUNs arranged by volumes, see [Viewing LUNs by Volume](#) below.

### Syntax

```
luns [options]
```



## Options

- `--volume` or `-v` *<string>* List LUNS on the current volume.
- `--target` or `-t` *<string>* List LUNS on a specified target.
- `--uuid` or `-u` Show UUIDs instead of readable IDs.
- `--property` or `-p` *<list>* One or more properties to display:
- `id` – Generated Logical Unit number, in string format
  - `number` – Generated Logical Unit number, integer
  - `uuid` – Machine-readable ID of the LUN
  - `device_uuid` – Machine-readable ID of the device (such as `t0Lo03-Vk1e-WKpe-KLwd-hDv5-yQhr-fmxypA`)
  - `target_uuid` – Machine-readable ID of the target (such as `iqn.2007-02.com.fusionio:sn.2m232406fw:eth5`)
- `--objects` or `-o` Return objects.
- `--separator` or `-s` *<type>*  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--sort` *<property>* Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns` Do not sort the output.
- `--order-with` *<function>*  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`
- `--where` or `-w` *<function>*  
Filter by a function, if the function is true.
- `--where-not` or `-wn` *<function>*  
Filter by a function, if the function is false.
- `--used` Show only objects that are in use.
- `--not-used` or `-nu` Show only objects that are not in use.
- `--node` or `-n` *<address(es)>*  
Issue this command to one or more nodes in the cluster.



`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This lists all the available LUNs:

```
> luns
cfff0cee-fd89-11e3-8e78-009c023ca2a8-LUN0
cfff0cee-fd89-11e3-8e78-009c023ca2a8-LUN1
cfff0cee-fd89-11e3-8e78-009c023ca2a8-LUN10
cfff0cee-fd89-11e3-8e78-009c023ca2a8-LUN11
cfff0cee-fd89-11e3-8e78-009c023ca2a8-LUN12
...
```

### Viewing LUNs by Volume

You can extract a subset of LUNs for one or more volumes by using the commands below.

1. List the LUNs:

```
luns -dt --volume vol1 --volume vol2
```

2. Get the connected initiators:

```
each (luns -o --volume vol1) { $1 connected }
```

3. Extract the desired details and print out a message:

```
admin1> each (luns -o --volume vol1) { echo Lun ($1 id) connections
($1 connected) } Lun 20eb1d58-ad3d-11e2-a54c-90b11c06e8d0-LUN0
connections [21:00:00:24:ff:66:a1:e8]
```

```
Lun 23887b50-ad3d-11e2-a54c-90b11c06e8d0-LUN0 connections
[21:00:00:24:ff:66:a1:e8]
```

4. Find these connected initiators, then collect details on them. Note the nested `each` commands:

```
Admin1> each (luns -o --volume vol1) { each -dt ($1 connected) {
initiator:get $1 } }
```

```
Id |UUID |Protocol|Discovered? |Group UUID
```

```
-----
21:00:00:24:ff:66:a1:e8 |21:00:00:24:ff:66:a1:e8 |FC |false
|ea65a7f2-aa4a-11e2-bb4f-90b11c06e928
```

```
Id |UUID |Protocol|Discovered? |Group UUID
```



```
-----  
21:00:00:24:ff:66:a1:e8 | 21:00:00:24:ff:66:a1:e8 | FC | false  
|ea65a7f2-aa4a-11e2-bb4f-90b11c06e928
```

## MANAGE COMMAND

The Manage command enables Oracle Enterprise Manager (OEM) integration.

### manage:oem

Controls integration with the Oracle Enterprise Manager product through a custom plug-in.

#### Syntax

```
manage:oem [options] verb
```

#### Options

`--oms-host <string>` OMS Host (required for ENABLE)

`--oms-port <integer>` OMS Port (required for ENABLE)

`--agent-password <string>`

Agent registration password (required for ENABLE and SECURE).

#### Arguments

`verb`

One of the following actions to take:

`DISABLE`: Disable the OEM integration.

`ENABLE`: Enable the OEM integration.

`SECURE`: Secure the OEM agent with a password.

`START`: Start the OEM agent.

`STATUS`: Show the status of the OEM agent.

`STOP`: Stop the OEM agent.

`UPLOAD`: Manually trigger a metric upload.



## NETWORK COMMANDS

The Network commands enable you to see details for network addresses, including Ethernet ports, IP addresses, and subnets.

### `network:addr`

Shows network address details for components of the ION Accelerator system.

#### **Syntax**

```
network:addr [options]
```

#### **Options**

(See `help --all` for details on all options.)

#### **Example**

This shows network address details for Ethernet ports:

```
> network:addr
eth3 inet 192.168.20.49/24
eth0 inet 10.60.34.49/24
eth5 inet 192.168.30.49/24
eth6 inet 192.168.1.2/24
eth7 inet 192.168.2.2/24
```

### `network:ping`

Specifies a target host on the network to ping.

#### **Syntax**

```
network:ping [options] target
```

#### **Options**

`--count` or `-c`          Number of pings to execute (defaults to 3)

(See `help --all` for details on all options.)

#### **Arguments**

*target*                      IP address of the host to ping



## Example

This shows network address details:

```
> network:ping 192.168.20.49
PING 192.168.20.49 (192.168.20.49) 56(84) bytes of data.
64 bytes from 192.168.20.49: icmp_seq=1 ttl=64 time=0.031 ms
64 bytes from 192.168.20.49: icmp_seq=2 ttl=64 time=0.010 ms
64 bytes from 192.168.20.49: icmp_seq=3 ttl=64 time=0.017 ms

--- 192.168.20.49 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.010/0.019/0.031/0.009 ms
```

## NODE COMMANDS

The Node commands enable you to list or get information for network nodes.

### node:get

Retrieves detailed information on a node. If no node is specified, information about the current local node is returned.

### Syntax

```
node:get [options] id
```

### Options

```
--node or -n <address(es)>
```

Issue this command to one or more nodes in the cluster.

```
--cluster See --urlList above.
```

(See help --all for details on all other options.)

### Arguments

*id* The ID or UUID of the node to get information for

### Example

This gets information on ionr8i48:

```
> node:get ionr8i48
      Id  ionr8i48
      UUID 16885952
```



```
      Status Member
      Errors
Warnings
      Local false
      Slots
        IP 192.168.1.1
          192.168.2.1
      Node# 1
Chassis Monitor URL
      Gateway
        DNS
        NTP
        TZ
      State Normal
USB Status UsbNotlocal(5)
      Uptime
```

[nodes](#) or [node:list](#)

Lists available nodes.

### Syntax

```
nodes [options]
```

### Options

- `--uuid` or `-u` Show UUIDs instead of readable IDs.
- `--property <list>` One or more properties to display:
- `id` – Node ID
  - `uuid` – Node UUID
  - `number` – Number of the node; a small integer, starting from 0
  - `status` – One of the following values:
    - 0 = `STATUS_MEMBER` – Node is a member of this cluster.
    - 1 = `STATUS_NOT_A_MEMBER` – Node is a not a member of this cluster.
    - 2 = `STATUS_STANDALONE` – Node is a standalone server.
  - `ipaddr` – Cluster IP addresses. For example:

```
[192.168.1.1 192.168.2.1]
[192.168.1.2 192.168.2.2]
```



- `gateway` – IP address of the gateway
  - `timezone` – Time zone (three characters) of the node
- `--objects` or `-o`      Return objects.
- `--separator` or `-s <type>`  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--node` or `--n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster`      Issue this command to all nodes in the cluster.
- `--sort <property>`      Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns`      Do not sort the output.
- `--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`
- `--where` or `-w <function>`  
Filter by a function, if the function is true.
- `--where-not` or `-wn <function>`  
Filter by a function, if the function is false.
- `--used`      Show only objects that are in use.
- `--not-used` or `-nu`      Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Example

This lists the available nodes on the cluster and their cluster IP addresses.

```
> nodes --property ipaddr
```





## node:local

Returns the ID or UUID of the local node in a cluster. In a cluster management scenario, it may not be obvious which node you are connected to, so this command returns the information for you.

### Syntax

```
node:local [options]
```

### Options

`--uuid` or `-u` Show the UUID instead of the readable ID.

(See `help --all` for details on these options: `--display`, `--output-file`)

### Example

This shows the UUID of the local node in the cluster.

```
> node:local -uuid
33663168
```

## POOL COMMANDS

The Pool commands enable you to create, delete, list, and get information for storage pools.



The capacity reported for storage pools and volumes is closely approximated. So if the CLI reports 1500.00 GB available, the actual amount may be slightly less than that, and therefore a file of that exact capacity might not fit.

## pool:create

Creates a storage pool.

### Syntax

```
pool:create [options] id device(s)
```

### Options

`--repair` Repair, after servicing.

`--pesize` or `-p <integer>` PE (Physical Extent) size, in KiB

`--if-not-exists` or `-ne`

If an object with the given identifier already exists, skip creation.



`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* Identifier for the new pool

*device* Device to include in the pool. This argument may be used multiple times.


### Example

This creates a new storage pool called `mainpool`, from the device IDs specified, with a physical extent size (`pesize`) of 512KB:

```
> pool:create --pesize 512 mainpool fioa fiob fioc fiod fiое fiof fiog  
fioh
```

### pool:delete

Deletes a pool.

 This will destroy any volumes and user data that are currently in the storage pool.

### Syntax

```
pool:delete id(s)
```

### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* ID, UUID, or WWPN of the pool to delete. This argument may be used multiple times.

### Example

This deletes the `test1` storage pool:

```
> pool:delete test1
```



## pool:get

Gets information about a pool, including pool capacity, errors and warnings (if any), devices, free/extents, free/usable space, extent size, maximum usable capacity, free usable capacity, profile ID and name, and volume names.

### Syntax

```
pool:get [options] id
```

### Options

```
--node or -n <address(es)>
```

Issue this command to one or more nodes in the cluster.

```
--cluster
```

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* The ID, UUID, or WWPN of the pool to get information for

### Example

This gets storage pool information for the `max` pool (from `pool:list`):

```
> pool:get --display-table max
      Id  max
Capacity 2,410.00 GB
Errors
Warnings
Devices  [/dev/md3]
Extents  73,547,326
Free     51,573,966
Extent Size 32 KiB
Profile Id RAID0
Profile Name Maximum Performance
Max Usable Capacity 2,409.93 GB
Free Usable Capacity 1,689.92 GB
      UUID Jinp8p-4lFS-qOMI-QcBc-tHpt-9c4N-yHG12N
Volumes  ion48_max_1
          ion48_max_2
          ion48_max_3
          ...
```



## `pools` or `pool:list`

Lists available pools.

### Syntax

```
pools [options]
```

### Options

- `--uuid` or `-u` Show UUIDs instead of readable IDs.
- `--property` or `-p <list>`  
One or more initiator group properties to display
- `--objects` or `-o` Return objects.
- `--separator` or `-s <type>`  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all nodes in the cluster.
- `--sort <property>` Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns` Do not sort the output.
- `--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{%1 method}`
- `--where` or `-w <function>`  
Filter by a function, if the function is true.
- `--where-not` or `-wn <function>`  
Filter by a function, if the function is false.
- `--used` Show only objects that are in use.
- `--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)



## Example

This lists all available storage pools in the system:

```
> pools
max
raid10
```

## pool:update

Updates a pool.

## Syntax

```
pool:update id(s)
```

## Options

```
--rename or -id or -i <newID>
```

Rename the pool with the specified UUID or WWPN.

```
--node or -n <address(es)>
```

Issue this command to one or more nodes in the cluster.

```
--cluster
```

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

## Arguments

*id* Existing ID, UUID, or WWPN of the pool to be updated. This argument may be used multiple times.

## Example

This updates the storage pool ID from `oldtest6` to `newtest7`.

```
> pool:update newtest7 oldtest6
```



## PORT COMMANDS

The Port commands enable you to get and set information for the ports on a CNA.

### port:get

Gets information on a port.

#### Syntax

```
port:get [options] id
```

#### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

#### Arguments

*id*

The ID, UUID, or WWPN of the port to get information for

#### Example

This gets information for the port (from `port:list`):

```
> port:get eth0
```

### ports or port:list

Lists available ports.

#### Syntax

```
ports [options]
```

#### Options

`--uuid` or `-u`

Show UUIDs instead of readable IDs.

`--property` or `-p` *<list>*

One or more properties to display:

- `id` – Port ID
- `uuid` – Node UUID
- `number` – Number of the port; a small integer, starting from 0



- `status` – One of the following values:
  - 0 = `STATUS_DISCONNECTED` – Port is disconnected.
  - 1 = `STATUS_CONNECTED` – Port is connected.
- `address` – MAC address
- `MTU` – Maximum Transmission Unit for the port
- `ip_address` – IP address

`--objects` or `-o`      Return objects.

`--separator` or `-s <type>`  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.

`--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.

`--cluster`              Issue this command to all nodes in the cluster.

`--sort <property>`      Sort the output, using the specified Property name to sort on.

`--no-sort` or `-ns`      Do not sort the output.

`--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`

`--where` or `-w <function>`  
Filter by a function, if the function is true.

`--where-not` or `-wn <function>`  
Filter by a function, if the function is false.

`--used`                  Show only objects that are in use.

`--not-used` or `-nu`      Show only objects that are not in use.

(See `help --all` for details on all other options.)



## Example

This displays the names of the available ports, separated by spaces:

```
> ports -s  
eth0 eth1 eth2 eth3 eth4 eth5 eth6 eth7
```

## port:update

Updates a port.

## Syntax

```
port:update [options] id
```

## Options

`--mode <portMode>` Mode for the port: `management` or `iscsi` or `cluster`



Ports cannot be changed to or from cluster mode.

`--ip-address` or `-ip <address>`

IP address to set for the port

`--subnet-mask` or `-s <value>`

Subnet mask to set for the port

`--mode <PortMode>` Mode of the port. This can transition only between `ISCSI` and `MANAGEMENT`.

`--node` or `-n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

## Arguments

*id* The ID or UUID of the port to update

## Example

This sets the port to management mode, with an IP address of 10.11.12.13:

```
> port:update --mode management --ip-address 10.11.12.13
```





## PROFILE COMMANDS

The `profile` commands enable you to create and examine profile configurations for storage pools.

### `profile:create`

Creates a storage pool with desired characteristics. You can run `profile:create -dt` to see the available profile types.



A storage profile created in the CLI will not be reflected or available in the GUI.

### Syntax

```
profile:create [options] profile (name)
```

### Options

```
--slot or -s <number(s)>
```

Allow the use of the specified slot; by default all are allowed. Use slot or node/slot.

```
--slot-list <number(s)>
```

Use the specified list of slots [`slot# slot#`] syntax. Use slot or node/slot.

```
--slot-count or --drive-count or -d <number>
```

Number of drives to use with the profile; the default is all

```
--dry-run
```

List the actions to perform, but do not perform them.

```
--node or -n <address(es)>
```

Issue this command to one or more nodes in the cluster.

```
--cluster
```

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

```
profile
```

Profile type to use: `maximum_performance`, `reliable_performance`, `reliable_capacity`, or `direct`



## Examples

- `profile:create maximum_performance`

This creates a storage pool that emphasizes maximum performance, across all available devices.

- `profile:create reliable_performance`

This creates a storage pool that ensures reliability, across all available devices.

- `profile:create reliable_capacity`

This creates a storage pool that ensures reliability but emphasizes capacity over performance.

- `profile:create -d 2 direct`

This creates a direct (JBOD) using two of the available devices.

- `profile:create -s 1 -s 3 maximum_performance`

This creates a performance pool using the devices in slots 1 and 3.

- `profile:create maximum_performance my_pool`

This creates a performance pool and names it `my_pool`.

## [profile:delete](#)

Deletes a storage profile, also removing its owned resources.

### Syntax

```
profile:delete [options] profile
```

`--force` or `-f` Force deletion of the profile, even if volumes exist.

(See `help --all` for details on all other options.)

### Arguments

*profile* `<name>` Name of an existing storage profile (or pool) to delete

### Example

This deletes the `maximum_performance` pool:

```
> profile:delete maximum_performance
```



`profiles` or `profile:list`

Lists available profiles for storage pools.

### Syntax

```
profiles [options]
```

### Options

- `--uuid` or `-u` Show UUIDs instead of readable IDs.
  - `--property` or `-p <list>` One or more properties to display
  - `--objects` or `-o` Return objects.
  - `--separator` or `-s <type>`  
Separator between property values when printing multiple properties;  
defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
  - `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
  - `--cluster` Issue this command to all nodes in the cluster.
  - `--sort <property>` Sort the output, using the specified Property name to sort on.
  - `--no-sort` or `-ns` Do not sort the output.
  - `--order-with <function>`  
Sort the output, extracting key with this function.  
Example: `{ $1 method }`
  - `--where` or `-w <function>`  
Filter by a function, if the function is true.
  - `--where-not` or `-wn <function>`  
Filter by a function, if the function is false.
  - `--used` Show only objects that are in use.
  - `--not-used` or `-nu` Show only objects that are not in use.
- (See `help --all` for details on all other options.)



## Example

This lists the profiles, separated by spaces::

```
> profiles -s
JBOD RAID0 DAID10 RAID5
```

## PSU COMMANDS

The PSU commands get information about available power supply units.

`psu` or `psu:list`

Lists the available power supply units.

### Syntax

```
psu:list [options]
```

### Options

- `--uuid` or `-u`            Show UUIDs instead of readable IDs.
- `--node` or `-n` *<address(es)>*  
                                 Issue this command to one or more nodes in the cluster.
- `--property` or `-p` *<list>*    Properties to display (can be used multiple times)
- `--objects` or `-o`        Return objects.
- `--separator` or `-s` *<type>*  
                                 Separator between property values when printing multiple properties;  
                                 defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--cluster`                Issue this command to all instances in the cluster.
- `--sort` *<property>*        Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns`        Do not sort the output.
- `--order-with` *<function>*  
                                 Sort the output, extracting key with this function.  
                                 Example: `{ $1 method }`



`--where` or `-w` *<function>*

Filter by a function, if the function is true.

`--where-not` or `-wn` *<function>*

Filter by a function, if the function is false.

`--used` Show only objects that are in use.

`--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)

### `psu:get`

Gets details about a power supply unit.

#### **Syntax**

```
psu:get [options] id
```

#### **Options**

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all instances in the cluster.

(See `help --all` for details on all other options.)

#### **Arguments**

*id* The ID, UUID, or WWPN of the power supply unit to get information for

#### **Example**



This gets details about the power supply unit with the ID `psu1`:

```
> psu:get psu1
```



## RAID COMMANDS

The RAID commands enable you to create, delete, list, get information about, and update RAID arrays. Multiple block devices are input to create a RAID 0 or a RAID 1.

-  If you need to create a RAID 10 configuration, use the *reliable\_performance* argument with the `profile:create` command.
-  You can use `--display-flavor detailed` to show more information about the RAID table.

### [raid:create](#)

Creates a RAID array with a unique ID.

#### Syntax

```
raid:create [options] raidtype drives
```

#### Options

- `--repair` Indicates repair, after servicing.
- `--chunksize` or `-c <integer>`  
Chunk size in KB; the default is 8KB.
- `--spare` or `-s <list>` Add one or more spare drives, listed by ioMemory module.
- `--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

#### Arguments

- raidtype* RAID type: `raid0`, `raid1`, or `raid5`
- drive(s)* Drives to create the RAID from

#### Example

This creates a RAID 1 array, with a chunk size of 16, from the four ioMemory modules specified.

```
> raid:create --chunksize 16 xtra raid1 fioa fiob
```

### [raid:delete](#)

Deletes a RAID.

#### Syntax

```
raid:delete [options] id(s)
```



## Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

## Arguments

*id*

The ID, UUID, or WWPN of the RAID to delete. This argument may be used multiple times.

## Example

This deletes the RAID array named `myRAID`:

```
> raid:delete myRAID
```

## `raid:get`

Gets details about a RAID, including capacity, chunk size, RAID device path, errors and warnings, other device paths, spares and faults (if available), rebuild percent, RAID state and status, sync status, and UUID.

## Syntax

```
raid:get [options] id
```

## Options

`--boot` or `-b`

Include only boot devices.

`--rescan` or `-r`

Force rescan of boot devices.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

## Arguments

*id*

The ID, UUID, or WWPN to get information for

## Example

This gets information for the RAID array named `md0` (from `raid:list`):



```
> raid:get md0
      Id  md0
      Type  raid1
      Capacity  1,204.87 GB
      Chunk Size
      RAID Device  /dev/md0
      Errors
      Warnings
      Devices  /dev/fiof
              /dev/fiod
      Spares  []
      Faults  []
      Rebuild  100%
      State  clean
      Status
      Sync  idle
      UUID  9e44f89f-4890-3df1-4bf0-e4723f85c54c
```

### raids or `raid:list`

Lists the RAID IDs.

#### Syntax

```
raids [options]
```

#### Options

- `--boot` or `-b`            Include only boot devices.
- `--rescan` or `-r`           Force rescan of boot devices.
- `--uuid` or `-u`            Show UUIDs instead of readable IDs.
- `--property` or `-p <list>` One or more properties to display:
  - `id` – RAID ID
  - `uuid` – RAID UUID
  - `chunksize_kb` – RAID chunk size in KB
  - `devices` – IDs of the ioDrives to RAID together. For example:  

```
[/dev/md0 /dev/md1]
[/dev/fiob /dev/fioa]
[/dev/fioe /dev/fioc]
[/dev/fiof /dev/fiod]
```





- `raidtype` – One of the following values:
  - 0 = RAID 0
  - 1 = RAID 1
  - 2 = RAID 10
  - 3 = RAID 5
- `status` – Current status of the RAID
- `rebuild_pct` – Current progress percentage toward completing the RAID rebuild

`--objects` or `-o` Return objects.

`--separator` or `-s <type>`

Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.

`--node` or `--n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

`--sort <property>`

Sort the output, using the specified Property name to sort on.

`--no-sort` or `-ns`

Do not sort the output.

`--order-with <function>`

Sort the output, extracting key with this function.

Example: `{ $\$1$  method}`

`--where` or `-w <function>`

Filter by a function, if the function is true.

`--where-not` or `-wn <function>`

Filter by a function, if the function is false.

`--used`

Show only objects that are in use.

`--not-used` or `-nu`

Show only objects that are not in use.

(See `help --all` for details on all other options.)





## RULES COMMANDS

The `rules` commands manipulate and get information for rule contexts in the CLI.

### `rules:compile`

Compiles rule contexts.

#### **Syntax**

```
rules:compile [options]
```

#### **Options**

```
--context or -c <string>
```

Name of the rule context

(See `help --all` for details on all other options.)

### `rules:delete`

Deletes rule contexts.

#### **Syntax**

```
rules:delete [options] contextName(s)
```

#### **Options**

(See `help --all` for details on all other options.)

#### **Arguments**

`contextName` Name of the rule context to delete. This argument can be used multiple times.

### `rules:facts`

Lists or counts the known facts.

#### **Syntax**

```
rules:facts [options]
```

#### **Options**

```
--count Return the number of facts.
```

```
--context or -c <string>
```

Name of the rule context



(See `help --all` for details on all other options.)

### `rules:insert`

Inserts objects into working memory.

#### **Syntax**

```
rules:insert [options] object(s)
```

#### **Options**

`--run` or `-r`                    Run rules after inserting objects.

`--context` or `-c <string>`

                                  Name of the rule context

(See `help --all` for details on all other options.)

#### **Arguments**

`object`                            Object to insert into working memory. This argument can be used multiple times.

### `rules:reset`

Resets rules.

#### **Syntax**

```
rules:reset [options]
```

#### **Options**

`--context` or `-c <string>`

                                  Name of the rule context

(See `help --all` for details on all other options.)

### `rules` or `rules:rules`

Shows information about rule contexts.

#### **Syntax**

```
rules:rules [options]
```

#### **Options**

`--all` or `-a`                    Show information about all known rule contexts.



`--verbose` or `-v` Show more details.  
`--context` or `-c <string>`  
Name of the rule context

(See `help --all` for details on all other options.)

### Example

This lists the rules:

```
> raids -s  
md0 md1 md2 md3
```

### rules:run

Runs rules and returns the number of rules that were fired.

### Syntax

```
rules:run [options]
```

### Options

`--max` or `-m <number>`  
Maximum number of rules to fire (defaults to all of them)

`--timeout` or `-s <seconds>`  
Timeout for rule execution, in seconds

`--context` or `-c <string>`  
Name of the rule context

(See `help --all` for details on all other options.)



## SAFT COMMANDS

The `saft` commands list objects in the system and manage SAFT service access.

### `saft:list`

Lists a selected type of objects.

#### **Syntax**

```
saft:list [options] type
```

#### **Options**

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--objects` or `-o` Return objects.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

#### **Arguments**

*type* Type of object to list: `bus`, `chassis`, `cluster`, `cna`, `cpu`, `drive`, `fan`, `inigroup`, `initiator`, `lun`, `node`, `pool`, `port`, `psu`, `raid`, `target`, `temp`, `volume`

### `saft:url`

Gets or sets the URL used to connect to the SAFT service.

#### **Syntax**

```
saft:url [options] url
```

#### **Options**

`--host` *URL* specifies the only host to connect to.

`--port` *URL* specifies the only port to connect to.

`--test` Test the URL before using it.

(See `help --all` for details on all other options.)



## Arguments

*URL* URL (or host or port, depending on the option set) used to connect to SAFT.

## SERVICE COMMAND

The `service` command gets the state of CLI services. The CLI may start certain services while operating, such as a zeroconf daemon. This command lists the services that are running, or have run at any point, and their current states.

### `service:services`

Lists CLI service states.

### Syntax

```
service:services [options]
```

### Options

(See `help --all` for details on all other options.)

## SHELL COMMANDS

See [Appendix: Shell Commands for Scripting](#).

## SNMP COMMANDS

The SNMP commands get and change SNMP configuration, as well as download available MIB files. For information types, see the `snmp:update` command.

### `snmp:get`

Gets SNMP information.

### Syntax

```
snmp:get [options]
```



## Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all instances in the cluster.

(See `help --all` for details on all other options.)

## Example

This gets SNMP information:

```
> snmp:get
    Enabled  true
    Location Server Room
Client Address 127.0.0.1
    Community public
    Contact  Sysadmin (root@localhost)
Trap Addresses []
Trap Community
```

## [snmp:mibs](#)

Downloads a .zip file containing the MIBs defined on the target ION Accelerator server.

## Syntax

```
snmp:mibs [options]
```

## Options

`--host` *<servername>* ION Accelerator server to download from

(See `help --all` for details on all other options.)

## [snmp:update](#)

Changes the SNMP information.

## Syntax

```
snmp:update [options]
```

## Options

`--client-address` *<string>*

Address of the client

`--community` *<string>* SNMP Community





```
--contact <string>    Contact information
--location <string>   Location information
--trap-address <string>
                        Set the trap destination address.
--trap-community <string>
                        Community for traps
--enable              Enable SNMP.
--disable             Disable SNMP.
--node or -n <address(es)>
                        Issue this command to one or more nodes in the cluster.
--cluster             Issue this command to all instances in the cluster.
```

(See `help --all` for details on all other options.)

### Example

This updates the current SNMP information to include a contact name of "sysadmin(root@localhost)" and a location of "server room":

```
> snmp:update --contact sysadmin(root@localhost) --location server room
```

## SOFTWARE COMMANDS

The Software commands enable you to update, validate and check history for the ION Accelerator software.

For step-by-step instructions on updating and reverting software versions via the CLI, refer to [Quick Start Tasks: Software Updates](#) earlier in this guide.



## soft:apply

Applies the software update in the drop-box to the ION Accelerator appliance. The drop-box is a temporary location for the pending software update file.



After the upgrade finishes, you must log out of the CLI. Although a console message will state that no restart is necessary, the system will automatically reboot.

### Syntax

```
soft:apply [options]
```

### Options

`--no-wait` Do not wait for the result of the apply/restart; return immediately after requesting.

`--node` or `-n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This applies the software update from the drop-box to the ION Accelerator appliance.

```
> soft:apply
```

## soft:dropbox

Validates the software update in the drop-box and returns information about it.

### Syntax

```
soft:dropbox [options]
```

### Options

`--node` or `-n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Return Values

The following values are returned:



- `retval` – Integer, with one of the following:
  - `0` = `SUCCESS` – The update software in the drop-box is valid.
  - `-801` = `UPDATE_BAD_SIGNATURE` – The update software is invalid; it has a bad signature.
  - `-802` = `UPDATE_BAD_METADATA` – The update software is invalid; it has bad metadata.
  - `-805` = `UPDATE_EMPTY_DROPBOX` – No update software was found in the drop-box.
- `release_date` – Release date of the software update
- `version` – Version number of the software update
- `build_number` – Build number of the software update
- `patch_level` – Patch level number of the software update, if any
- `description` – Comments about the software update file
- `hotfix_id` – Identifies the hot fix used with this update, if any
- `reboot_required` – True if a reboot is required after installing the update; False otherwise
- `estimated_update_time_mins` – Estimated number of minutes needed for the software update to complete

### Example

This gets information about the software update file currently in the drop-box.

```
> soft:dropbox
```

### [soft:history](#)

Displays software update history, compared to the current version. A `history` list is returned for each software update that occurred.

### Syntax

```
soft:history [options]
```

### Options

```
--node or -n <address(es)>
```

Issue this command to one or more nodes in the cluster.

```
--cluster
```

Issue this command to all nodes in the cluster.



(See `help --all` for details on all other options.)

### Return Values

The following values are returned in each history list:

- `update_date` – Date this software update was applied
- `release_date` – Release date of the software update
- `version` – Version number of the software update
- `build_number` – Build number of the software update
- `patch_level` – Patch level number of the software update, if any
- `description` – Comments about the software update file
- `hotfix_id` – Identifies the hot fix used with this update, if any

### Example

This gets the software update history, in list format:

```
> soft:history -display-list
```

### `soft:revert`

Reverts the software update in the drop-box.

### Syntax

```
soft:revert [options]
```

### Options

`--no-wait` Do not wait for the result of the apply/restart; return immediately after requesting it.

`--node` or `-n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on these options: `--display`, `--nodelist`, `--url`, `--urlList`, `--output-file`, `--parallel`)

### Example

This reverts the software update in the drop-box to whatever the previous software version is:

```
> soft:revert
```



## soft:update

Uploads an update package and then applies it to the ION Accelerator system.

### Syntax

```
soft:update [options]
```

### Options

- `--no-wait` Do not wait for the result of the apply/restart; return immediately after requesting.
- `--quiet` or `-q` Do not print status messages.
- `--file` or `-f <filename>` File containing the update package
- `--web` or `-w <URL>` Web address (URL) to download the update package from
- `--noparts` Force upload of update as one file (for older ION systems).
- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This updates the ION Accelerator software while suppressing status messages:

```
> soft:update -q
```

## soft:upload

Uploads a saved ION Accelerator software update file (.IOP, from Fusion-io) to the drop-box area.

### Syntax

```
soft:upload [options]
```

### Options

- `--quiet` or `-q` Do not print status messages.
- `--file` or `-f <file>` File containing the update package
- `--web` or `-w <URL>` URL to download the update package from
- `--noparts` Force upload of update as one file (for older ION systems).
- `--node` or `-n <address(es)>`



Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This uploads the `iontest.iop` software update file from the specified web address to the drop-box area, so it can be installed with the `soft:apply` command:

```
> soft:upload --file iontest.iop --web https://exampledownload.fusionio.com
```

### `soft:version`

Returns the current software version information.

### Syntax

```
soft:version [options]
```

### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This returns the software version:

```
> soft:version
Version 2.4.0
      Build Number 119
      Hotfix Id ""
      Update Applied
      Release Date "Tue Jun 3 20:06:07 MDT 2014"
      Description "ION Accelerator"
      Update State COMPLETE
Estimated Update Time 0
      Reboot Required false
      Reason
```



## soft:versions

Displays the ION software update history.

### Syntax

```
soft:versions [options]
```

### Options

--node or -n <address(es)>

Issue this command to one or more nodes in the cluster.

--cluster

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This returns the software update history in a table format:

```
> soft:versions --display-table
```

Version	Build	Hotfix	Update Applied	Released	Description	State	Reboot	Reason
2.4.0	59		Mon Apr 28 15:08:31 2014	Sat Apr 26 22:16:47 MDT 2014	ION Accelerator			
2.4.0	92		Thu May 15 08:08:09 2014	Wed May 14 17:21:00 MDT 2014	ION Accelerator		true	
2.4.0	112		Thu May 29 16:27:10 2014	Thu May 29 18:45:34 MDT 2014	ION Accelerator		true	
2.4.0	114		Fri May 30 13:49:48 2014	Fri May 30 05:57:33 MDT 2014	ION Accelerator		true	
2.4.0	119		Tue Jun 3 13:20:24 2014	Tue Jun 3 20:06:07 MDT 2014	ION Accelerator		true	

## SSH COMMANDS

### ssh:close

Closes SSH tunnels.

### Syntax

```
ssh:close [options] host(s)
```

### Options

--all or -a

Close all tunnels.

(See `help --all` for details on these options: `--display`, `--output-file`)



## Arguments

*host* Host to close tunnels to. This argument can be used multiple times.

## Example

This closes all active SSH tunnels.

```
ssh:close --all
```

## ssh:exec

Executes a command over SSH.

## Syntax

```
ssh:exec [options] command(s)
```

## Options

--user or -u *<string>* User name

--password or -p *<string>*

Password

--port *<integer>* Port ID (defaults to 22)

--host *<string>* Host to connect to

(See `help --all` for details on all other options.)

## Arguments

*command* Command to execute. This argument can be used multiple times.

## ssh:scpput

Copies a file to a remote system.

## Syntax

```
ssh:scpput [options] localFile destination
```

## Options

--user or -u *<string>* User name for the remote system login

--password or -p *<string>*

Password for the remote system login

--port *<string>* Port to use for the file copy (defaults to 22)





`--host <string>` Remote host to connect to

(See `help --all` for details on all other options.)

### Arguments

`localFile` Local file to copy

`destination` Remote filename or directory. Most (but not all) systems will accept a path and file name here. If this argument is left blank, the local filename will be used.

### Examples

```
> scpput config.xml someone@srv
```

This copies `config.xml` from local directory to the home directory of someone on host `srv`.

```
> scpput /tmp/config.xml someone@srv:/tmp
```

This copies `/tmp/config.xml` to the `/tmp` directory on host `srv`.

```
> scpput config.xml someone@srv:/tmp/my_config.xml
```

This copies `config.xml` to `/tmp/my_config.xml` on host `srv`.

```
> scpput config.xml someone@srv --password pass
```

or

```
> scpput config.xml someone:pass@srv
```

This copies `config.xml` to the home directory of someone on host `srv`, using password `pass`.

### ssh:sftp

Executes SFTP (SSH File Transfer Protocol) commands over SSH. The initial call to this command will start an SFTP session against the host. Successive commands will reuse the same session information, until SFTP disconnect is invoked. This allows the user to perform a series of commands, such as changing local and remote directories, followed by `put` and `get`.

### Syntax

```
ssh:sftp [options] command(s)
```

### Options

`--quiet` or `-q` Hide the displayed progress.

`--from <long>` Byte position to begin transfer (for the `get` command)



`--disconnect` Disconnect after executing the command.  
`--user` or `-u <string>` User name for the remote system  
`--password` or `-p <string>`  
Password for the remote system  
`--port <string>` Port to use for the file copy (defaults to 22)  
`--host <string>` Host to connect to

(See `help --all` for details on these options: `--display`, `--output-file`)

## Arguments

*command* Command to execute. This argument can be used multiple times.  
Any of the following values can be used:

- CD: Change the remote directory.
- CHGRP: Change the file group `<groupid:int> <file> [file]*`
- CHOWN: Change the file owner `<ownerid:int> <file> [file]*`
- DIR: List the remote directory contents `<directory>`.
- DISCONNECT: Disconnect the currently cached session.
- GET: Get a remote file `<file> [local]`.
- GET\_APPEND: Append to a local `<file> [local]`.
- GET\_RESUME: Resume get `<remote file> [local]`.
- LCD: Change the local directory `<directory>`.
- LDIR: List the local directory contents `<directory>`.
- LLS: List the local directory contents `<directory>`.
- LN: Link a remote file `<current> <new>`.
- LPWD: Print the local directory.
- LS: List the remote directory contents `<directory>`.
- LSTAT: Retrieve information about a local file `<file>`.
- MKDIR: Make a remote directory `<directory> [directory]*`.
- PUT: Copy to remote `<local> [remote]`.
- PUT\_APPEND: Append to remote `<local> [remote]`.



PUT\_RESUME: Resume copy to remote <local> [remote].

PWD: Print the remote directory.

READLINK: Print the target of a link <link>.

REALPATH: Print the full path of a file <file>.

RENAME: Rename a remote file <current> <new>.

RM: Remove a remote file <file> [file]\*.

RMDIR: Remove a remote directory <directory>+.

STAT: Retrieve information about a remote file <file>.

SYMLINK: Link a remote file <current> <new>.

VERSION: Print the remote SSH version.

*args* Command arguments to use

### Example

This starts an SFTP session against the NodeX host and then issues the `pwd` command (print the remote directory) and the `stat Testfile` command (get information about Testfile):

```
ssh:sftp --NodeX pwd stat Testfile
```

### ssh:tunnels

Lists the active SSH tunnels.

### Syntax

```
ssh:tunnels [options]
```

### Options

(See `help --all` for details on all other options.)

### Example

This lists the active tunnels in display table format:

```
> ssh:tunnels -display-table
```



## SYSTEM COMMANDS

### system:keys

Sets up interconnect key pairs.

#### Syntax

```
system:keys [options] verb(s) key
```

#### Options

- `--force` or `-f` Force the creation or removal of specified keys
- `--user` or `-u <username>`  
Username to use for another node
- `--password` or `-p <string>`  
Password to use for the remote user
- `--map-password` or `-mp <string>`  
Password map of format (node=password, node=password)
- `--type <KeyPairType>`  
Key pair type: `dsa` (Digital Signature Algorithm) or `rsa` (public-key encryption)

(See `help --all` for details on all other options.)

#### Arguments

- `verb` One of the following values (this argument may be used multiple times):
  - `AUTHORIZE`: Authorizes a new key pair
  - `CHECK`: Verifies the key configuration
  - `CREATE`: Creates a new key pair
  - `PUSH`: Pushes keys to other nodes in the cluster
  - `MOVE`: Removes the current key pair
- `key` Public key to authorize



## system:maintenance

Sets maintenance mode on or off. Maintenance mode disables all storage access, but management tasks are available. Entering maintenance mode is useful when hardware needs to be replaced in a server, for example.

### Syntax

```
system:maintenance [options] mode
```

### Options

`--wait` or `-w` *<integer>* Maximum seconds to wait for the system to respond to the update request. Use `--wait 0` to return immediately.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*mode* `on` to enter maintenance mode; `off` to exit maintenance mode

### Example

This turns on maintenance mode so hardware changes can be made safely in the ION Accelerator system:

```
> system:maintenance on
```

## system:messages

Displays ION Accelerator system messages, if any, such as system alerts.

### Syntax

```
system:messages [options]
```

### Options

`--clear` or `-c` Clear the system messages after displaying them.

(See `help --all` for details on all other options.)

### Example

This captures ION Accelerator system messages and saves them in the `messages1.txt` file:

```
system:messages -output-file messages1.txt
```



## system:restart

Restarts a designated node.

### Syntax

```
system:restart [options]
```

### Options

`--wait` or `-w` *<integer>* Maximum seconds to wait for the system to respond to the update request. Use `--wait 0` to return immediately.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This restarts `node2` in the system:

```
> system:restart --node node2
```

## system:setup

Configures a selected aspect of the ION system by invoking the corresponding Setup dialog. This temporarily exits the CLI and returns when the dialog is closed. For details on the dialogs and their respective fields, refer to the *ION Accelerator GUI Guide*.



Before using this command, be sure you have put the affected nodes into maintenance mode (see the [system:maintenance](#) command).

### Syntax

```
system:setup setup
```

### Arguments

*setup*

One of the following types of setup to perform:

`lan` – Invokes the LAN configuration screen

`cluster` – Invokes the Cluster Configuration screen

`timezone` – Invokes the Timezone screen

`password` – Invokes the Password Screen

`resetios` – Shuts down `fio-agent` and `fio-msrv`, resets the



database, and restarts the services

`resetvols` – Removes constraints on a failed node so failover can occur

### Example

This enables you to configure the time zone for the server at the console:

```
system:setup timezone
```

See also [Quick Start Tasks: Changing Node Names and IP Addresses](#) earlier in this guide.

### system:shutdown

Shuts down a designated node.

### Syntax

```
system:shutdown
```

### Options

`--wait` or `-w <integer>` Maximum seconds to wait for the system to respond to the update request. Use `--wait 0` to return immediately.

`--node` or `-n <address(es)>`

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Example

This shuts down `nodeB` in the system:

```
> system:shutdown --node nodeB
```

### system:status

Shows the configuration and maintenance state.

### Syntax

```
system:status
```

### Options

(See `help --all` for details on all other options.)



## Example

This shows the current configuration and maintenance state for the ION Accelerator system:

```
> system:status
```

## TARGET COMMANDS

The Target commands represent a protocol-specific endpoint for SCSI communication. Initiators connect to targets via a discovered network address. See also [Port Commands](#).

### target:create

Manually creates a target, specifying a name and optionally specifying a UUID for the target. The UUID field accepts WWPNs (such as `f8:e9:d2:c3:b4:a5:f6:e7`), IQNs (such as `iqn.1992-01.com.example:storage.disk2.sys1.xyz[3]`), and GIDs (such as `0ab0:0ab0:0ab0:0ab0:0ab0:0ab0:0ab0:0ab0`).

### Syntax

```
target:create [options] id
```

### Options

- `--uuid <string>` Specify the exact UUID to use for the created target, instead of generating one.
- `--node` or `-n <string>` Route target creation to another node in the cluster.
- `--if-not-exists` or `-ne`  
If an object with the given identifier already exists, skip creation.

(See `help --all` for details on all other options.)

### Arguments

*id* Human-readable target identifier

### Example

This creates `target2` using the Fibre Channel protocol:

```
> target:create -protocol FC target2
```





## target:delete

Deletes a target.

### Syntax

```
target:delete [options] id(s)
```

### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* ID, UUID, or WWPN of target to delete. This argument may be used multiple times.

### Example

This deletes the target named `testTarget`:

```
> target:delete testTarget
```

## target:get

Gets details about a target.

### Syntax

```
target:get [options] id
```

### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* The ID, UUID, or WWPN of the target to get information for



## Example

This gets details about the eth3 target (from `target:list`):

```
> target:get target
Id eth3
  UUID iqn.2007-02.com.fusionio:sn.2m232406fw:eth3
  Protocol iSCSI
  Enabled true
  State Online
  Err/Warn []
Statistics
```

[targets](#) or [target:list](#)

Lists the target IDs.

## Syntax

```
targets [options]
```

## Options

- `--uuid` or `-u` Show UUIDs instead of readable IDs.
- `--property` or `-p <list>` One or more properties to display:
  - `id` – Target ID
  - `uuid` – Target UUID
  - `protocol` – FC, IB, or iSCSI
- `--objects` or `-o` Return objects.
- `--separator` or `-s <type>`  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all nodes in the cluster.
- `--sort <property>` Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns` Do not sort the output.
- `--order-with <function>`



Sort the output, extracting key with this function.

Example: `{ $1 method }`

`--where` or `-w` *<function>*

Filter by a function, if the function is true.

`--where-not` or `-wn` *<function>*

Filter by a function, if the function is false.

`--used` Show only objects that are in use.

`--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)

## Statistics

Using `--property FC`, the following Fibre Channel statistics are displayed:

- `tx_frames`
- `tx_words`
- `rx_frames`
- `rx_words`
- `lip_count`
- `nos_count`
- `error_frames`
- `dumped_frames`
- `link_failure_count`
- `loss_of_sync_count`
- `loss_of_signal_count`
- `invalid_tx_word_count`
- `invalid_crc_count`

## Example

This lists the available targets, separated by spaces:

```
> targets  
eth3 eth5
```

[target:update](#)

Updates a target.

## Syntax

```
target:update [options] id
```

## Options



- `--issue-lip` or `-l` Issue a LIP to the target.
- `--rename` or `--id` or `-i` *<string>*  
Set a new ID.
- `--remove-id` or `-r` Remove the ID assigned to a target, reverting to its natural identifier.
- `--all` or `-a` Issue the command against all targets.
- `--node` or `-n` *<address(es)>*  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id* ID or UUID of the LUN to update

### Examples

```
> target:update --all --issue-lip
```

This issues LIP to all targets on the node.

```
> target:update --all --remove-id
```

This removes any aliases applied to the targets on this node.

```
> target:update --cluster --all --issue-lip
```

This issues LIP to all targets on all nodes in the cluster.

```
> target:update 13:32:45:32 mytarget
```

This changes the ID of target 13:32:45:32 to `mytarget`.



## TEMP (TEMPERATURE) COMMANDS

The Temp commands get information about temperature sensors.

### temp:get

Gets information on a temperature sensor.

#### Syntax

```
temp:get [options] id
```

#### Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

#### Arguments

*id* Human-readable target identifier

### temps or temps:list

Lists available temperature sensors.

#### Syntax

```
temps [options]
```

#### Options

`--uuid` or `-u` Show UUIDs instead of readable IDs.

`--property` or `-p` *<list>* One or more properties to display

`--objects` or `-o` Return objects.

`--separator` or `-s` *<type>*

Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.



`--sort <property>` Sort the output, using the specified Property name to sort on.

`--no-sort` or `-ns` Do not sort the output.

`--order-with <function>`

Sort the output, extracting key with this function.

Example: `{$1 method}`

`--where` or `-w <function>`

Filter by a function, if the function is true.

`--where-not` or `-wn <function>`

Filter by a function, if the function is false.

`--used` Show only objects that are in use.

`--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)

## VIEW COMMAND

### view:graph

Creates a configuration graph of the ION Accelerator system. The possible elements to include in the graph are listed in the `--exclude` option below.



To create configuration graphs, you must have the open-source Graphviz Dot tool available in your command path.

### Syntax

```
view:graph [options] configuration
```

### Options

`--format` or `-f <format>`

Output format; defaults to SVG. Other output formats include these:

`bmp, canon, cmap, cmapx, cmapx_np, dot, emf, emfplus, eps, fig, gd, gd2, gif, gv, imap, imap_np, ismap, jpe, jpeg, jpg, metafile, pdf, plain, png, ps, ps2, svg, svgz, tif, tiff, tk, vml, vmlz, vrml, wbmp, xdot`



`--exclude` or `-e` *<format>*

Exclude one or more of the following element types from the graph:  
`boot_drives`, `boot_raids`, `bus`, `chassis`, `cluster`, `cna`, `cpu`,  
`drive`, `fan`, `inigroup`, `initiator`, `lun`, `node`, `numa`, `pool`, `port`,  
`profile`, `psu`, `raid`, `snmp`, `software`, `target`, `temp`, `volume`

`--from` *<domaintype>* Specify the start of a range of elements. See the above list for the  
`--exclude` option.

`--to` *<domaintype>* Specify the end of a range of elements. See the above list for the  
`--exclude` option.

`--browse` or `-b` Displays the graph in a browser, if available on your platform.

`--input-file` *<file>* Load configuration from a file.

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

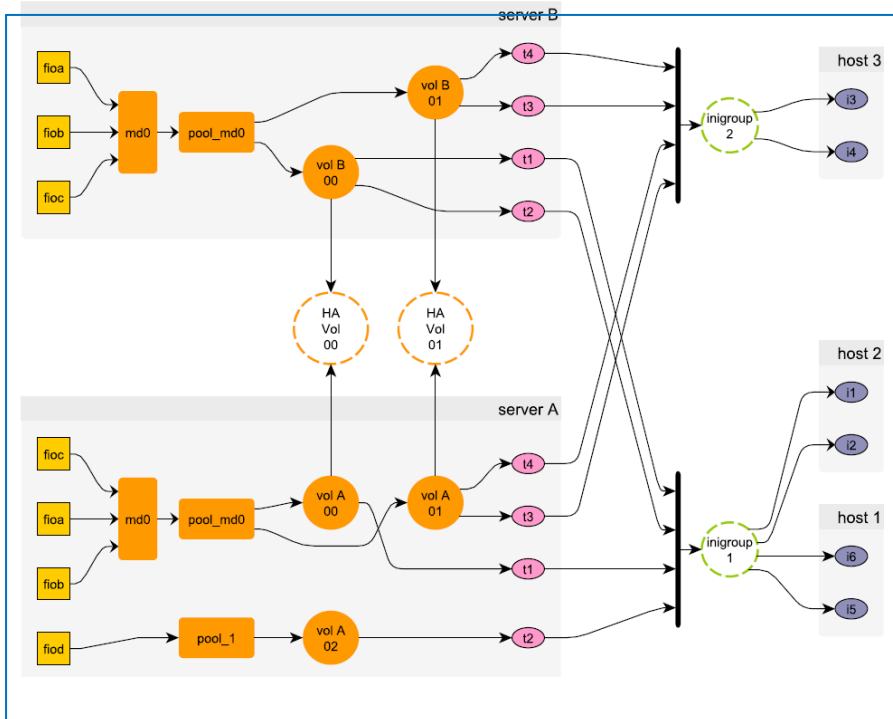
*configuration* Configuration value to be used, either as a String (XML), a Domain, a  
DomainSet, or a ResultMap

### Example

This traverses all nodes in the cluster to gather configuration elements (in parallel) and outputs  
them to the `mychart.jpg` file.

```
> view:graph --cluster --parallel --format jpg --output mychart.jpg
```

The example below (enhanced) shows a sample graph from a clustered configuration (HA).



You can also capture the current configuration into a variable, and reuse it:

```
admin@url> cfg = (config --cluster --parallel)
admin@url> graph --format dot $cfg
admin@url> graph --output configuration.svg $cfg
admin@url> graph --format pdf --output configuration.pdf $cfg
```

## VOLUME COMMANDS

The Volume commands model storage to be presented as a LUN of a target. A volume of specified capacity is allocated from a pool. Volumes can be expanded after creation and are replicated across cluster nodes for high availability, if in ION Accelerator HA mode.



The capacity reported for storage pools and volumes is closely approximated. So if the CLI reports 1500.00 GB available, the actual amount may be slightly less than that, and therefore a file of that exact capacity might not fit.

### volume:create

Creates a volume.

### Syntax

```
volume:create [options] id capacity_gb pool
```





## Options

<code>--repair</code>	Repair, after servicing.
<code>--minor &lt;integer&gt;</code>	Supply a code for the minor version, to be used with the <code>--repair</code> option.
<code>--if-not-exists</code> Or <code>-ne</code>	If an object with the given identifier already exists, skip creation.
<code>--local</code> or <code>-l</code>	Create a volume on this node only, if in a cluster.
<code>--node</code> or <code>-n &lt;address(es)&gt;</code>	Issue this command to one or more nodes in the cluster.
<code>--cluster</code>	Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

## Arguments

<code>id</code>	Identifier for the new volume
<code>capacity_gb</code>	Size of the volume to be created, in GB or percentage of available space
<code>pool</code>	Pool to create the volume in

## Examples

This creates a volume on the `mynode` host, called `newvolume`. It has a capacity of 8GB, using the `xperfpool` storage pool:


```
> volume:create --url mynode newvolume 8 xperfpool
```

This creates a series of 10 volumes, each with 50GB, belonging to the `MYPOOL` storage pool:

```
> each (seq 10) { volume:create vol${1} 50 MYPOOL }
```

## volume:delete

Deletes a volume by ID or UUID. In HA mode, this command also deletes associated volumes on other cluster nodes.

 This will destroy any user data currently on the volume, as well as initiator access to it. Before deleting a volume, make sure there is no initiator traffic on the volume.

## Syntax

```
volume:delete [options] volume(s)
```



## Options

(See `help --all` for details on all other options.)

## Arguments

*volume* ID or UUID of the volume to delete. This option can be used multiple times.

## Example

This deletes the volume named `testVol`:

```
> volume:delete testVol
```

## volume:get

Gets a variety of information on a volume.

## Syntax

```
volume:get [options] id
```

## Options

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster` Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

## Arguments

*id* The ID, UUID, or WWPN of the volume to get information for

## Example

This gets information for the `ion48_max1` volume (from `volume:list`):

```
> volume:get ion48_max1
      Id  ion48_max_1
      Device  /dev/drbd33
      Errors
      Warnings
      T10 Id  2f05ca9e-ion48_max_1
          USN  2f05ca9e
      Capacity  50.00 GB
          Pool  max
          Status  Connected
      Bytes Read  48,971,776
```



```
Bytes Written 0
             UUID mcl0vs-aBLF-xRf0-eNxW-yClb-Q6UN-kcuYrd
             Nodes ionr8i48
                   ionr8i49
```

[volumes](#) or [volume:list](#)

Lists available volumes.

### Syntax

```
volumes [options]
```

### Options

- `--uuid` or `-u` Show UUIDs instead of readable IDs.
- `--property` or `-p <list>` One or more properties to display:
  - `id` – Volume ID
  - `uuid` – Volume UUID
  - `capacity_kb` – Capacity of the volume in KB
  - `device` – Device where the volume resides
  - `pool` – Storage pool where the volume resides
  - `nodes` – HA nodes where the volume resides; “\*” for all
  - `status` – Status of the volume (Connected or Disconnected)
- `--objects` or `-o` Return objects.
- `--separator` or `-s <type>`  
Separator between property values when printing multiple properties; defaults to `tab`. Valid values are `space`, `comma`, and `tab`.
- `--node` or `-n <address(es)>`  
Issue this command to one or more nodes in the cluster.
- `--cluster` Issue this command to all nodes in the cluster.
- `--sort <property>` Sort the output, using the specified Property name to sort on.
- `--no-sort` or `-ns` Do not sort the output.
- `--order-with <function>`  
Sort the output, extracting key with this function.



Example: `{$1 method}`

`--where` or `-w` *<function>*

Filter by a function, if the function is true.

`--where-not` or `-wn` *<function>*

Filter by a function, if the function is false.

`--used`

Show only objects that are in use.

`--not-used` or `-nu` Show only objects that are not in use.

(See `help --all` for details on all other options.)

### Example

This displays the volumes available on the current host, separated by spaces:

```
> volumes -s
```

```
ion48_max_1 ion48_max_2 ion48_max_3 ion48_max_4 ion48_max_5 ion48_max_6
```

[volume:update](#)

Updates a volume.

### Syntax

```
volume:update [options] id
```

### Options

`rename` or `--id` or `-i` *<string>*

Rename the volume to the specified string (not allowed in HA mode).

`--capacity_gb` or `-c` *<float>*

Set the capacity in GB (capacity can only be increased).

`--node` or `-n` *<address(es)>*

Issue this command to one or more nodes in the cluster.

`--cluster`

Issue this command to all nodes in the cluster.

(See `help --all` for details on all other options.)

### Arguments

*id*

ID or UUID of the volume to update



### Example

This increases the capacity of myVolume to 100GB:

```
> volume:update -capacity_gb 100 myVolume
```



## Appendix A: Shell Commands for Scripting

---

The Shell command group contains core commands similar to the functions commonly found in Unix shells. Commands in this group include piping and routing output, formatting output, control structures (looping and closures), and miscellaneous functions.

These shell commands can be useful for running scripts to manage or configure ION Accelerator systems.



These commands do *not* manipulate the conventional Linux file system. While they have syntax and names like certain Linux commands, they manipulate the CLI's environment tree, which is an in-memory structure. You must use the Save command to keep whatever changes you make. When you set options or passwords into the environment (or declare sub-environments to reach other systems), those changes are held in memory only until you save them. The entire environment is kept in the user's home directory in a file. Passwords stored there are scrambled but should not be considered to be secure.

### shell:auth

Display or change authorization information in the current environment.

#### Syntax

```
shell:auth [options]
```

#### Options

- `--host <string>` Store the host.
- `--user` or `-u <string>` Store the username, or `user.[host]` if the host option is supplied.
- `--password` or `-p <string>`  
Store the password, or `password.[host]` if the host option is supplied.

(See `help --all` for details on all other options.)



## shell:cat

Displays the content of a file or URL.

### Syntax

```
shell:cat [options] paths OR URLs
```

### Options

--n                                   Number the output lines, starting at 1.

### Arguments

*paths* OR *URLs*                   List of file paths or URLs to display, separated by whitespace (use for STDIN)

### Example

This displays the contents of both `file1` and `file2`, with numbered lines for each:

```
shell:cat file1 file2
```

## shell:cd

Changes the current environment path.

### Syntax

```
shell:cd [options] path
```

### Options

(See `help --all` for details on all other options.)

### Arguments

*path*                                   Desired environment path (root if not provided)

## shell:clear

Clears the console buffer.

### Syntax

```
shell:clear
```



## shell:compare

Uses an operator to compare two arguments.

### Syntax

```
shell:compare [options] left operator right
```

### Options

`--not` Negate the logic of the operator.

### Arguments

*left* Left argument for the operator

*operator* Any of the following:

- CONTAINS, `con`,
- CONTAINS\_MATCH, `cm`,
- ENDS\_WITH, `ew`,
- EQUALS, `==`, `eq`, `is`,
- GREATER, `>`, `gt`,
- GREATER\_EQUAL, `>=`, `ge`,
- IN, `in`,
- LESS, `<`, `lt`,
- LESS\_EQUAL, `<=`, `le`,
- MATCHES, `m`,
- NOT\_EQUALS, `<>`, `neq`,
- NOT\_IN, `nin`,
- STARTS\_WITH, `sw`

*right* Right argument for the operator

## shell:cp

Copies a variable or subtree.

### Syntax

```
shell:update [options] from to
```

### Options

(See `help --all` for details on all other options.)

### Arguments

*from* Name of item to move

*to* New name or location





## shell:display

Sets the default display/formatting.

### Syntax

```
shell:display [options] displayType (flavor)
```

### Options

(See `help --all` for details on all other options.)

### Arguments

*displayType*            Formatting or display type (see [Common Options](#) for details)

*flavor*                Flavor of the display type, if available

## shell:each

Executes a closure on a list of arguments. See also [Filtering Output](#) in *Other Functionality* earlier in this guide.

### Syntax

```
shell:each [options] values function
```

### Options

`--arg <list>`            Additional arguments to pass to the function (numbered \$2 and up).

`--flatten` or `-f`        Flatten nested lists of results into a single output list.

`--threads` or `-t <integer>`  
                          Number of threads (parallel threads implied)

`--parallel` or `-p <string>`  
                          Use one thread for each item (unless `--threads` is provided as well).

`--timeout` or `-t <integer>`  
                          Timeout for parallel activity, in seconds.

`--nulls` or `-n`         Include nulls in results (by default nulls are discarded).

`--rule` or `-r <integer>`  
                          Send results from each completed iteration to the rule system.

`--sort <property>`      Sort the output, using the specified Property name to sort on.



`--order-with <function>`

Sort the output, extracting the key with this function.

Example: `{ $1 method }`

`--unique`

Remove duplicates from the results.

`--where` or `-w <function>`

Filter by a function.

`--where-not` or `-wn <function>`

Keep where this function is false.

(See `help --all` for details on all other options.)

### Arguments

*values* Collection of arguments to iterate on

*function* Function to execute

### Examples

```
> each (seq 5) { echo $1 }
```

This loops over the numbers 1 through 5, printing each of them.

```
> each (drives) --where { $1 starts_with fi } { (drive:get $1) uuid }
```

This lists the IDs of drives whose ID starts with `fi`, and prints their UUIDs.

```
> each (volumes -o) --where { ($1 id) sw vol }
```

This lists volumes whose IDs start with `vol`.

### shell:echo

Echoes or prints arguments, or sends them to the log.

### Syntax

```
shell:echo [options] argument(s)
```

### Options

`--stderr` Sends the output to the error stream.

`--n` Do not print the trailing newline character.

`--log` Send to the log.



`--logLevel <level>` Log level to use (implies `--log`). Valid values include:  
`trace, debug, info, warning, error`

`--logCategory <string>`  
Log category to use (implies `--log`).

(See `help --all` for details on all other options.)

### Arguments

*argument* One or more arguments to display, separated by spaces

### shell:eval

Evaluates a binary operation, or executes a command provided as a string. This command can also be used to do mathematics, or to construct a string as a command and then evaluate it.

### Syntax

`shell:eval [options] left operator right`

### Options

(See `help --all` for details on all other options.)

### Arguments

*left* Left argument for the operator, or string to execute as a command

*operator* Any of the following:  
`AND, DIVIDE, /, MINUS, -, MOD, %, OR, PLUS, +, TIMES, *, XOR`

*right* Right argument for the operator

### shell:exit

Exits the CLI, optionally returning an exit code.

### Syntax

`shell:exit [options] exitCode`

### Options

(See `help --all` for details on all other options.)

### Arguments

*exitCode* Integer code to return to the OS when exiting



## shell:explain

Analyzes the last error and attempts to provide additional information.

### Syntax

```
shell:explain [options]
```

### Options

- `--max` or `-m <level>` Maximum number of rules to fire (defaults to all)
- `--timeout` or `-s <num>` Timeout for rule execution, in seconds (defaults to 30)
- `--context` or `-c <string>`  
Name of rule context (defaults to "explain")

(See `help --all` for details on all other options.)

## shell:filter

Keeps or discards objects that match a pattern, optionally extracting a field.

### Syntax

```
shell:filter [options] input(s)
```

### Options

- `--not` Discard objects that match, keeping the ones that don't match.
- `--no-flatten` Keep the nested list structure, if any, of the input objects
- `--extract <string>` Collect the return values from the matched pattern expressions, which allows returning the properties of objects.
- `--xpath <string>` Provide an XPath-based pattern.

### Arguments

*input* Input object(s) to match against. If not provided, standard input will be parsed.

(See `help --all` for details on all other options.)

## shell:find

Recursively search the CLI environment tree, returning items that match conditions.

### Syntax

```
shell:find [options] locationOrFunction
```



## Options

`--name` or `-name` *<string>*

Name to search for (such as "file.txt" or "\*.txt"). This option is repeatable.

`--regex` or `-regex` *<string>*

Regex pattern to match against names. This option can be specified multiple times.

`--maxdepth` or `-maxdepth` *<num>*

Maximum depth to search

`--mindepth` or `-mindepth` *<num>*

Minimum depth to search (matches must be at least at this depth)

`--not` or `-not`

Find items that do *not* match.

`--dir` or `-dir`

Match only directories (environments).

`--exec` or `-exec` *<function>*

Execute a function, where \$1 is the path, \$2 is the directory, and \$3 is the name.

`--execdir` or `-execdir`

Execute a function in the directory of the item.

(See `help --all` for details on all other options.)

## Arguments

*locationOrFunction* Location to search or function to run (\$1 is the path, \$2 is the directory, and \$3 is the name). You can supply multiple locations and functions, in any order.

## shell:fold

Calls a closure with an input value (\$1) and a list element (\$2), passing the result of each call as the input to the next (unless suppressed with `--curry`).

## Syntax

```
shell:fold [options] values input function
```

## Options

`--curry` or `-c`

Curry instead of fold, passing input for each invocation and returning a list of results.



`--arg <list>` Additional arguments to pass to the function (numbered \$3 and higher)  
(See `help --all` for details on all other options.)

### Arguments

*values* Collection of arguments to iterate on  
*input* First input value  
*function* Closure, where \$1 is the value and \$2 is the input

### shell:grep

Prints lines matching the given pattern.

### Syntax

```
shell:grep [options] pattern
```

### Options

`--line-number` or `-n` Prefix each line of output with the line number within its input file.

`--invert-match` or `-v` Invert the sense of matching, to select non-matching lines.

`--word-regexp` or `-w` Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore.

`--line-regexp` or `-x` Select only those matches that exactly match the whole line.

`--ignore-case` or `-i` Ignore case distinctions in both the PATTERN and the input files.

`--count` or `-c` Print only a count of matching lines per FILE.

`--color <colorOption>` Use markers to distinguish the matching string. WHEN may be 'always', 'never' or 'auto'. The default is 'auto'.

`--before-context` or `-B`  
Print NUM lines of leading context before matching lines. This places a line containing '--' between contiguous groups of matches. The default is -1.

`--after-context` or `-A`  
Print NUM lines of trailing context after matching lines. This places a line containing '--' between contiguous groups of matches. The default is -1.



`-context` or `-C` Print NUM lines of output context. This places a line containing '--' between contiguous groups of matches.

`--only-matching` or `-o` Print only the part of the line that matches the expression.

`--group` or `-g <string>` Print the contents of the regex group. Group 0 is the entire pattern.

`--group-separator` or `-sep <string>`

When printing multiple groups, use this item to separate them.

(See `help --all` for details on all other options.)

### Arguments

*pattern* Regular expression, following the syntax given at <http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html>

### shell:head

Displays the first lines of a file.

### Syntax

`shell:head [options] paths` or `URLs`

### Options

`-n <integer>` Number of lines to display, starting at 1

### Arguments

*paths* or *URLs* List of file paths or URLs to display, separated by spaces

### shell:if

Conditionally executes commands. See *Details* below for more information.

### Syntax

`shell:if [options] condition ifTrue` or `ifFalse`

### Options

`--not` Negate the logic of the condition.

### Arguments

*condition* Condition; a function or value

*ifTrue* Function to execute if the condition is true

*ifFalse* Function to execute if the condition is false



## Details

This CLI command checks a condition: if true, it executes the `ifTrue` function; if false, it executes the optional `ifFalse` function. The condition can be a value or a function itself. If it's a function, it will be called to get the condition value. Boolean values will be used directly. Numeric values are true if not equal to 0. String values are true if non-empty; null is false.

Use `shell:if` in conjunction with the `shell:test` command:

```
> if (test exists volume voll) { echo "Voll exists" } { echo "Voll
doesn't exist"
}
```

You can call a function as the condition. The example below is equivalent to the previous one:

```
> if {test exists volume voll} { echo "Voll exists" } { echo "Voll
doesn't exist"
}
```

You can call a defined function:

```
> my_test={test exists volume $1}
> if (my_test voll) { echo "Found" }
```

You can use functions to organize your code:

```
> yes={echo "Yes"}
> no={echo "No"}
> if (test exists volume voll) { yes } { no }
```

## shell:join

Joins the arguments together into a single string, and optionally into a string (with delimiters between them).

### Syntax

```
shell:join [options] arguments
```

### Options

- `--flatten` or `-f` Un-nest the list arguments.
- `--string` or `-s` Join the arguments together into a string.





`--delimiter` or `-d <string>`

Place a delimiter between the arguments. This implies the `--string` option.

(See `help --all` for details on all other options.)

## Arguments

*arguments*            Items to join into a string

## shell:load

Loads the CLI environment tree from a file.

## Syntax

```
shell:load [options] treeFile
```

## Options

`--input-file` or `-if <filename>`

Use file input.

`--input-url` or `-ir <URL>`

Use URL input. For example, `http://somehost/filename` or `ftp://[username[:password]@]host/path/file`

`--input-usb` or `-iu <file>`

Use content retrieved from the USB drive.

`--input-share` or `-ic <string>`

Use CIFS/Windows input. For example, `domain/user[:password]@host/share/filename`

`--input-scp` or `-is <string>`

Use SCP input. For example, `user[:password]@host:filename`

`--input-ssh` or `-ih <string>`

Use Unix shell file input: `user[:password]@host:filename`

`--input-pipe` or `-ip` Use `stdin` as input to the command line (non-interactive only).

(See `help --all` for details on all other options.)



## Arguments

*treeFile* Tree file to load. This defaults to the standard CLI tree location in your profile.

## [shell:ls](#)

Lists the contents of the current directory, or a provided path.

## Syntax

```
shell:ls [options] path
```

## Options

--long or -l Use the long form.  
--all or -a List hidden entries as well.  
(See `help --all` for details on all other options.)

## Arguments

*path* Tree path to list

## [shell:man](#)

Shows detailed information for one or more CLI commands at the console.

## Syntax

```
shell:man [options] command
```

## Options

--all or -a Create a full manual for all commands.  
--html Format as HTML.  
--lyx or -l Format as Lyx.  
--keys or -k Display a table of keyboard shortcuts.  
(See `help --all` for details on all other options.)

## Arguments

*command* Command to show details for

## [shell:markdown](#)

Transforms text with the markdown processor.



### Syntax

```
shell:markdown [options]
```

### Options

(See `help --all` for details on all other options.)

### shell:mkdir

Creates a new environment path in the CLI tree.

### Syntax

```
shell:mkdir [options] path
```

### Options

`--if-not-exists` or `-i` Create the path if it doesn't already exist.

`--parents` or `-p` Create parent directories as needed.

(See `help --all` for details on all other options.)

### Arguments

*path* Path to create

### shell:more

View the contents of a text file one screen at a time.

### Syntax

```
shell:more [options]
```

### Options

`--lines <number>` Display the specified number of lines, per screen.

### shell:mv

Renames or moves a variable or sub-tree.

### Syntax

```
shell:mv [options] from to
```

### Options

(See `help --all` for details on all other options.)



## Arguments

*from*                      Name of item to move  
*to*                         New name or location

## shell:printf

Returns a formatted string, based on arguments.

## Syntax

```
shell:printf [options] format arguments
```

## Options

`--echo` or `-e`             Echo the formatted string to the output stream, appending a newline if it doesn't end with one.

(See `help --all` for details on all other options.)

## Arguments

*format*                      Format pattern to use (quotes recommended)  
*arguments*                  Arguments for the given format pattern

## Example

```
shell:printf "%017d\n" 77
```

For detailed instructions about the allowable format strings, see <http://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html#syntax>

## shell:pwd

Shows the current working directory.

## Syntax

```
shell:pwd [options] id
```

## Options

(See `help --all` for details on all other options.)

## shell:quit

Quits the CLI.

## Syntax

```
shell:quit [options]
```



### Options

(See `help --all` for details on all other options.)

#### shell:rm

Removes a variable from the CLI environment.

### Syntax

```
shell:rm [options] path
```

### Options

(See `help --all` for details on all other options.)

### Arguments

*path* Path to the variable to remove

#### shell:rmdir

Removes a CLI environment path.

### Syntax

```
shell:rmdir [options] path
```

### Options

(See `help --all` for details on all other options.)

### Arguments

*path* Desired environment path (root, if not provided)

#### shell:save

Saves the environment tree. This includes the aliases, options, passwords, etc. that you have created, so they can be used in later sessions.

### Syntax

```
shell:save [options]
```

### Options

`--console` or `-c` Display the environment's XML to the console.

(See `help --all` for details on all other options.)



## shell:seq

Generates a sequence of numbers, or pattern-formatted strings. For a detailed discussion of format strings usable with the `--format` option, see <http://docs.oracle.com/javase/7/docs/api/java/util/Formatter>

### Syntax

```
shell:seq [options] last
```

### Options

`--first` or `-f` *<integer>*

First number to generate (default is 1)

`--first-letter` or `-fl` *<char>*

First letter to generate (implies `--letter`)

`--increment` or `-i` *<integer>*

Amount to add to the sequence (default is 1)

`--hex` or `-x`

Format numbers as hex (return strings).

`--octal`

Format numbers as octal.

`--binary` or `-b`

Format numbers as binary.

`--letter` or `-l`

Change generated numbers to letters, where 1 is 'a', 2 is 'b', etc.

`--uppercase` or `-u`

Generated uppercase letters (implies `--letter`).

`--format` *<string>*

Formatting string following Java's `String.format` rules

`--down` or `-d`

Count down, instead of up.

(See `help --all` for details on all other options.)

### Arguments

*last*

Last number or string to generate; or number of letters when the `--letter` option is used

## shell:set

Sets a flag in the current environment.

### Syntax

```
shell:set [options] setting value
```



## Options

(See `help --all` for details on these options: `--display`, `--output-file`)

## Arguments

*setting*

Any of the following values:

`ACTOR_SYSTEM_NAME`: Name of the actor system, when used

`ANSI`: Show color text.

`AUTOCOMPLETE_LIMIT`: Time, in seconds, to wait for the auto-completer to retrieve information

`CACHE_SAFT`: Cache CLI instances.

`COMPATIBILITY`: Handle backwards compatibility.

`CONFIRMATION`: For some commands, prompt the user before execution takes place.

`CONNECTION_TIMEOUT_SECONDS`: Time, in seconds, to wait for a connection to a CLI host

`DISABLED`: Disable SAFT connection.

`DISPLAY_FLAVOR`: Default flavor of display to use

`DISPLAY_TYPE`: Default display type

`EXTENDED_COMPLETION`: Display source information when completing certain types.

`LOG_MODE`: Enable automatic logging.

`MEMOIZE_SAFT`: Put the CLI results into memo format.

`MSRV_URL`: URL to the MSRV

`PARALLEL_EXECUTION`: Use parallel execution as a default.

`PASSWORD`: Password to use

`PREFERRED_PEER_PORT`: Port expected to be used for the actor system

`PROFILEDIR`: Default directory to save and load environments

`PROMPT_MILLIS`: Milliseconds to wait for prompt status construction.

`PROMPT_SHOW_BUSY`: Show the busy indicator in the prompt.

`PROMPT_SHOW_NODE_NAME`: Shows the node name in the prompt.



`PROMPT_SHOW_USER`: Show the current user name in the prompt.

`READ_TIMEOUT_SECONDS`: Time, in seconds, to wait for a response from a CLI host

`REST_LOG_PROMPT`: When logging REST, log transactions related to the prompt.

`REST_LOG`: REST call logging

`REST_LOG_URLS_ONLY`: When logging REST, record only the URLs, not the responses.

`ROOT_SAFT_URL_OVERRIDE`: Provide a CLI URL to be used instead of the one contained in the root of the environment tree.

`RULE_CONTEXT`: Name of the default rule context

`SAFT_EXCERPT_LIMIT`: Maximum size of SAFT log excerpts, in KiB.

`SAFT_LOG_EXCERPTS`: Include excerpts from `fio-saft` log in the CLI log.

`SAFT_REDIRECTOR`: Use the CLI's general redirection for distributed operations.

`SAFT_THREADS`: Suggested number of threads to use to communicate with the CLI

`SAFT_URL`: URL of the CLI

`STACKTRACE`: Show full stack traces when exceptions are encountered.

`STRICT`: Emit errors if SAFT responses do not conform to the known schema.

`SUPPRESS_EXECUTION`: Parse and validate commands, but suppress execution.

`TERMINAL_HEIGHT`: Height of terminal

`TERMINAL_WIDTH`: Width of terminal

`TIME_COMMAND`: Show execution times for commands.

`TIME_SAFT`: Show execution times for CLI calls.

`TRACE`: Print additional information regarding command execution.

`TREEFILE`: Location of the CLI's environment file

`UNICODE_TABLE`: Use Unicode table drawing characters.





USERNAME: A user name

VALIDATE: If false, prevent checking of command parameters prior to execution.

WATCH\_AUTO: Build out necessary watches automatically during execution.

*value* Value to set

### shell:sleep

Causes the CLI to sleep for a short time and then wake up.

#### Syntax

```
shell:sleep [options] duration
```

#### Options

`--second` or `-s` Use a duration time of seconds instead of milliseconds.

#### Arguments

*duration* Amount of time to sleep. The default time unit is milliseconds; use the `-s` option to specify seconds instead.

### shell:sort

Writes a sorted concatenation of all specified files to standard output.

#### Syntax

```
shell:sort [options] files
```

#### Options

`--ignore-case` or `-f` Fold lowercase to uppercase characters.

`--reverse` or `-r` Reverse the result of comparisons.

`--unique` or `-u` Output only the first of an equal run.

`--field-separator` or `-t <string>`

Use SEP instead of non-blank to blank a transition.

`--ignore-leading-blanks` or `-b`

Ignore leading blanks.

`--key` or `-k <list>` Fields to use for sorting, separated by spaces



`--numeric-sort` or `-n` Compare according to string numerical value

### Arguments

*files* List of files separated by spaces

### shell:source

Runs a script.

### Syntax

```
shell:source script arg(s)
```

### Options

`--input-file` or `-if` *<filename>*

Use file input.

`--input-url` or `-ir` *<URL>*

Use URL input. For example, `http://somehost/filename` or `ftp://[username[:password]@]host/path/file`

`--input-usb` or `-iu` *<file>*

Use content retrieved from the USB drive.

`--input-share` or `-ic` *<string>*

Use CIFS/Windows input. For example, `domain/user[:password]@host/share/filename`

`--input-scp` or `-is` *<string>*

Use SCP input. For example, `user[:password]@host:filename`

`--input-ssh` or `-ih` *<string>*

Use Unix shell file input, such as `user[:password]@host:filename`

`--input-pipe` or `-ip` Use `stdin` as input to the command line (non-interactive only).

(See `help --all` for details on all other options.)

### Arguments

*arg* Argument to use for the script. This can be specified multiple times.



## Examples

```
shell:source --input-file hello.fik
```

Load the `hello.fik` file, executing the script it contains.

```
shell:source --input-scp user:pass@host:setup.fik
```

Run `setup.fik` from an `scp` source, then execute it.

```
shell:source --input-share domain/user@host/share_name/setup.fik
```

Run `setup.fik` from CIFS/Windows share named `share_name`.

```
shell:source --input-url http://somehost/setup.fik
```

Run `setup.fik` from the given URL.

## shell:tac

Concatenates input to a string and returns the result. This command can also send output to a file.

## Syntax

```
shell:tac [options]
```

## Options

- `--no-return` or `-n` Don't return the input string.
- `--file` or `-f <file>` Store input to a file.
- `--binary` Store the stream contents directly to a file; do not perform any text translation. This option must be used in conjunction with `--file`.

## shell:tail

Displays the last lines of a file.

## Syntax

```
shell:tail [options] path OR URL
```

## Options

- `--n <integer>` The number of lines to display, starting at 1.
- `--f` Follow file changes
- `--s <long integer>` Sleep interval (used for the `--follow` option)

(See `help --all` for details on all other options.)



## Arguments

*path* or *URL*                      File path or URL to display

## shell:tee

Sends `stdin` to `stdout` and other specified locations.

## Syntax

```
shell:tee [options]
```

## Options

`--file` or `-f`                      Send content to a file.

`--binary` or `-b`                    Use no text decoding/encoding; just do a binary copy.

`--encoding <string>`                Encoding to use for output

`--input-encoding <string>`  
    Encoding to use for the input

`--log`                                 Send lines to the log

`--log-level <level>`                Log level to use (implies the `--log` option):  
    `trace, debug, info, warning, error`

`--log-category <string>`  
    Log category to use (implies the `--log` option)

(See `help --all` for details on all other options.)

## shell:test

Evaluate a specified condition, returning true or false.

## Syntax

```
shell:test [options] test type term(s)
```

## Options

`--not`                                Negate the result.

`--any`                                If multiple terms are used, the condition is true if any terms pass.

`--all`                                If multiple terms are used, the condition is true if all terms pass.

(See `help --all` for details on all other options.)



## Arguments

<i>test</i>	Type of test: exists, used, in_cluster, or connection
<i>type</i>	Object type:  bus, chassis, cluster, cna, cpu, drive, fan, inigroup, initiator, lun, node, pool, port, psu, raid, target, temp, volume
<i>terms</i>	Terms to use

## shell:throw

Throws a Java exception. This is useful for simulating error conditions.

### Syntax

```
shell:throw [options] className message
```

### Options

(See `help --all` for details on all other options.)

## Arguments

<i>className</i>	Qualified name of the exception class to throw
<i>message</i>	Optional message to pass to the exception constructor

## shell:types

Returns a list of the type names for the CLI.

### Syntax

```
shell:types [options]
```

### Options

(See `help --all` for details on all other options.)

## shell:unset

Removes the specified setting(s) from the environment.

### Syntax

```
shell:unset [options] setting
```

### Options

`--setting` For a list of settings, see the `shell:set` command.



## Appendix B: Common CLI Tasks

This appendix describes some common tasks that may be useful in working with the ION Accelerator CLI. Other common tasks are outlined in the [About the Command-Line Interface \(CLI\)](#) section. For complete details on command syntax, as well as usage examples for most commands, see the [Command-Line Reference](#) section.

### COPYING TO/FROM ION ACCELERATOR



The `config:backup` command is used for the following examples, but any other command that supports output routing could also be used.

#### Routing Output

Task	Example	Description
Back up to scp (Unix Secure Copy) destination with a generated filename.	<code>backup --output-scp user@host</code>	You will be prompted for a password to connect to the remote host; a filename will be generated based on the name of the ION system you are backing up, together with a timestamp.
Back up to scp, using a specific filename.	<code>backup --output-scp user@host:filename.xml</code>	You will be prompted for the password; the configuration will be stored with your specified filename.
Back up to scp, specifying a password, generated filename.	<code>backup --output-scp user:password@host</code>	A generated filename will be used.



Back up to a Windows/CIFS share with a generated filename.	<code>backup --output-share domain/user@host/shareName</code>	Saves to a Windows share; you will almost always need to provide the domain. You will be prompted for a password.
Back up to a Windows/CIFS share, using a specific filename.	<code>backup --output-share domain/user@host/shareName/filename.xml</code>	You will be prompted for a password.
Back up to a Windows/CIFS share, providing a password.	<code>backup --output-share domain/user:password@host/shareName</code>	A generated filename will be used.
Back up to a specific file.	<code>backup --output-file my_config.xml</code>	Backs up to my_config.xml in the user's home directory. When logged in as admin, this will be /home/admin.
Back up to the USB drive.	<code>backup --output-usb</code>	If a USB drive is plugged in to the ION appliance, this will write the configuration to the USB drive with a generated filename.
Store a tabular list of luns into a text file in the home directory.	<code>luns --output-file luns.txt -dt</code>	
Store the list of LUNs in JSON format to a file in the home directory.	<code>luns --output-file luns.json -dj</code>	
Store a list of LUNs in XML format to a file in the home directory.	<code>luns --output-file luns.xml -dx</code>	
Store the list of luns in tabular text format to an scp destination, using a generated filename.	<code>luns --output-scp user@host -dt</code>	You will be prompted for a password; see the notes on generated filenames below.
Store the list of luns in JSON format to a Windows share destination, using a generated filename.	<code>luns --output-share domain/user@host/shareName -dj</code>	You will be prompted for a password; see the notes on generated filenames below.



## Routing Input

Some commands require files as input. Here are some examples:

Task	Example	Description
Restore from a config file in the user's home directory	<pre>restore my_config.xml</pre>	Reads and applies the configuration in the file. Tab completion is available for choosing the file.
Restore from a USB drive	<pre>restore --input-usb my_config.xml</pre>	Reads a configuration from the USB drive. Tab completion is available for the files on the USB drive; you are limited to choosing from files available there.
Restore from an scp source	<pre>restore --input-scp user@host:my_config.xml</pre>	Reads the configuration using Unix security copy; you will be prompted for a password.
Restore from a Windows share	<pre>restore --input-share domain/user@host/shareName/my_config.xml</pre>	Reads the configuration from a Windows share; you will be prompted for a password.
Restore from an http URL	<pre>restore --input-url http://host:port/my_config.xml</pre>	Uses the http protocol to read the configuration from the given host/port and filename.
Restore from an ftp URL	<pre>restore --input-url ftp://user:password@host/path/my_config. xml</pre>	Uses the ftp protocol to read a configuration file. You must specify the password in the URL.





## WORKING WITH THE CLI ENVIRONMENT (TREE)

The CLI can store settings, aliases, and other configuration into its preferences file. By default this file is stored in `~/ .fikon/tree.xml`.

You interact with the tree in a way that is similar to working with a file system. Fikon's tree is a nested set of environments (directories). Each environment has variables in it, and each environment can contain child environments.

Task	Example	Description
List the contents of the current environment.	<code>ls</code>	Shows the variables that are bound in the current environment.
Make a child environment.	<code>mkdir child</code>	Creates a new child environment nested inside the current one.
Change into a child environment.	<code>cd child</code>	"Enters" the child environment, setting the current working environment
Change to a parent environment.	<code>cd ..</code>	Changes the working environment to the parent
Change to the root environment.	<code>cd /</code>	Goes to the top of the tree
Remove a child environment.	<code>rmdir child</code>	
Create a variable.	<code>varname=value</code>	Sets a value into the environment
Remove a variable.	<code>rm varname</code>	Removes a variable from the environment
Save the entire tree.	<code>save</code>	Saves the entire Fikon tree into your preferences file, which is at <code>~/fikon/tree.xml</code> by default.
Save the tree, showing its contents on the console.	<code>save -c</code>	Saves the tree, and shows you what is being saved.
Save your CLI environment tree to an scp destination.	<code>save --output-scp user@host:my_env.xml</code>	Stores your Fikon environment (setup) to an scp destination, prompting you for a password.
Reload the tree file.	<code>load</code>	Reloads the tree file from the default location
Reload the tree from a file.	<code>load treeFile.xml</code>	Reloads the tree from the specified file.



Load your Fikon environment from an scp source	<code>load --input-scp user@host:my_env.xml</code>	Loads the Fikon environment from an scp source, prompting you for the password to use.
--	--	--

## Working with Tree Settings

Settings show up in your current tree location and are visible as uppercase entries. When you create a setting, it is inherited by all child environments below your current environment.

Task	Example	Description
Remove a setting.	<code>unset read_timeout_seconds</code>	Unset removes an entry from the environment (as does <code>rm</code> ). Unset tab-completes the available settings.
Remember settings.	<code>save</code>	Writes your environment tree to <code>./.fikon/tree.xml</code> , which is read each time Fikon starts (unless <code>--norc</code> is specified on Fikon's command line).
Change the fio-saft timeout.	<code>set read_timeout_seconds 60</code>	If fio-saft is taking a long time to respond (but <i>is</i> eventually responding) you can change the default timeout, which is 30 seconds.
Show interactions with fio-saft.	<code>set rest_log console</code>	This will display every interaction with fio-saft on the console.
Show only key fio-saft interactions.	<code>set rest_log_urls_only on</code>	Reduces the output of <code>set rest_log_console</code> to show only the URLs and response codes, without response bodies.
Set time commands.	<code>set time_command on</code>	For each command entered, shows how long it takes to execute it.
Change the prompt.	<code>set prompt_show_busy &lt;val&gt;</code> <code>set prompt_show_node_name &lt;val&gt;</code> <code>set prompt_show_user &lt;val&gt;</code>	These control the contents of the prompt that's displayed. Note that in non-interactive mode a simplified prompt is used and no display options are allowed.



## ATTACHING TO A REMOTE ION ACCELERATOR APPLIANCE

You can run the CLI on a workstation or laptop and then attach it to a remote ION Accelerator system. This is done with an SSH tunnel.

Task	Example	Description
1. Make an environment.	<code>mkdir remote</code> <code>cd remote</code>	Creates an environment in the Fikon tree, and changes into that environment. The environment is now ready to accept settings.
2. Set up a tunnel.	<code>url=ssh://&lt;ip address&gt;</code>	Tells the CLI that it should use the <code>ssh</code> protocol to connect to a remote ION system at the given IP address.
3. Set up authentication	<code>user=&lt;user&gt;</code> <code>password=&lt;password&gt;</code>	The user name is often "admin" in a standard ION setup. When done in this way, the settings are saved into the user's environment tree. You can also use a different form of the url property. <code>ssh://user:password@&lt;ip address&gt;</code> That format will not require an additional user/pass property.
4. Test the connection	<code>drives</code>	This checks to see if an SSH tunnel can be formed to the target node. If so, the <code>drives</code> commands is executed and the results are displayed.
5. Save the connection	<code>save</code>	This tells Fikon to save its environment tree into <code>~/.fikon/tree.xml</code> , so it will be available the next time you start up.



## Appendix C: About the ION Accelerator Guides

---

The *ION Accelerator CLI Reference* helps you use the ION Accelerator software in a command-line environment, including setting up a storage profile and pools, creating volumes, adding initiators, managing ioMemory, etc.

Other ION Accelerator guides include:

- *ION Configuration Guide* – an introduction to the ION Accelerator software, as well information on installation, setup, host multipathing, application tuning, and platform configuration
- *ION Accelerator GUI Guide* – explains how to use the ION Accelerator GUI to administer shared PCIe flash storage