ExaLINK Fusion User Guide

Please note this PDF is auto-generated from https://fusion.exablaze.com, so formatting may render incorrectly in some places.

Table of Contents

About the ExaLINK Fusion

Introduction

The ExaLINK Fusion is a uniquely flexible top-of-rack switch with a unique and modular architecture. As a low latency switch, it allows for 95ns switching, 39ns forwarding and aggregation and sub 5ns layer 1 patching and tapping, all within the same device.

Extensible by design, the Fusion features two internal module bays that are both fully interconnected via a high speed, reconfigurable layer 1 switch. This allows deployment of both Exablaze developed modules as well as customer developed logic directly to the FPGA itself.

The ExaLINK Fusion

Overview of features

A unique architecture that provides an unprecedented level of control over your network.

- Tap, patch and replicate data electronically and with very low and deterministic latency. Tapping using the Fusion is far more flexible than using traditional optical taps, and does not suffer from signal degradation associated with high-fanout tapping. Patching allows the Fusion to dynamically reconfigure the physical topology of your network with very low latency overhead. Both patching and tapping/replication is done at port to port latencies of 5ns.
- Mux and aggregate feeds with low and deterministic latency. Useful in situations where sharing of a common network resource is required (for example, an order line), muxing delivers the industry's lowest latency of 55ns port to port.
- Full Layer 2 switching where many hosts communicate with each other is fully supported and is performed at 95ns latency, the industry's lowest.
- Timestamping and logging of data flows through the Fusion to 2.8ns of accuracy is supported, with multiple time synchronization options available. Additionally, the Fusion can be delivered with a stable oven baked oscillator option for low drift hold-over in the event that PPS (pulse-per-second) input or other sources of reference time are lost.
- Development of custom applications can be achieved by pushing FPGA applications directly to the Fusion.

Accepts all SFP+ modules

The Fusion works with all brands of SFP+ modules and is not tied to a particular brand.

Other features

- Dedicated Ethernet management port
- Robust and intuitive command line interface (CLI)
- http/https GUI (coming soon)
- SNMP-compatible management, for integration with network management and monitoring infrastructure
- Switch configuration, including per-port filtering
- Software updates via USB, TFTP, console or web interface
- Access to management interfaces via serial console and Ethernet interfaces
- Monitoring of available SFP+ parameters, including SFP+ model information and dynamic parameters such as received power, transmitted power and temperature
- Log access via CLI and remote logging via syslog
- TACACS+ and AAA

Safety and installation warnings

Environment	**Ambient temperature** Make sure the ambient temperature does not exceed the maximum ambient temperature allowed for the ExaLINK Fusion (104F, 40C). If installed in a closed or multi-unit rack assembly, the ambient temperature of the rack during operation will be greater than room ambient. **Air flow** Install the ExaLINK Fusion in the rack in a way that provides sufficient air flow for safe operation. **Mechanical loading** Mount the ExaLINK Fusion in the rack with a mechanical load that is evenly distributed and not excessive. **Circuit overloading** Ensure that no overloading of the circuits occurs which might affect overcurrent protection and supply wiring. The ratings are provided on the unit. **Earthing** Ensure that the rack-mounted equipment is earthed reliably. Consider using supply connections other than direct connections to the branch circuit (e.g. use of power strips).
Power	Check that your ExaLINK Fusion is rated to be used with the mains power in your country. Total ExaLINK Fusion ratings: • 90-264V AC @ 3A max, 50/60 Hz; or • 40-72V DC @ 6.7A max
Before servicing	Disconnect the two power supply cables before servicing.
Power cables	Ensure the ExaLINK Fusion uses mains power cables approved in the country of operation
Clock battery 	**CAUTION:** The ExaLINK Fusion has a battery-powered real-time clock circuit. There is a danger of explosion if the battery is replaced incorrectly. Replace only with CR2032 type coin cells. Discard used batteries according to the manufacturer's instructions.
Laser safety 	**CAUTION:** SFP+ module used in the ExaLINK Fusion can be a CLASS 1 LASER PRODUCT. Invisible laser radiation may be emitted from the aperture of an SFP+ module when the fiber cables are disconnected. Do not stare into the open aperture of an SFP+ module and avoid exposure to laser radiation when a fiber cable is disconnected from an SFP+ module.
Warranty void if opened 	**Do not open the case of the ExaLINK Fusion** The warranty of the ExaLINK Fusion will be void if the case is opened. An internal tamper switch logs the time and date of events when the lid is removed.
FCC Compliance 	This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense

Getting Started

This section provides information on how to install the ExaLINK Fusion.

Package contents

The ExaLINK Fusion box should contain the following items:

- ExaLINK Fusion box
- ExaLINK Fusion Chassis
- Rackmount kit
  - 8x M6 rack-mounting nuts
  - 8x M6 rack-mounting bolts
  - 8x M6 rack-mounting washers
- 2x IEC power leads
- 1x Serial port adapter cable
- 2x Mounting extension rails (units shipped after 2018)
- 2x Mounting extension brackets (units shipped before 2019)

Already installed should also be:

- 2x Power supply
- 2x Fan modules

Note: that the power supplies have an arrow on the exhaust fan that points in the direction of airflow. Fan modules are colored red for front-to-back (FTB, i.e. port side intake) airflow and blue for back-to-front (BTF, i.e. port side exhaust) airflow.

Mounting the ExaLINK Fusion

Since the power supplies and fans add significant weight to the ExaLINK Fusion, it is recommended that the system is rack mounted *prior* to installing them. **Note:** that two people are required to complete the installation.

1. Clip the supplied-mounting nuts into the rack as shown, noting the gap of one notch between the nuts.
2. One person should hold the ExaLINK Fusion, aligning the front panel of the ExaLINK Fusion with the nuts.
3. The second person can then affix the washers and bolts to each of the four mounting holds on the front panel of the ExaLINK Fusion, securely fastening them to the nuts.

Mounting the ExaLINK Fusion into a rack

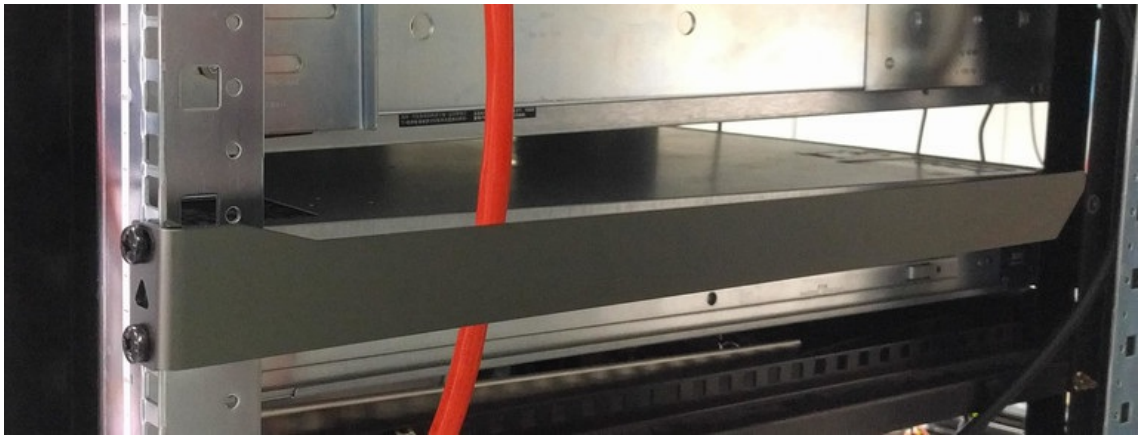
Mounting the ExaLINK Fusion with rear support rails (units shipped after 2018)

The ExaLINK Fusion ships with rear support rails. These can be added for extra structural support for the rack system. The process for installing these is as follows:

1. Fasten the rear rails at the desired height. Ensuring that the indicator arrow on the flange is pointing upwards.

Attaching the right hand side rail.

Attaching the left hand side rail.



Example of attached left hand side rail.

2. Lift the Fusion to the desired height and align the installed rails with the rail mating wholes on the rear of the Fusion.

Inserting the right hand side rail.

Inserting the left hand side rail.

3. Slowly rack the Fusion by sliding back onto the rails, until the front is flush with the rack.

Sliding back on both rails.

4. Fasten the ExaLINK Fusion onto the rack as per the method for installing the front mount bolts. Due to the ExaLINK Fusion's weight it may be easier with the aid of another person.

Fasten the Fusion in the normal manner.

Mounting the ExaLINK Fusion with rear supports (units shipped before 2019)

The ExaLINK Fusion ships with rear support mounting brackets. These can be added for extra structural support for the rack system. The process for installing these is as follows:

1. Remove both of the power supplies from the ExaLINK Fusion (see above).
2. Remove both screws from each of the rear mount bars to separate the mounts into their constituent pieces.
3. Locate, unscrew and remove the brackets in the rear of the ExaLINK Fusion (pictured below).

Removing the rear brackets of the ExaLINK Fusion (screw locations also pictured).

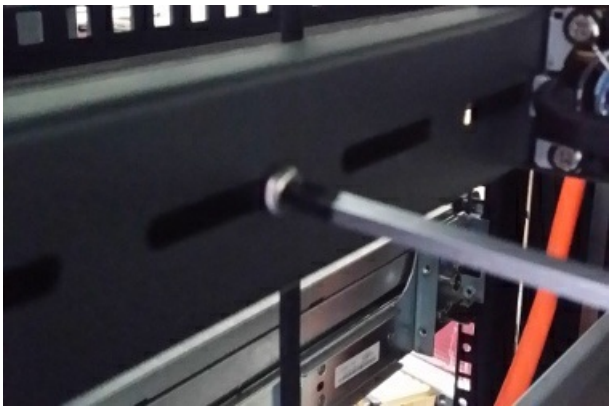
4. Replace the power supplies.
5. Fasten the rear mounts to where the removed brackets were previously. Ensure that the flaps for the bolts are facing outwards (i.e. in line with the front mount positions).

Attaching the rear mounts of the ExaLINK Fusion, with inserted power supplies.

6. Mount the ExaLINK Fusion onto the rack as per the method for installing the front mount bolts. Due to the ExaLINK Fusion's weight it may be easier with the aid of another person.
7. Remove the two screws restricting the rear mounting extension and adjust it so it fits the required depth for the mounting solution (Pictured below). When in place, apply the rack mount brackets and bolts.

Rear mounts being screwed into place.

8. Replace the two screws for the rear mounting extension to lock the ExaLINK Fusion in place.



Variable length sections being secured in.

Note: It is also acceptable to reinsert the power supplies after the ExaLINK Fusion has been successfully mounted into the rack to reduce the weight during rack mounting.

Installing power supplies and fans

The power supplies and fans can now be installed into the ExaLINK Fusion chassis.

To install a fan module:

1. **Note:** the position of the power connector on the fan module, relative to the power connector on the ExaLINK Fusion. Rotate the fan module so that it aligns with the ExaLINK Fusion connector.
2. Push the fan module into the ExaLINK Fusion until it is flush with the rear panel.
3. Tighten the two screws to lock the fan module into position.

A fan module can be removed in the future by loosening the screws and pulling out the module.

Installing a fan module

To install a power supply module:

1. The power supply orientation is the same for both left and right supplies. Note the position of the IEC power input, which should be on the right when viewed from the rear.
2. Push the power supply module in until a click is heard. The quick release lever should engage at this point.

Installing a power supply. Note the position of the IEC connector and the use of the quick release lever (left image).

The ExaLINK Fusion power supplies are redundant and failover from one to the other is automatic. It is recommended to operate the device with both power supplies installed, however, it will operate with only a single supply.

To remove a power supply module in the future simply press the quick release lever whilst pulling on the removal handle on the rear of the supply

Installing Line cards

Line cards are installed into the front of the ExaLINK Fusion. Your ExaLINK Fusion will arrive with line cards pre-installed as per your order, however if you wish to install additional line cards in the future, the following steps are required.

1. Ensure the ExaLINK Fusion is powered down. Hot-plugging of line cards will not damage the device, but currently the software will not recognize additional line cards until a power cycle.
2. Take the line card and align the edges with the card-guides in the line card bay.
3. Push the line card in. The front panel door will swing upwards and out of the way. The line card will engage the connector at the rear of the bay, and a noticeable increase in force will result.
4. Push until the line card is flush with the front panel.

The bottom bracket on the line card doubles as an ejection lever. To eject the line card:

1. Apply force with a thumb to the 'ejection' symbol on the line card bracket. The lever should swing out at the opposite end.
2. Pull on the lever. A cam mechanism will extract the line card from its connector.
3. Once the connector force has been overcome, the line card should easily slide out of the line card bay.

Removing a line card. Note the use of the ejection mechanism.

Installing cables and SFPs

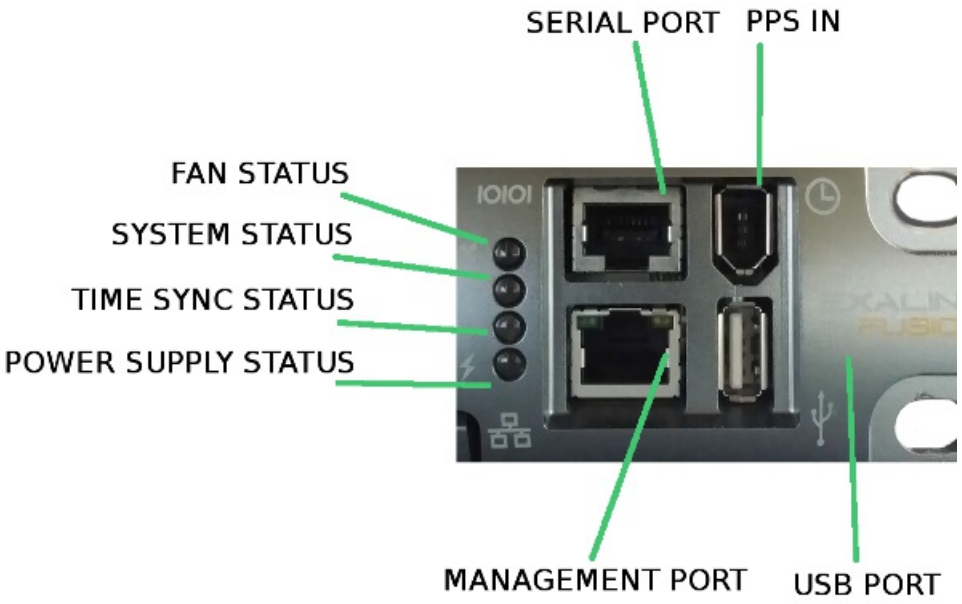
Once power supplies, fans and line cards have been installed, it is possible to install the power supply cables, management interface cables and any SFP or passive twinax cables to the ExaLINK Fusion.

See Configuration for information on setting up the management interface of the ExaLINK Fusion.

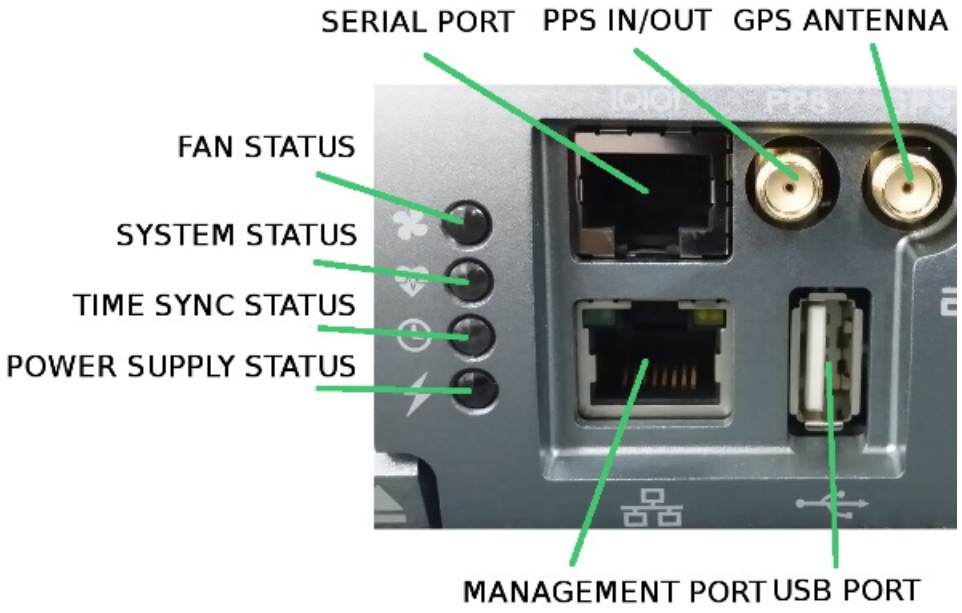
Ports and Indicators

Management ports and indicators

The right hand side of the front panel of the ExaLINK Fusion includes a number of indicators and connectors for the management of the device. Refer to the image below for details on the connectors and indicators.



*Management connectors and indicators for serial numbers EXAFSN-A-**



*Management connectors and indicators for serial numbers EXAFSN-B-**

- Fan Status will be GREEN if both fan modules are operating correctly, and RED if a fault is detected in one or more fan modules.
- System Status will be RED if the management software experiences an error, GREEN otherwise
- Power Supply Status will be GREEN if both power supply modules are operating correctly, and RED if a fault is detected in one or more power supply modules.

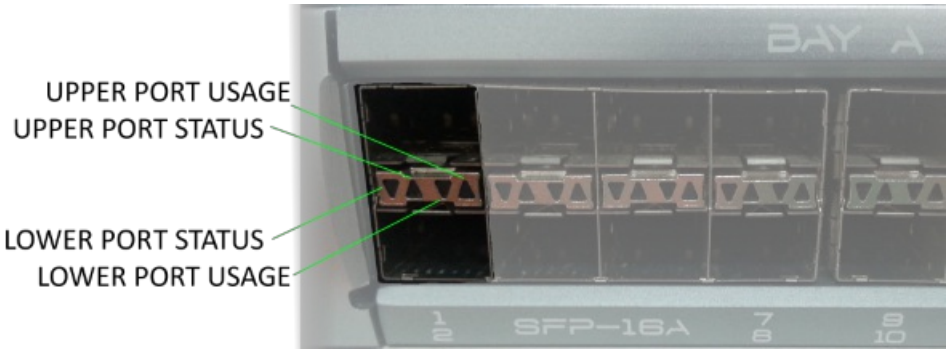
Note

Note that if all four management status indicators remain ORANGE on Fusion power up it is most likely the Fusion was prevented from booting normally.

Check the management serial connection and ensure data is not being inadvertently sent to the Fusion console port, which can interrupt the Fusion's boot process.

Line card port indicators

Each SFP+ port on a line card has two LEDs that convey port status and configuration.



Line Card Port Indicators

Port Usage Indicators:

- OFF: Port is idle (TX is OFF)
- ORANGE: SFP+ module detected and port is idle due to being unallocated. Check your Fusion configuration.
- RED: Port is idle due to internal port allocation error. Check your Fusion configuration.
- GREEN: Port is in use and part of a valid Fusion configuration.

Port Status Indicators:

- OFF: No SFP+ module detected.
- ORANGE: SFP+ module detected and idle.
- RED: SFP+ module detected, failure detecting input signal.
- GREEN: SFP+ module detected, input signal received OK.

System Time

The system time on the ExaLINK Fusion can be set manually or synchronized to a number of remote time sources. All time references within the ExaLINK Fusion are UTC. Maintaining accurate time is particularly important for packet capture and mirror ports with timestamping enabled. Multiple time synchronization options are available, including PTP (Precision Time Protocol) and NTP (Network Time Protocol).

The time can be set and synchronized via any of the following methods:

- Manually setting the system clock
- GPS sync which can achieve nanosecond level accuracy.
- PTP sync which can achieve microsecond level accuracy.
- NTP sync which can achieve millisecond level accuracy.
- A PPS input can be used to further improve the time sync.

Manual Time Set

In order to set the time manually, use the `time set` command. The time format provided should be YYYY-MM-DD HH:MM:SS, for example:

```
admin@EXALINK-FUSION> config time set 2015-12-19 15:37:00
System time updated
```

GPS Time Sync

Any Fusions with a serial number of EXAFSN-B-* include an onboard GPS receiver. A GPS antenna can be attached to the appropriate SMA connector on the front panel allowing the Fusion to synchronize the system clock to highly accurate UTC time. This is enabled using the `timesync gps` command, as follows:

admin@EXALINK-FUSION> configure timesync gps Using GPS receiver

GPS position and precise UTC time is determined at the point of the GPS antenna, not at the Fusion itself. Since the antenna would typically be located some distance from the Fusion via a cable, signal propagation delays through the cable cause the time calculated by the receiver to be slightly behind UTC.

Cable delay is a function of the cable type. RG-58 cable for example typically delays the signal 1.54 ns/ft. For 50 feet of cable (15 m) the delay would be 77 nanoseconds.

The `timesync gps` command can optionally take an `antenna-delay` parameter to take antenna cable delay into account, for example:

```
admin@EXALINK-FUSION> configure timesync gps antenna-delay 77
Using GPS receiver with 77 ns antenna delay
```

To check the status of the GPS timesync, use the `show timesync` command:

```
admin@EXALINK-FUSION> show timesync
Sync method    : gps

GPS fix        : yes
Location       : 41.796754°N 88.24276°W 19.2m
Time accuracy  : 9 ns
Satellites     : 12
Last sync time : 2016-04-01 05:41:49
```

PTP Time Sync

The Precision Time Protocol (PTP) is a protocol used to synchronize clocks throughout a network. On a local area network, it achieves clock accuracy in the sub-microsecond range.

To use PTP, use the following command, optionally specifying the PTP domain to use:

```
admin@EXALINK-FUSION> config timesync ptp 0
Using PTP on management interface
```

To check the status of the PTP timesync, use the `show timesync` command:

```
admin@EXALINK-FUSION> show timesync
Sync method    : ptp

PTP domain     : 0
PTP state      : SLAVE
Best master    : 192.168.220.13
Last offset    : 0.000000400 s

Clock eth0     : state LOCKED, adev 2308.405000
Clock syst     : state LOCKED, adev 61.970000
```

NTP Time Sync

To synchronize time to an NTP server, use the following command specifying one or more NTP server addresses:

```
admin@EXALINK-FUSION> config timesync ntp 192.168.220.13 121.0.0.41
Using NTP on management interface
```

Status can again be checked using the `show timesync` command:

```
admin@EXALINK-FUSION> show timesync
Sync method   : ntp

NTP stratum   : 2
NTP server    : 192.168.220.13
Poll interval : 32
Last offset   : 0.044545 s
```

Disabling Time Sync

To disable timesync, issue the `no timesync` command:

```
admin@EXALINK-FUSION> config no timesync
Time synchronization disabled
```

PPS Input and Output

A Pulse-Per-Second source can be connected to the ExaLINK Fusion to improve the time sync accuracy. Making use of a PPS input is supported for all timesync modes except GPS.

For example, to use a PPS input in conjunction with NTP, use the following command:

```
admin@EXALINK-FUSION> config timesync ntp+pps 192.168.220.13 121.0.0.41 rising
Using NTP on management interface and PPS
```

In all cases where a PPS input is used, a time offset can be specified to calibrate out the effect of PPS cable length/delay. For example, to synchronize time using PPS only, and to calibrate out 20ns worth of cable delay, use the following command:

```
admin@EXALINK-FUSION> config timesync pps rising cable-delay 20
Using PPS
```

Setting PPS cable delay requires version 1.10.0 or later

In all cases where a PPS input is used, it's possible to add an internal 50ohm termination resistor to reduce reflections and improve the quality of the PPS signal. In cases where a "daisy chain" of devices are sitting on the PPS network, only the last device in the chain should have PPS termination enabled.

To enable PPS termination, add the `termination` keyword to the `timesync` command. In the example above where NTP+PPS was used, the command would be:

```
admin@EXALINK-FUSION> config timesync ntp+pps 192.168.220.13 121.0.0.41 rising term
Using NTP on management interface and PPS
```

Enabling PPS termination requires version 1.13.0 or later

The Fusion will report whether a valid PPS is being received when the `show timesync` command is issued:

```
admin@EXALINK-FUSION> show timesync
Sync method   : ntp

NTP stratum   : 2
NTP server    : 192.168.220.13
Poll interval : 32
Last offset   : 0.043768 s

PPS edge      : rising
Termination   : enabled
Cable delay   : 0.000 ns

PPS signal    : yes
```

A PPS output can also be generated by the ExaLINK Fusion. The accuracy of this PPS edge will be a function of the accuracy of the time reference the Fusion is currently using. For example, the PPS output generated from a Fusion sync'd via NTP will be much less accurate than that of a Fusion sync'd via GPS.

The Fusion can generate a PPS output with either a rising or falling edge generated on the second boundary, for example:

```
admin@EXALINK-FUSION> configure timesync output pps rising
Pulse-per-second time synchronization output enabled
```

The voltage level of the Fusion PPS input and output is 3.3V, however the circuitry is 5V tolerant.

To disable PPS output, issue the `no timesync output pps` command:

```
admin@EXALINK-FUSION> config no timesync output pps
Pulse-per-second time synchronization output disabled
```

PPS Time Synchronization Fine-tuning

> The PPS time sync fine-tuning parameters are only available on the **ExaLINK Fusion HPT** and requires version 1.11.0 or later.

When a highly accurate PPS source is used with the Fusion HPT, better time synchronization may be achieved by fine-tuning the PPS time sync parameters.

The PPS time synchronization algorithm works by calculating the frequency and offset correction to be applied to the internal clock. The frequency and offset are measured over a number of PPS samples, both of which are configurable by adding the optional `window-size` parameter followed by one or two numbers, for example:

```
admin@EXALINK-FUSION> config timesync ptp+pps window-size 64 4096
```

If `window-size` is left out, the default values 64 and 4096 are used. If `window-size` is present but only one number is provided, the same number is used for both parameters.

The first parameter is the number of samples (seconds) used to calculate the clock offset. A larger number smooths out jitter in the PPS signal and the sampling thereof. It is recommended to set this value smaller than the second parameter.

The second parameter is the number of samples (seconds) used to calculate the frequency error. A larger number allows a more accurate measurement, but also takes longer to converge and responds slower to changes in frequency.

The smoothed clock offset and the frequency error can be observed using statistics logging. You can use this to guide the selection of the best parameters for your PPS source.

Displaying Time

The current system time can be displayed using the `show time` command:

```
admin@EXALINK-FUSION> show time
2015-12-19 17:12:22
```

The system uptime can be displayed using the `show uptime` command:

```
admin@EXALINK-FUSION> show uptime
3 days, 05:44:46.44
```

Timezones

> This feature requires version 1.10.0 or later

The system timezone can be set with the `config timezone` command. Tab completion can be used after typing timezone to list the available countries and timezones defined. For example:

```
admin@EXALINK-FUSION> config timezone America/Chicago
Timezone set to America/Chicago
```

Updating the firmware

The ExaLINK Fusion can be updated by one of following methods:

- SFTP
- USB
- TFTP
- HTTP

The updated firmware is made available in a tar archive. There are various ways to get the tar file onto the ExaLINK Fusion. The update process will verify the integrity of the contents of the tar file before applying the update and rebooting the ExaLINK Fusion. The update process takes a few minutes.

Firmware versions from 1.7.0 onwards have a support date requirement. Each ExaLINK Fusion will only accept firmware updates that are released before the end of support date for that device. Please see licensing for more information about the end of support date.

Note: You may continue to download and use firmware updates released before the end of support date, even after the support contract has expired. There are no time limits on firmware updates.

Please refer to the firmware downloads page for the firmware files and release notes.

Updating with SFTP

From Windows, a graphical tool like winscp can be used to transfer the release tar to the ExaLINK Fusion. SFTP is available as a command line tool as well.

```
$ sftp admin@192.168.220.10
admin@192.168.220.10's password:
sftp> put fusion_update.tar update/release.tar
```

Once the release is available on the ExaLINK Fusion, the firmware is updated using the command line interface.

```
admin@EXALINK-FUSION> configure update file release.tar
Starting update process...
```

Updating with a USB drive

The release tar can also be installed from a USB drive. The drive should be formatted with vfat, and the tar file should be copied onto the first partition. Plug the USB drive into the ExaLINK Fusion, and then use the command line interface.

```
admin@EXALINK-FUSION> configure update usb release.tar
Starting update process...
```

Updating with TFTP

If the release tar is made available though a TFTP server, then it can be copied and installed through the command line interface.

```
admin@EXALINK-FUSION> configure update tftp 192.168.220.160 /path/to/update_fusion.
Starting fetch and update process...
```

Updating with HTTP

The Fusion can download updates from a HTTP server:

```
admin@EXALINK-FUSION> configure update http https://fusion.exablaze.com/files/exali
Starting fetch and update process...
```

Licensing

The ExaLINK Fusion has several features that can be unlocked by adding licenses. Licenses also contain the expiry date of the Fusion's support contract.

A license is a text file provided by Exablaze that can be copied to the Fusion via sftp or entered in on the command line.

License files can contain information for a single or multiple Fusions, meaning that a single, common license file can be used for all Fusions owned by an organization.

Support Expiry Date

A support contract is included with all Fusion purchases that provides access to Exablaze engineering and support staff, hardware warranty and access to firmware upgrades as they are released.

The Fusion will not accept firmware updates that are released after the expiry date of that Fusion's support contract. The end of support date for a Fusion can be checked by issuing the `show version` command. The license file itself can also be opened in a text editor to check the end of support date.

Please contact the Exablaze sales team or your reseller if you wish to renew the support contract for your Fusions and get new license files.

Note

You may continue to download and use firmware updates released before the end of support date, even after the support contract has expired. There are no time limits on firmware updates.

Checking Licensed Features

In order to see which additional features your Fusion has been licensed for (if any), simply type the `show version` command. If there are any licenses found they will be listed as follows:

```
admin@EXALINK-FUSION> show version
Hardware type      : EBFSN-02
Serial number      : EXALINK-FUSION
Software version   : 1.6.0
Software date      : 2017-03-29 14:24:26 +1100 (58417c3)
Licensed features  : switch
End of support     : 2017-10-19 03:14:07

Line card A
-----------
Hardware type      : LC10G-03

Line card B
-----------
Hardware type      : LC10G-03

Module X
--------
Hardware type      : KU115-03
Firmware function  : switch
Firmware date code : 1490699472 (35ad9d29)
```

Adding a License

The license can be entered into the Fusion by issuing the `config license set` command. Once entered, this command will prompt you to enter in the license key provided to you by Exablaze sales/support. You should paste the license text and then press ctrl-D, as follows:

```
admin@EXALINK-FUSION> config license set
Enter license key, press Ctrl-D when done
Serial Number : EXALINK-FUSION
End of Support: 19-Oct-2017
-------------------------------LICENSE KEY--------------------------------
WHMUv8JbVavcbYpXgIFM2sxuPiAH2VN58ax6Abqiadk/aJziWi3jD0WghdK6aa4CxQSJxBNa8sIm
7x72YQcaHSAnLaUp2GWHgbwHynrujDLvnM85DJagDawFnvHHcsaJIap9W0DpYlBb9xvaC+w80SRa
JbSNDxMbCixvfUNSAX5unRgaC3x4aaqzlC/vaj38CKS0YzJarR+RIW+F2vM4spYca6iLKrY/xkvS
IQMiV3GvaN9rknkIa1La7lMurcHhRgNhAHMfCq7ihalkh2RxfX3+aZcRxb86UCRia3JJLnlQs4az
a8VJaa0mR6Niw7ImscAn+1L9QivDKfr6S3JaDa6gCak+r+Ya0aafAjRg4Fm4k+xiaqQRRUfaX3xW
NunRLGvFp4x9dCDkicIfvXFgrq70hKaf1kaR9kl2LwRPIin0aqZy8AaaaDwaiG5dUxCHp62MjUxh
laX9MHBWhq0iaPL/fUpu77x4DuX2KWZLh/i42fyMsdwqqD5lShaqIxm7bzH2a9wU9DsHvCDv7BjZ
/FRAqagUuZLyAQ/gLkmZfCaL6XmDVMaLPxsMvHnUa3fvP8xyLagB77ci3w2h8UwLDwxSgU1mC6Hr
M3agza0MWqagY4H662vAKwLmLI08F+HriJK3xXac19ImHmzxaRaXvq7bvLAlxzGlXx/gZDYBGa0=
----------------------------------------------------------------------------

Number of licenses     : 1

License serial number  : EXALINK-FUSION
License features       : switch
End of support time    : 2017-10-19 03:14:07

License activated
```

Alternatively, the license file can be copied across from your PC to the Fusion using sftp, for example:

```
$ sftp admin@192.168.220.10
admin@192.168.220.10's password:
sftp> put EXALINK-FUSION_license.txt license.txt
```

Once the license file has been copied, the Fusion can be instructed to reload the license to enable any new features using the `config license reload` command:

```
admin@EXALINK-FUSION> config license reload
Reloading license file...
License file reloaded
```

Note

License files are stored in a persistent area of the Fusion's file system and are not overwritten by firmware upgrades. It is possible to load on a license file enabling features that don't exist on your current version, then upgrading to the new version at which point the features will be enabled and available at boot.

Evaluation Licenses

Evaluation licenses can be issued by Exablaze that are time limited. These allow you to trial a certain feature or set of features for a limited time prior to making an upgrade decision.

Warning

When the time limit in the license expires, **all** features in that license will be disabled.

Only one license file can be stored on the Fusion at a time, so adding an evaluation license will replace any purchased license stored on the Fusion. After your evaluation license has expired, or when you have finished testing, be sure to reconfigure your Fusion with your original license file.

You can retrieve the original license from your Fusion using sftp prior to loading on an evaluation license

```
$ sftp admin@192.168.220.10
admin@192.168.220.10's password:
sftp> get license.txt my_original_license.txt
```

If the current license installed on a Fusion is an evaluation license, the expiry date will be listed on the output of `show version`, as follows:

```
Software date      : 2016-01-19 16:12:11 +1100 (5c46718)
Licensed features  : switch
License expiry     : 2016-01-23 11:09:07
```

Please contact Exablaze if you require your original license to be resent at any stage.

Removing Licenses

The installed license can be removed using the `license erase` command, as follow:

```
admin@EXALINK-FUSION> configure license erase
Are you sure you want to remove the license file? y
License removed
```

Rebooting

The `reboot` command can be used to reset the ExaLINK Fusion and load the `startup-config` on boot.

```
admin@EXALINK-FUSION> reboot
Are you sure you want to reboot this device? y
Rebooting system
Connection to EXALINK-FUSION closed by remote host.
Connection to EXALINK-FUSION closed.
```

Delayed Reboot

The Fusion can be instructed to reboot at a given time in the future using the `reboot in` command. This can be useful when changing the management interface settings of a Fusion at a remote site, where there is concern that an mistake in setting the config would disrupt connectivity to the CLI and prevent rollback of the incorrect config.

```
admin@EXALINK-FUSION> reboot in
Usage: reboot in [<hh>:]<mm>
Schedule a reboot of the device

admin@XALINK-FUSION> reboot in 2
Rebooting in 2 minutes
Use "reboot cancel" command to cancel reboot
WARNING: Rebooting in 1 minute 59 seconds
```

Tapping enter on the CLI will show the remaining time left before the device will reset itself:

```
admin@EXALINK-FUSION>
WARNING: Rebooting in 1 minute 45 seconds
admin@EXALINK-FUSION>
WARNING: Rebooting in 1 minute 44 seconds
admin@EXALINK-FUSION>
WARNING: Rebooting in 1 minute 44 seconds
admin@EXALINK-FUSION>
WARNING: Rebooting in 1 minute 43 seconds
```

The delayed reboot can be cancelled using the `reboot cancel` command:

```
admin@EXALINK-FUSION> reboot cancel
Scheduled reboot canceled
```

Installing internal modules

The ExaLINK Fusion is a modular product that can be upgraded and expanded over time, via the addition of FPGA modules and x86 processor modules. Installation of these modules can be done by the user in the field, or the device can be returned to the factory for upgrade.

When ordering an FPGA module, you will be supplied with:

- 1x FPGA Module
- 1x Power cable
- 2x M3x12mm screw
- 2x M3 washer
- 2x M3 spring washer
- 1x M2.5x25mm screw
- 1x M2.5x16mm screw

When ordering an x86 module, you will be supplied with:

- 1x x86 Module
- 1x Power cable
- 2x M3x12mm screw
- 2x M3 washer
- 2x M3 spring washer

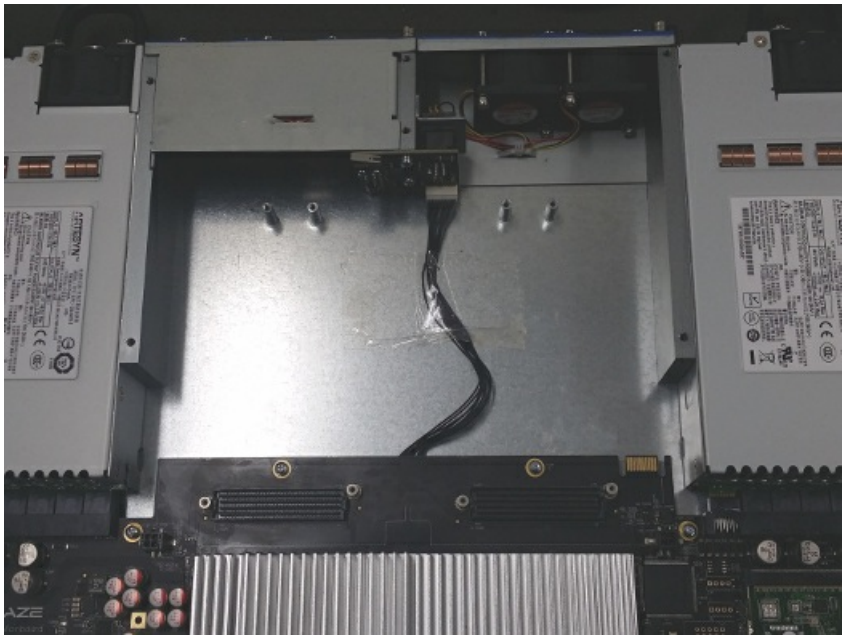
Both modules presented here (FPGA left, x86 right)

Installation

Warning

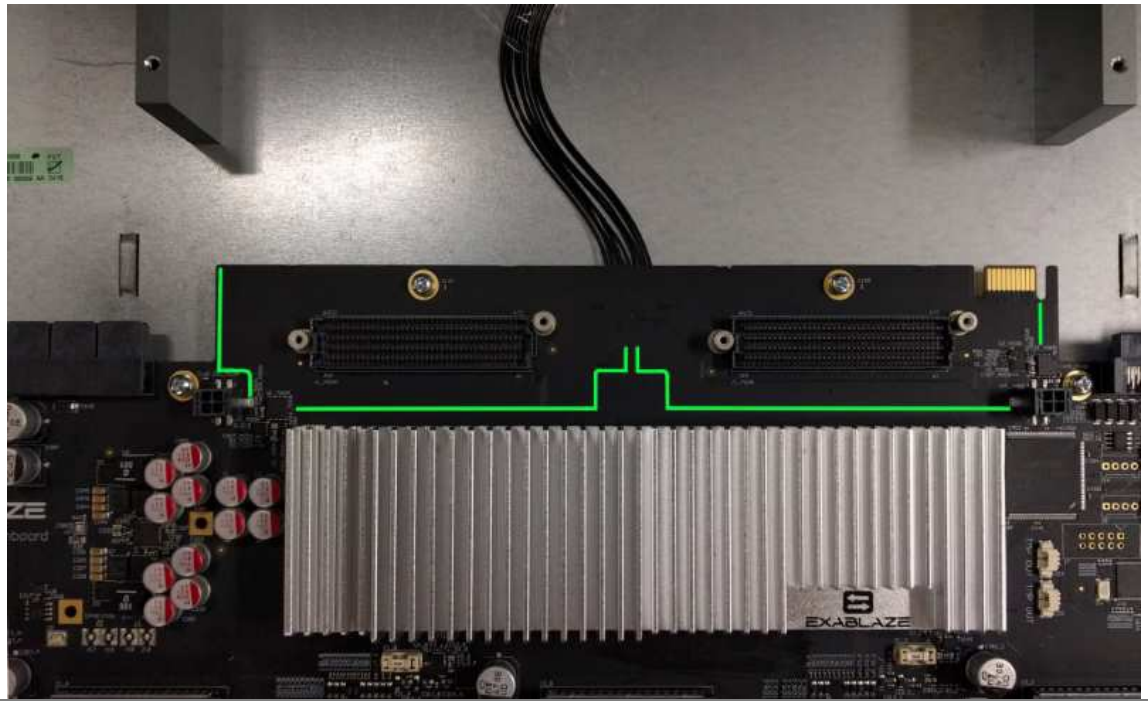
Care must be taken during this installation process to prevent damage to the sensitive electronics in the Fusion and internal modules. Please contact us at Exablaze support if you have any questions about these instructions, or have any difficulty during the installation process.

1. Before installing an internal module ensure the Fusion is unpowered.
2. Unscrew the lid of the ExaLINK Fusion and remove it.



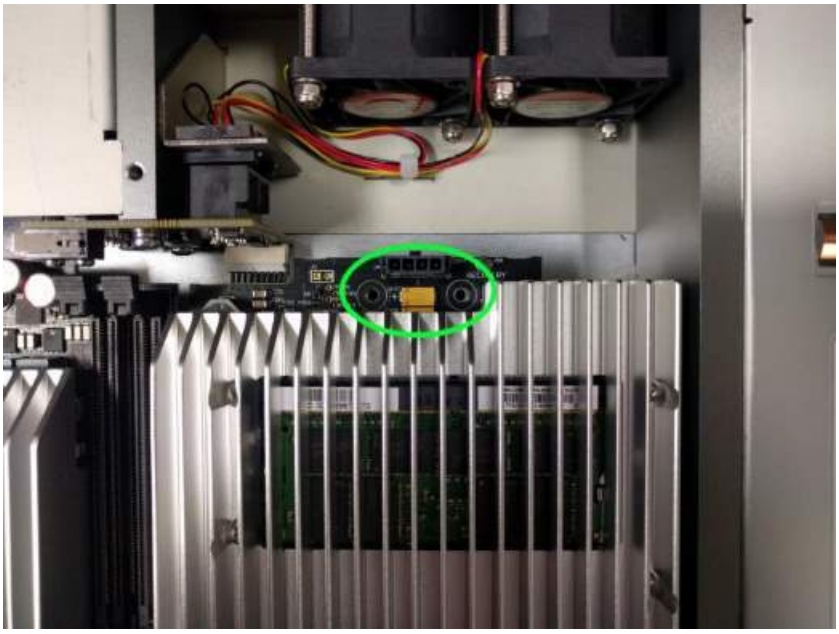
The Fusion with its lid removed, both module bays empty

When installing an internal module, care must be taken to ensure the module is aligned properly with the mainboard. There is a large fine pitch connector on the underside of the internal module that connects to a mating connector on the mainboard. The mainboard has an outline showing the alignment position of each of the internal modules to assist with this process, highlighted in green below:



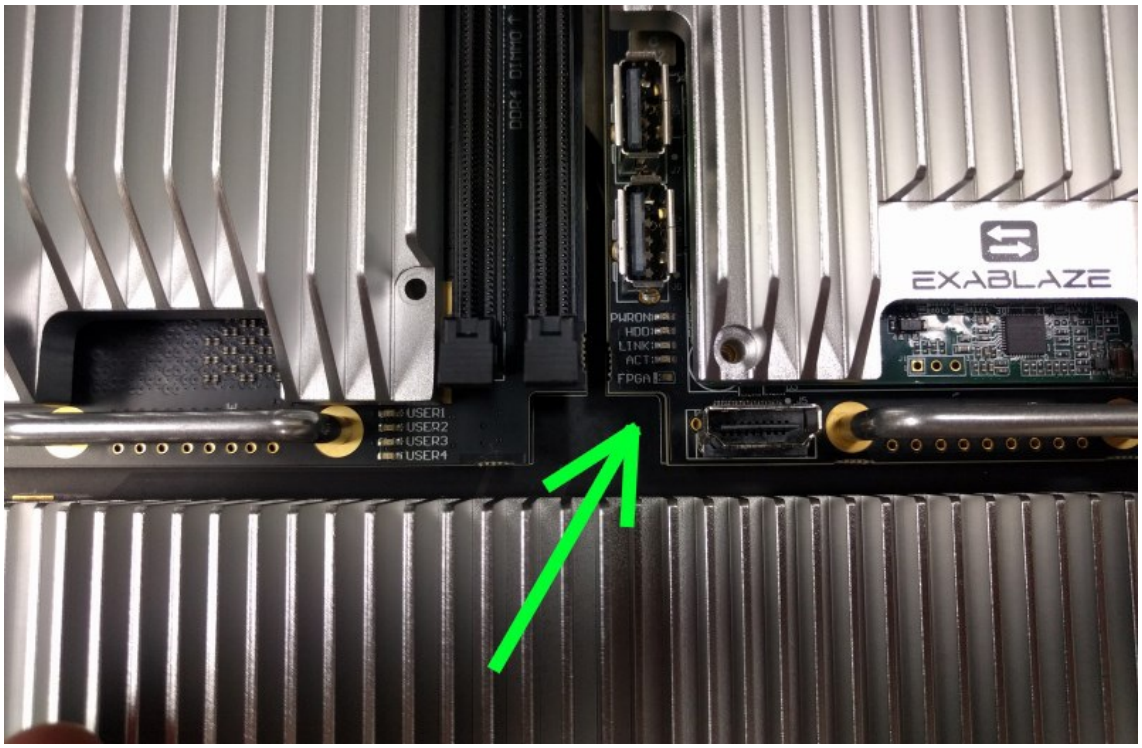
Internal module alignment markers highlighted on the mainboard

3. Insert the module into the desired bay. Start by ensuring the mounting holes at the rear of the module are aligned with the mounting posts in the chassis, as shown below:



Ensure module mounting holes align with mounting posts on the chassis

4. Carefully align the front of the module with the mainboard. As can be seen, the outline of the module circuit board matches the markings on the mainboard.

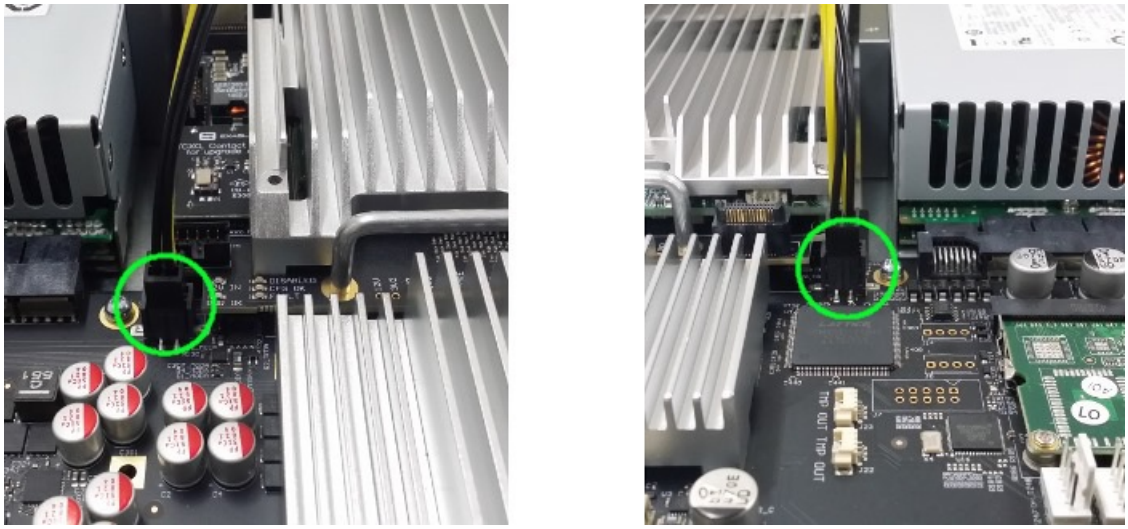


Module aligned with markings on mainboard

5. When the front of the module is aligned with the mainboard, ensure the rear is still aligned with the mounting

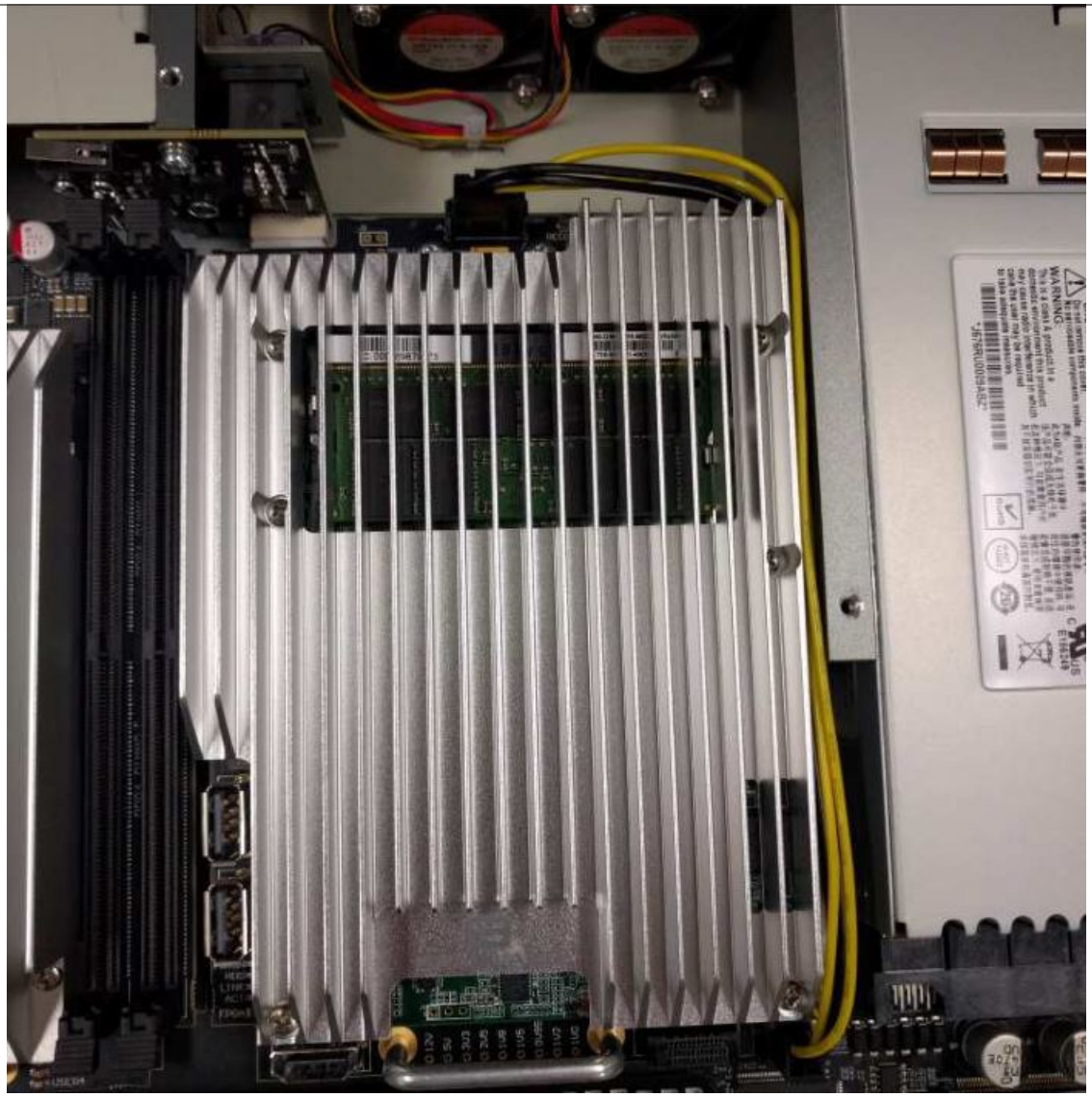
holes.

6. Apply firm pressure downwards to the mounting handle on the internal module to mate the 2 connectors. You should not need to use excessive force - if the connectors are aligned properly you will feel them snap into place with firm pressure.
7. Screw the module into place at the 2 rear mounting holes with M3 screws. The flat washer should make contact with the circuit board, so the split washer should be placed onto the screw before the flat washer.
8. If the FPGA module is being installed, insert the additional 2 screws into the heatsink and screw into place. The longer M2.5 screw is used on the left hand side of the heatsink, and the shorter M2.5 is used on the right.
9. Attach the appropriate end of the power cable to the 4-way power connector at the rear of the module.
10. Attach the power cable to the 2x2 connector on the mainboard, as shown below



The power cable connection points on the mainboard

11. Ensure both ends of the connectors are mated firmly, such that the locking latch engages to prevent the cable from shaking loose.
12. Position the power cable alongside the module heatsink to prevent it from interfering with the lid, as shown below.



Module power cable positioned alongside heatsink

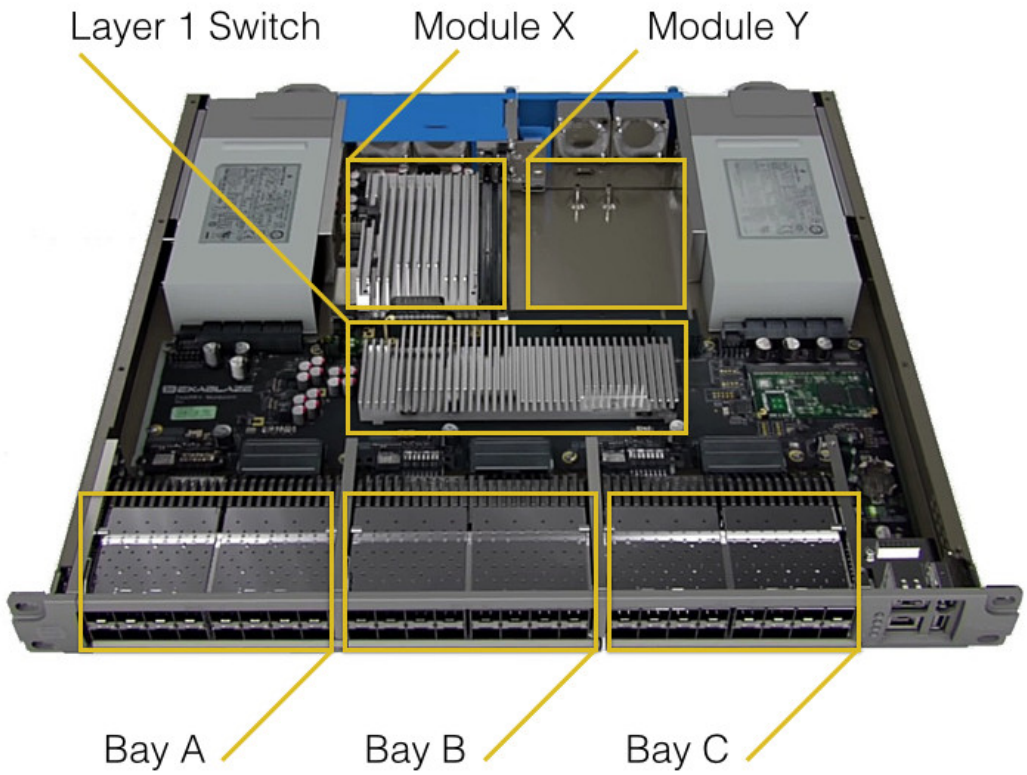
13. Replace the lid and screws

Introduction

The ExaLINK Fusion has a modular architecture that is uniquely configurable. This overview explains the architecture of the ExaLINK Fusion and how the user interface exposes this functionality.

The ExaLINK Fusion architecture is centered around a dynamically reconfigurable layer 1 switch. The layer 1 switch allows any of its inputs to be dynamically patched or replicated to multiple outputs. Since forwarding happens at layer 1 it involves very little latency overhead and is protocol agnostic.

Each of the three front panel line card bays have 16 dedicated inputs and outputs to the layer 1 switch. The currently available line card includes 16 SFP+ ports that are suitable for 10 GbE and 1GbE operation. Since all three line cards connect directly to the layer 1 switch, the user can create any combination of layer 1 interconnections between front panel ports using patch or tap objects. From left to right, the line card bays are designated A, B and C.



Overview of the ExaLINK Fusion architecture.

Also connected to this layer 1 switch are two internal module bays. Both module bays have 48 dedicated input and outputs connected to the layer 1 switch. Any of these 48 inputs or outputs can be dynamically reconfigured to connect directly to any front panel port, or to any input or output on the adjacent module. From left to right, module bays are referred to as X and Y. When ordered with switching or muxing functionality included, bay X will have a FPGA module fitted. When performing switching or muxing, the management interface will automatically route the required signals from the front panel to inputs or outputs on the module bay.

The FPGA module that is included in the mux and switch capable ExaLINK Fusions can be configured

to run different firmware types. At present, three firmware versions are available:

- The `mux` firmware is optimized for aggregation functionality. This firmware can be used to combine up to 47 'downstream' ports into a single upstream port. Up to four 'upstream' ports can be configured with different combinations of downstream ports. Configuration is exposed using the mux object. Upstream ports can also be used as logging and timestamping ports, and can be configured in this mode using the mirror object.
- The `switch` firmware performs general purpose layer 2 switching. With this firmware, a 48 port layer 2 switch can be configured using the switch object. Multiple, independent switches can be defined that incorporate any of the front panel ports. The switch firmware can also be used for mux and mirror objects.
- The `fastmux` firmware is optimized for lowest latency 10G aggregation. Currently, only one mux object per FPGA module is supported with capability of aggregating 15 downstream ports into a single upstream port.

The user can change between firmware functions at any time using the command line.

Command Line Interface

The ExaLINK Fusion can be configured via a command line interface. This interface can be accessed via SSH, telnet, or an industry standard serial interface.

Configuration mode

The command line interface can be used to both inspect the current configurations of the ExaLINK Fusion, and modify the configuration. The command line can be used in either a modal or non-modal way. For example, commands that change the current configuration are prefixed with `config`. When executing multiple configuration commands, it is convenient to enter config mode. For example, to set a port alias, one can enter:

```
admin@EXALINK-FUSION> configure port A1 alias exchange
Alias set on port A1
```

Alternatively, you can enter config mode and then set the port alias:

```
admin@EXALINK-FUSION> config
admin@EXALINK-FUSION(config)> port A1 alias exchange
Alias set on port A1
```

Note: in the above example that the command line prompt indicates that config mode is in use. To exit a mode, simply type `exit`:

```
admin@EXALINK-FUSION(config)> exit
admin@EXALINK-FUSION>
```

Note: that the interface has dropped back to the default mode. Importantly, typing `exit` within the default mode will log you out of the device.

Modal states can be nested, and typing `exit` will drop out of that level and up to the previous level. Typing `end` will drop out of all levels and return you to the top modal level. For example:

```
admin@EXALINK-FUSION> config
admin@EXALINK-FUSION(config)> port A1
admin@EXALINK-FUSION(config-port:A1)> end
admin@EXALINK-FUSION>
```

Creating and removing objects

The ExaLINK Fusion is configured by creating or removing objects. An object represents a specific configuration and usually has one or more ports as members, as well as a number of properties. There is no limit to the number of objects that can be created on the ExaLINK Fusion apart from that imposed by the number of ports on the device.

All objects are created or removed using the same convention. Creation of an object is achieved by entering the desired object type followed by any required properties. For example, to create a tap:

```
admin@EXALINK-FUSION> config tap A1 A4
Added input tap on port "A1", send to port "A4"
```

Removal of an object from the configuration is the same as creating the object, however the command is prefixed with `no`. To remove the tap created above, one would enter:

```
admin@EXALINK-FUSION> config no tap A1 A4
Removed input tap on port "A1", send to port "A4"
```

Some objects are more complicated than taps. For example, the mux object and switch object. These objects have a name property and are configured from within their own mode, however the convention for creation and removal is the same.

Help and autocompletion

The interface includes help and tab completion. Help can be requested at any point during use by pressing the `?` character. If there are multiple parameter options that are valid from that point on, pressing `?` will list them, for example:

```
admin@EXALINK-FUSION> config management ?
configure management access-list allow Allow connections from the supplied IPv4 add
configure management access-list deny  Deny connections from the supplied IPv4 addr
configure management address dhcp      Configure management interface using DHCP
configure management address none      Disable IPv4
configure management address static    Configure a static IPv4 address on the manag
configure management name-server       Configure name servers for the management in
```

If there are no further parameter options valid, and `?` is pressed, detailed usage information will be displayed, for example:

```
admin@EXALINK-FUSION> config management address static ?
Usage: configure management address static <address> <netmask> [<gateway>]
Configure a static IPv4 address on the management interface
```

Autocompletion and suggestions for any command can be obtained by pressing tab after having entered any partial command. For example, when creating a tap object, a list of available source or output ports can be obtained by pressing tab whilst the interface is expecting a port. In this example, the interface shows a list of available source ports after pressing tab:

```
admin@EXALINK-FUSION> config tap
A1        A2        A3        A4        A5        A6        A7        A8
A9        A10       A11       A12       A13       A14       A15       A16
down      exchange  output
```

Command pipelines

> This feature requires version 1.11.0 or later

The command line interface supports redirecting command output to standard Unix utilities such as `grep` and `less`, for example:

```
admin@EXALINK-FUSION> show port A16 | grep 'Link status'
Link status        : up
```

Because this feature allows the user to run arbitrary shell commands, it is only available to users with the `admin` role.

Login banner

A login banner can be installed by uploading a file to the device with the name *banner.txt* using sftp, for example:

```
$ sftp admin@192.168.220.10
admin@192.168.220.10's password:
sftp> put banner.txt
```

The contents of *banner.txt* will be displayed prior to the login prompt, for example:

```
$ ssh admin@192.168.220.10

You are attempting to log into an ExaLINK Fusion - unauthorized access prohibited!

admin@192.168.220.10's password:
admin@192.168.220.10>
```

MOTD banner

A message of the day banner can be installed by uploading a file to the device with the name *motd.txt* using sftp, for example:

```
$ sftp admin@192.168.220.10
admin@192.168.220.10's password:
sftp> put motd.txt
```

The contents of *motd.txt* will be displayed on a successful login to the command line interface, for example:

```
$ ssh admin@192.168.220.10
admin@192.168.220.10's password:

Welcome.  This device is located in ROW AF, CAB 2, RU 36

admin@192.168.220.10>
```

Session timeouts

An idle timeout can be configured where command line sessions will be logged out after a certain amount of inactivity. This can be set with the `session-timeout` command and passing in a timeout value in seconds, for example:

```
admin@EXALINK-FUSION> config session-timeout 600
Session idle timeout set
```

A maximum value of 60 minutes applies to the session timeout (3600s).

SSH Keys

Users can authenticate their login into the Fusion through the use of SSH keys, rather than entering a password. This is done by adding the user's public key into the Fusion, for example:

```
admin@EXALINK-FUSION> config user admin sshkey "ssh-rsa AAAAB3NzaC1yc2EAAAA
DAQABAAABAQC15sjG4cYSAbYU0VIwPkdQkIKb0A2xxhPCjOAnzt91CrRQZibirZNqqW71TX3QVt
Ruqp2ZQjo19Nd9bk2iwa3qDITQIOlRdSJgwEBdklfgjkrjkd8KLSDfklwjkrjksjkXyztI3sKRM
mcYOEFjt9Bv+5JPw3o3Pja5GQOO5VEjM//QhsbMZ+G/4Sfx5GiLTaktenqNWflPaMcDWqq1wuuf
6mG7lEM55UDp5xWRrh5vIfyOh9Llosdfsasjkejk89sv9+cjkfhjksjhcjkhskhkljashdfiuhs
dfv98734kjFDjhjhdjhbfjh489jkJKZDJKHFVjkhi6SBbTu5v9 bob@myserver.com"
Added ssh key for user "admin"
```

The user will then be able to login without being prompted for a password:

```
$ ssh admin@EXALINK-FUSION
admin@EXALINK-FUSION>
```

In order to remove a key for a user, use the `no` form of the command, passing in the public key to remove, for example:

```
admin@EXALINK-FUSION> config user admin no sshkey "ssh-rsa AAAAB3NzaC1yc2EA
AADAQABAAABAQC15sjG4cYSAbYU0VIwPkdQkIKb0A2xxhPCjOAnzt91CrRQZibirZNqqW71TX3
QVtRuqp2ZQjo19Nd9bk2iwa3qDITQIOlRdSJgwEBdklfgjkrjkd8KLSDfklwjkrjksjkXyztI3s
KRMmcYOEFjt9Bv+5JPw3o3Pja5GQOO5VEjM//QhsbMZ+G/4Sfx5GiLTaktenqNWflPaMcDWqq1w
uuf6mG7lEM55UDp5xWRrh5vIfyOh9Llosdfsasjkejk89sv9+cjkfhjksjhcjkhskhkljashdfi
uhsdfv98734kjFDjhjhdjhbfjh489jkJKZDJKHFVjkhi6SBbTu5v9 bob@myserver.com"
Removed ssh key for user "admin"
```

Configuration management

Showing and saving the current config

The configuration of the ExaLINK Fusion is described by a `running-config`. The `running-config` contains all of the settings that are currently in effect on the device.

```
admin@EXALINK-FUSION> configure show running-config
hostname EXALINK-FUSION
management address static 192.168.220.10 255.255.255.0
mux my_mux
  mode layer2
  port upstream A2
  port A1
timesync ptp 0
module X
  function mux
telnet enable
```

To make the running-config persistent across reboots of the device, it must be saved to the `startup-config`. To save the `running-config` to the `startup-config`, use the `copy` command:

```
admin@EXALINK-FUSION> configure copy running-config startup-config
Saved running config to startup config
```

Any setting in the `startup-config` will be applied to the device at boot time.

Erasing configurations

The `startup-config` can be erased with the `erase startup-config` command. This command will revert the Fusion's power-up settings to the factory default settings.

Various elements of the `running-config` can be erased, depending on how much of the config you want to reset.

```
admin@EXALINK-FUSION(config)> erase running-config
Usage: erase startup-config
       erase running-config [all] [management] [module] [data-plane]
Erase configuration files
```

- `all` will reset the running-config to the factory default settings, potentially rendering the device inaccessible from the current connection.
- `management` will reset only those parts of the running-config related to the management of the device, not the high speed port config. For example, hostname management address, passwords, logging etc are all covered in this reset group. This could potentially render the device inaccessible from the current connection.
- `data-plane` will reset all port configs and delete all patch/tap/mux/switch and mirror objects.
- `module` will reset any internal module specific settings, for example the selection between `mux` or `switch` firmware, or any `custom` module settings applied.

Note

Erasing the running-config takes place immediately after the command is executed. A warning is given prior to executing any `erase running-config` command that could potentially disrupt your connection to the management interface.

Backing up the config

The `startup-config` is stored within the ExaLINK Fusion as a number of .conf files. These can be retrieved from the device using sftp for backup purposes, and put back onto the device if necessary.

```
$ sftp admin@192.168.220.10
admin@192.168.220.10's password:
Connected to 192.168.220.10.
sftp> get *.conf
```

User Management

Overview

The factory default setting of the ExaLINK Fusion has one user defined, *admin*. Other users can be created which have limited permissions for monitoring etc.

A number of pre-defined roles are available on the Fusion. These are:

- `guest`
- `monitor`
- `user`
- `operator`
- `admin`

The figure below depicts the capabilities of each role.

Capability	Examples	Role				
		Guest	Monitor	User	Operator	Admin
Can read system/chassis information	`show uptime`	Yes	Yes	Yes	Yes	Yes
	`show diagnostics`	Yes	Yes	Yes	Yes	Yes
	`show temperature`	Yes	Yes	Yes	Yes	Yes
	`show power-supply`	Yes	Yes	Yes	Yes	Yes
Can read all information	`show running-config`	No	Yes	Yes	Yes	Yes
	`show startup-config`	No	Yes	Yes	Yes	Yes
	`show port A1`	No	Yes	Yes	Yes	Yes
	`show mux`	No	Yes	Yes	Yes	Yes
	`show timesync`	No	Yes	Yes	Yes	Yes
	`show lldp neighbors`	No	Yes	Yes	Yes	Yes
Can configure ports	`conf port A1 speed 1000`	No	No	Yes	Yes	Yes
	`conf port A1 disable`	No	No	Yes	Yes	Yes
	`conf port A1 alias my_port`	No	No	Yes	Yes	Yes
Full access to non-security related settings	`conf patch A1 A2`	No	No	No	Yes	Yes
	`conf mux my_mux`	No	No	No	Yes	Yes
	`conf switch my_switch`	No	No	No	Yes	Yes
	`conf timesync ptp 0`	No	No	No	Yes	Yes
	`reboot`	No	No	No	Yes	Yes
Full access	`bash`	No	No	No	No	Yes
	`conf copy running-config startup-config`	No	No	No	No	Yes
	`conf erase startup-config`	No	No	No	No	Yes
	`conf remote-logging disable`	No	No	No	No	Yes
	`conf telnet enable`	No	No	No	No	Yes
	`conf tacacs secret pass123`	No	No	No	No	Yes
	`conf management address dhcp`	No	No	No	No	Yes
	`conf snmp disable`	No	No	No	No	Yes
	`conf user my_new_user`	No	No	No	No	Yes
	`conf update file update.tar`	No	No	No	No	Yes

Note

The user editing described on this page relates to local users on the Fusion being configured only. It is not possible to edit TACACS+ users on the Fusion.

Also, when TACACS+ is enabled any locally defined users will not be able to login using their local credentials. Any local accounts that have the same username as an account on the TACACS+ server will take on roles/permissions as defined on the server.

Creating Users

Users are created using the `user` command, for example to create a new user `donald`:

```
admin@EXALINK-FUSION> config user donald
Creating new user "donald"
New password:
Re-enter new password:
Created new user "donald"
```

To delete a user, use the `no user` command with the user to be deleted, for example:

```
admin@EXALINK-FUSION> config no user donald
Deleted user "donald"
```

Note that the *admin* user cannot be deleted, nor can it have the `admin` role removed from it.

Changing Password

Any user with the `admin` role is able to change the password of any other user using the `password` command, for example:

```
admin@EXALINK-FUSION> configure user donald password
Changing password for user "donald"
New password:
Re-enter new password:
Password changed for user "donald"
```

Assigning Roles

Once a user has been created, a role should be added specifying the access this new user should have.

```
admin@EXALINK-FUSION(config-user:donald)> role monitor
Added role "monitor" for user "donald"
```

Whilst it is possible to add multiple roles to a single user, the current (fixed) set of possible roles means that there is no point in doing this. For example if a user was assigned both the `monitor` and `operator` roles, as the `operator` role is capable of performing all of the actions enabled by the `monitor` role, adding `monitor` is unnecessary.

Roles can be removed using the `no role` form of the command, for example:

```
admin@EXALINK-FUSION(config-user:donald)> no role monitor
Removed role "monitor" for user "donald"
```

Viewing Users

The list of users and each of their roles can be viewed using the `show user` command:

```
admin@EXALINK-FUSION> show user
Username Type  Roles
-------- ----- -----------
admin    local admin
donald   local operator
fred     local guest
jane     local admin
```

SSH Keys

Users can add & remove an SSH key for themselves to facilitate authenticated logins without using a password. SSH keys can only be set/removed for the current user. To add a key:

```
admin@EXALINK-FUSION(config-user:admin)> sshkey
Usage: sshkey <"ssh key">
Add a public ssh key for the selected local user.

admin@EXALINK-FUSION(config-user:admin)> sshkey "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQ
Added ssh key for user "admin"
```

To remove a key, use the "no" form of the command:

```
admin@EXALINK-FUSION(config-user:admin)> no sshkey "ssh-rsa AAAAB3NzaC1yc2EAAAABIwA
Removed ssh key for user "admin"
```

To see what SSH keys are added to a particular user, you can drop to the bash shell as follows:

```
admin@EXALINK-FUSION> bash

ExaLINK Fusion shell

admin@EXALINK-FUSION:~$ cat /home/admin/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAxTW4KRfzjiuEDu3thDRGrJeD5dwUDQdSYyeMy/8Yy/CMiRj
```

Diagnostics

The ExaLINK Fusion incorporates many in-built diagnostics functions that can be used to monitor the device.

Sensors and monitors

Power

The state of the power supplies in the ExaLINK Fusion can be shown using `show power-supply`:

```
admin@EXALINK-FUSION> show power-supply
Power supply 0
--------------
Manufacturer : EMERSON
Model        : DS460
Input        : 243.0 V  0.3 A
Output       : 12.2 V  3.8 A
Temperature  : 23.5 C (ambient)  34.0 C (internal)

Power supply 1
--------------
Manufacturer : EMERSON
Model        : DS460
Input        : 240.5 V  0.3 A
Output       : 12.2 V  5.0 A
Temperature  : 28.0 C (ambient)  44.2 C (internal)
```

It is also possible to show the current power on many of the internal sensors within the device. This can be done with the `show power` command:

```
admin@EXALINK-FUSION> show power
             Voltage  Current
             -------  -------
Line card A  12.2 V    1.1 A
Line card B  12.3 V    1.5 A
Line card C  12.3 V    1.1 A
Module X     12.3 V    3.3 A
Module Y     12.3 V    0.0 A
```

Temperature

The `show temperature` command can be used to show the temperature of various internal modules within the ExaLINK Fusion.

```
admin@EXALINK-FUSION> show temperature
             Temperature
             ------------------------------
Crosspoint   41.3 C  40.4 C  36.1 C  39.5 C
Line card A  35.6 C
Line card B  32.4 C
Line card C  33.9 C
Mainboard    29.6 C  27.2 C
Module X     36.0 C  45.0 C
```

Fans

The `show fan-speed` command shows the speed of the four fans in the ExaLINK Fusion. These fans are individually monitored and controlled.

```
admin@EXALINK-FUSION> show fan-speed
       Fan speed
       ---------
Fan 0  8160
Fan 1  8040
Fan 2  8160
Fan 3  8160
```

Viewing local logs

The system log can be viewed from the ExaLINK Fusion command line, by using the following command:

```
show log
```

By default this will show all the entries that have priority of `notice` or higher, and have a system log identifier of `exalinkd` or `switchd`. The command accepts various parameters that allow the filters to be adjusted. Below are some examples of how to apply the filters.

To show messages with priority `error` or higher:

```
show log level error
```

Valid priority levels are:

- `emergency`
- `alert`
- `critical`
- `error`
- `warning`
- `info`
- `debug`

To show messages with a syslog identifier of `systemd` or `exalinkd`:

```
show log systemd exalinkd
```

To show the last 20 messages from `all` syslog identifiers of level `debug` and higher:

```
show log all level debug last 20
```

To show the first five messages:

```
show log reverse last 5
```

To filter the normal output for entries that contain the word `authenticated`:

```
show log contains authenticated
```

Remote logging

The ExaLINK Fusion supports remote logging using the syslog protocol. To configure remote logging, use the `remote-logging` command. The first step is to configure a remote target for log messages. Both TCP and UDP modes of syslog are supported. As an example, to log all error messages using UDP to a host at 192.168.220.13, use the command:

```
admin@EXALINK-FUSION> configure remote-logging target udp 192.168.220.13 all err
Remote logging configuration updated
```

Multiple targets can be configured using this command. Ports other than the default of 514 can also be specified, for example:

```
admin@EXALINK-FUSION> configure remote-logging target udp 192.168.220.13 all err
Remote logging configuration updated
admin@EXALINK-FUSION> configure remote-logging target tcp 192.168.220.14 all all
Remote logging configuration updated
admin@EXALINK-FUSION> configure remote-logging target udp 192.168.220.15:12345 all
Remote logging configuration updated
```

To remove all remote logging targets use the `no` form of the remote logging command:

```
admin@EXALINK-FUSION> configure no remote-logging
Reset remote-logging configuration
```

Remote logging can be globally enabled or disabled using the `enable` or `disable` command:

```
admin@EXALINK-FUSION> configure remote-logging enable
Remote logging enabled
```

To view all of the current remote logging targets as well as the global state, use the `show remote-logging` command:

```
admin@EXALINK-FUSION> configure show remote-logging
Type Address            Facility Level
---- ------------------ -------- -----
udp  192.168.220.13       all      err
udp  192.168.220.14       all      all
udp  192.168.220.15:12345 all      debug

remote-logging is enabled
```

Power failure

The management processor and interface in the ExaLINK Fusion will keep running for several seconds after the power has been completely removed from the device. This allows for the message that a power failure event has occurred to be sent to the remote logging server. The following entries can be expected in the logs when power is removed:

```
Jul  2 01:41:45 EXALINK-FUSION kernel: [74253.339458] Power fail event detected
Jul  2 01:41:45 EXALINK-FUSION exalinkd[542]: exalink: Power fail event detected
Jul  2 01:41:45 EXALINK-FUSION exalinkd[542]: exalinkd: Shutting down on power fail
Jul  2 01:41:45 EXALINK-FUSION exalinkd[542]: system: Rebooting in 5 seconds
```

If SNMP traps are enabled, a SNMP notification will also be sent.

Warning

Note that if power is removed and restored quickly, the management processor will automatically reboot the device in order to ensure all hardware has been properly re-initialized.

Non-Volatile Log

The Fusion contains a small log of specific events that is non-volatile, ie a log that is not cleared on system reboot. This log includes the following events:

- Power fail
- User reboot
- Firmware update
- Tamper events, ie lid removal

The `show nvlog` command can be used to display this log, for example:

```
admin@EXALINK-FUSION> sh nvlog
2017-02-22T23:56:10 reset-reason *system warm_sw
2017-02-23T03:59:39 exalinkd *system powerfail
2017-02-23T03:59:39 system *system reboot
2017-02-23T03:59:57 reset-reason *system cold
2017-02-23T04:06:14 system *system reboot
2017-02-23T04:06:14 exalinkd *system powerfail
2017-02-23T06:18:18 exalink_hw *system tamper 2017-02-23T05:16:13
2017-03-29T01:13:45 update_manager admin update_file exalink_fusion_1.6.0.tar
```

Debug dump

It is possible to generate a compressed file containing a large amount of useful information showing the state of the ExaLINK Fusion. This can be sent to the Exablaze support team to aid in resolving any issues.

Note: this file contains version and config information, hardware status, byte/packet counters etc. The dump file does not contain payload data from any packets received by the device.

To generate the dump file:

```
admin@EXALINK-FUSION> debug dump
Debug data written to debug/debug_info_20160112T172853.tar.gz
```

This tarball can then be retrieved by sftp or tftp, for example:

```
$ sftp admin@192.168.220.10
admin@192.168.220.10's password:
sftp> get debug/debug_info_20160112T172853.tar.gz
```

Statistics logging

Overview

This feature requires version 1.9.0 or later

The ExaLINK Fusion can log time series statistics to a remote database. Only InfluxDB databases are supported in the current version.

Collected statistics

System sensors and information including:

- Temperature sensors
- Current sensors
- Power supply sensors
- Fan speeds
- Management CPU load average
- Management CPU available memory

Port status and port counters including:

- SFP present
- Signal status reported by the SFP, ie. light received or not
- CDR lock
- Current RX link state
- RX link state change count
- Packet and byte counters for RX and TX
- CRC error counters for RX (FCS)

Switch statistics including:

- Buffer memory used (Fusion HPT only)

Measurements from the Fusion HPT time synchronization algorithm:

- Raw offset at PPS edge
- Smoothed offset
- Raw frequency error of the oscillator

Configuration

Before statistics logging can be enabled, a database connection must be configured.

InfluxDB connection

To set up a connection to an InfluxDB server, the server address and database name are needed. This is configured using the following commands:

```
admin@EXALINK_FUSION> config statistics influxdb server myserver
InfluxDB server address configured
admin@EXALINK_FUSION> config statistics influxdb database mydb
InfluxDB database name changed
```

If the InfluxDB server listens on a port other than 8086, the port can be specified using the following command syntax:

```
admin@EXALINK_FUSION> config statistics influxdb server myserver:1234
InfluxDB server address configured
```

By default, the ExaLINK Fusion will use HTTP to connect to the InfluxDB server. To enable HTTPS, use the `ssl` command:

```
admin@EXALINK_FUSION> config statistics influxdb ssl
Enabled SSL for InfluxDB
```

If the InfluxDB database requires authentication, this can be configured using the `authentication` command. For example, to authenticate using the username `donald` and password `mypassword`:

```
admin@EXALINK_FUSION> config statistics influxdb authentication donald mypassword
Enabled InfluxDB username/password authentication
```

Use the `no` form of these commands to disable HTTPS and disable authentication, for example:

```
admin@EXALINK_FUSION> config statistics influxdb no ssl
Disabled SSL for InfluxDB
admin@EXALINK_FUSION> config statistics influxdb no authentication
Disabled InfluxDB authentication
```

Enable statistics logging

After a database connection is set up, statistics logging can be enabled using the `statistics enable` command:

```
admin@EXALINK_FUSION> config statistics enable
Statistics logging enabled
```

At this point, statistics will be published to the remote server every 60 seconds. Currently, the logging interval is fixed and not configurable.

The `statistics disable` command can be used to disable statistics logging:

```
admin@EXALINK_FUSION> config statistics disable
Statistics logging disabled
```

Configuring ports

This section covers configuration of front panel ports. Port related functions are accessed through the `port` command. Without the `config` modifier these commands are limited to showing the current port status. With the `config` modifier it is possible to change settings on the port. Ports are analogous to interfaces on other vendor's products.

All of the commands shown in this section can be entered either on a single line or from within the port modal state. To enter the modal state for a given port, simply enter:

```
admin@EXALINK-FUSION> config port A1
admin@EXALINK-FUSION(config-port:A1)>
```

When in the port modal state, it is possible to omit the `config port A1` prefix from commands.

The keyword `interface` can be used instead of `port` anywhere on the system, for example:

```
admin@EXALINK-FUSION> show interface A1
```

Using `interface` instead of `port` requires version 1.10.0 or later

Naming conventions

The ExaLINK Fusion has three front panel line card bays. From left to right, these bays are referred to as A, B and C. Ports are referenced by their line card bay, followed by the port number on the line card. As an example, port number 1 in line card bay A is referred to as port A1.

A port can be given an alias, which is interchangable with its positional name. This is a convenient way to associate ports with devices on the network. For example, if port A1 was connected to an exchange, an alias could be configured as follows:

```
admin@EXALINK-FUSION> config port A1 alias exchange
Alias set on port A1
```

An alias can be removed using the `no` form of the `alias` command:

```
admin@EXALINK-FUSION> config port A1 no alias
Alias removed on port A1
```

If a more detailed description of the port is required a port description can be set. For example to add a description to the exchange facing port we described above:

```
admin@EXALINK-FUSION> config port A1 description "Primary exchange order line"
Description set on port A1
```

Port speed

The ExaLINK Fusion supports both 1GbE and 10GbE standards. The speed of a port should be set such

that it matches the speed of the device that is connected to it. Even if a port is operating in a pure layer 1 configuration, it should still be configured to the correct speed as this will ensure optimal signal recovery.

To configure the speed of a port, use the speed command as shown:

```
admin@EXALINK-FUSION> config port A1 speed 1000
Port A1 speed set to 1000
```

The speed is set in megabits per second (Mbps) and can be either 1000 or 10000.

Port details

Detailed port statistics are available on all ports, and can be accessed with the `show` command. For example, to show detailed statistics on port A1:

```
admin@EXALINK-FUSION> show port A1
Port name          : A1
Alias              : exchange
Description        : Primary exchange order line
Speed              : 1 Gbps
Autonegotiation    : enabled (setting applies for switch and mux ports only)
LLDP agent         : receive only
Link generator     : disabled
Status             : SFP present, signal detected

Vendor name        : OEM
Vendor OUI         : 00-90-65
Part number        : SFP-10G-SR
Revision           : 02
Connector type     : LC
Transceiver codes  : 10G Ethernet: 10G Base-SR
Serial encoding    : 64B/66B
Bit rate (nominal) : 10300 Mbps
Laser wavelength   : 850 nm

Serial number      : CSSR1405880
Date code          : 140501

Temperature        : 37.6836 C
Supply voltage     : 3.3058 V
Tx bias current    : 3.751 mA
Tx output power    : 0.61 mW (-2.1 dBm)
Rx input power     : 0.44 mW (-3.6 dBm)

CDR lock           : yes

Link status        : up
Link up/down count : 3
Autoneg status     : complete
Packets received   : 373
Bytes received     : 223300
Receive errors     : 0
Runt frames        : 0
Pause frames       : 0
Packets sent       : 82
Bytes sent         : 64104
Errors sent:       : 0
```

In the printout above the following statistics are present:

- The `port name`, `alias`, and `description` map to the settings described in the port name section.
- The `speed` field shows the currently configured port speed.

- The `autonegotiation` field shows whether autonegotiation on 1GbE is turned on. This setting does not apply if the port is operating in layer 1 mode.
- The `status` field shows whether an SFP or cable is present, and whether that SFP is reporting a signal present.
- The details of the installed SFP or cable are available in the fields from `vendor name` through to `Rx input power`.
- The state of the clock and data recovery circuitry on the line card are reported in the `CDR lock` field. When a valid 1GbE or 10GbE physical layer signal is present, this field will report `yes`, otherwise it reports `no`.
- When an FPGA module is present, the `link status` field shows whether the MAC within the FPGA has acquired a lock to the incoming data stream.
- The `Link up/down count` shows the number of times the link has gone up & down, which can be useful in debugging connection issues.
- `Autoneg status` shows the current status of 1GbE Autonegotiation.
- Packet and byte sent and receive count statistics.

Port details, verbose

This requires version 1.11.0 or later

A more verbose counter view can be shown by adding the `detailed` keyword, for example:

```
admin@EXALINK-FUSION> show port A1 detailed
Port name            : A1
...
...
CDR lock             : yes

Link status          : up
Link up/down count   : 3
Autoneg status       : complete
Packets received     : 373
 - Unicast           : 63
 - Multicast         : 310
 - Broadcast         : 0
 - 64B frames        : 0
 - 65-127B frames    : 0
 - 128-255B frames   : 373
 - 256-511B frames   : 0
 - 512-1023B frames  : 0
 - 1024-1518B frames : 0
 - 1519-1922B frames : 0
Bytes received       : 223300
Receive errors       : 0
Runt frames          : 0
Pause frames         : 0
Packets sent         : 82
 - Unicast           : 82
 - Multicast         : 0
 - Broadcast         : 0
 - 64B frames        : 0
 - 65-127B frames    : 0
 - 128-255B frames   : 0
 - 256-511B frames   : 82
 - 512-1023B frames  : 0
 - 1024-1518B frames : 0
 - 1519-1922B frames : 0
Bytes sent           : 64104
Errors sent:         : 0
```

Tip

If the source of data for a port is a layer 1 path, then the source port is shown in the output of `show port`. For example, consider the case where port B1 and C1 were patched together, the output of `show port B1` would show:

```
admin@EXALINK-FUSION> show port B1
...
...
Layer 1 Source     : C1, link status up
```

Note this feature requires version 1.11.0 or later.

Resetting Port Counters

The port statistics counters can be reset.

```
admin@EXALINK-FUSION> configure port A1 reset counters
Port A1 counters reset
```

Port overview

In addition to the detailed output available for all ports, an overview can be obtained using the generic `show port` command without specifying a port. This command will show a summary of all ports installed in the device along with its current status.

```
admin@EXALINK-FUSION> show port
Port Status           Description
---- ---------------- --------------------------
A1   up               Primary exchange order line
A2   up
A3   down (no signal)
A4   disabled
A5   disabled
A6   down (no signal)
A7   disabled
A8   up
A9   up
A10  down (no SFP)
A11  disabled
A12  disabled
A13  disabled
A14  disabled
A15  disabled
A16  disabled
```

DWDM SFP Transmission Tuning

This feature requires version 1.8.0 or later

The ExaLINK Fusion supports the configuration of SFF-8690 compliant DWDM tunable SFP connectors through the `config port` interface, allowing for the stacking of multiple optical signals onto one fibre cable by using wavelength division multiplexers. Configuration is done using the `tx-tuning` command. It takes one of two parameters: `channel` or `wavelength`.

`channel` can be used to specify a value from a range stored in the SFP:

```
admin@EXALINK-FUSION> configure port A1 tx-tuning channel 1
Port A1 tunable transmitter configuration sent to SFP module: channel 1
```

Alternatively you can use `wavelength` to specify a value supported by your model. For example:

```
admin@EXALINK-FUSION> configure port a1 tx-tuning wavelength 1610
Port A1 tunable transmitter configuration sent to SFP module: wavelength 1610
```

Simulated Link Generation

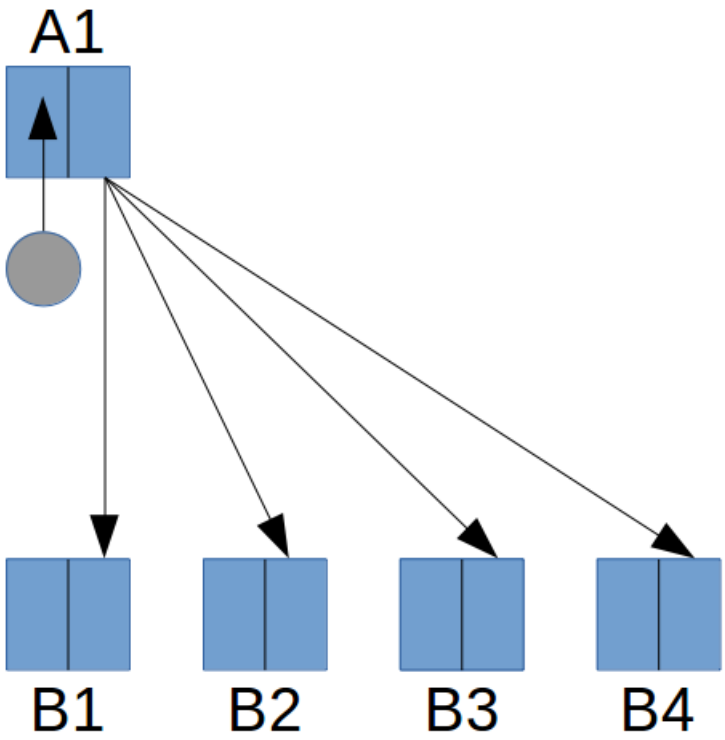
This feature requires version 1.10.0 or later.
Before version 1.12.0, this feature must be enabled manually using the `link-gen` command:

```
admin@EXALINK-FUSION(config)> port A1 link-gen
Simulated link generation on port A1 enabled
```

In version 1.12.0 and later, this feature is automatically enabled.

A common use for Layer 1 switches is for the distribution of market data. This can be done with patch or tap objects. If *only* `tap` objects are used, the link partner providing market data will not detect incoming link (i.e. report the switch interface as down), and as a result *may* not send traffic.

If a port does not have a source of data, the Fusion will generate and transmit link out a particular port, so that the remote end detects link and will bring the interface up. The image below shows a tap of data coming into port A1 to ports B1-B4, and the link generator on port A1 is enabled. This means the device connected to port A1 will detect the link as being up irrespective of what happens to the connections on ports B1-B4.



Link generation in use along with taps

Link generation will be enabled when:

- A SFP is plugged in to the port, and
- The port is **not** part of an object that outputs data to the port

Line Cards that have a Hardware Type of `LC10G-02` or `LC10G-03` do not support this feature. An FPGA module is not necessary, i.e. this feature is supported on Layer 1 only Fusions also.

Link generation is only supported at 1G and 10G.

Packet capture

A basic packet capture feature is enabled when the Fusion is fitted with the latest model Line Card. This allows the user to see packets traversing the high speed data plane on the CLI, which is very useful during commissioning, debugging etc.

Clearly it is not possible for the CLI to capture traffic flowing at line rate, so this intention of this feature is for packet capture when there is a small amount of traffic flowing on the wire.

In order to capture traffic flowing into a port, the `packet-dump` command should be used on the desired port as follows:

```
admin@EXALINK-FUSION> config port c1 packet-dump
22:10:51.779814 ARP, Request who-has 192.168.10.20 (ff:ff:ff:ff:ff:ff) tell 192.168
        0x0000:  ffff ffff ffff 643f 5f01 2700 0806 0001  ......d?_.'.....
        0x0010:  0800 0604 0001 643f 5f01 2700 c0a8 0a0a  ......d?_.'.....
        0x0020:  ffff ffff ffff c0a8 0a14 0000 0000 0000  ................
        0x0030:  0000 0000 0000 0000 0000 0000            ............
```

Alternatively, the packet can be piped into `tcpdump` to facilitate specific flags to be used. This is done by adding the `raw` parameter to the `packet-dump` command, then piping this into tcpdump. Note that tcpdump dump must be invoked with the `-r -` flags so it takes input from stdin, as follows:

```
admin@EXALINK-FUSION> port c1 packet-dump raw | tcpdump -r - -n -vvv -x
07:32:14.453680 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.23.1 t
        0x0000:  0001 0800 0604 0001 643f 5f01 2ea2 c0a8
        0x0010:  1716 0000 0000 0000 c0a8 1701 0000 0000
        0x0020:  0000 0000 0000 0000 0000 0000 0000
07:32:15.456001 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.23.1 t
        0x0000:  0001 0800 0604 0001 643f 5f01 2ea2 c0a8
        0x0010:  1716 0000 0000 0000 c0a8 1701 0000 0000
        0x0020:  0000 0000 0000 0000 0000 0000 0000
```

Note

Note that the port must be used as part of a config in order for the packet capture function to work.

Note

Line Cards that have a Hardware Type of `LC10G-02` or `LC10G-03` do not support this feature.

The output of `show version` can be used to determine the model of line card installed into the Fusion.

Patches and Taps

Patches and taps expose the ultra low latency layer 1 functionality of the ExaLINK Fusion. Ports that are members of patch or tap objects have a low and deterministic port to port latency of under 5ns with virtually no jitter.

Patches are useful for creating bidirectional connections between two devices connected to the ExaLINK Fusion. They can be used as an alternative to directly connecting two devices together, in cases where remote reconfiguration and patching is desired.

Taps are analogous to optical taps but with several advantages. Optical taps suffer from reduced output power and signal integrity issues as more tap outputs are introduced. Since the ExaLINK Fusion incorporates advanced clock and data recovery circuitry on all inputs along with electronic drivers on all outputs, it is possible to tap one input to a large number of outputs. For example, a single input can be replicated up to 47 times with no loss of signal integrity and with no latency penalty.

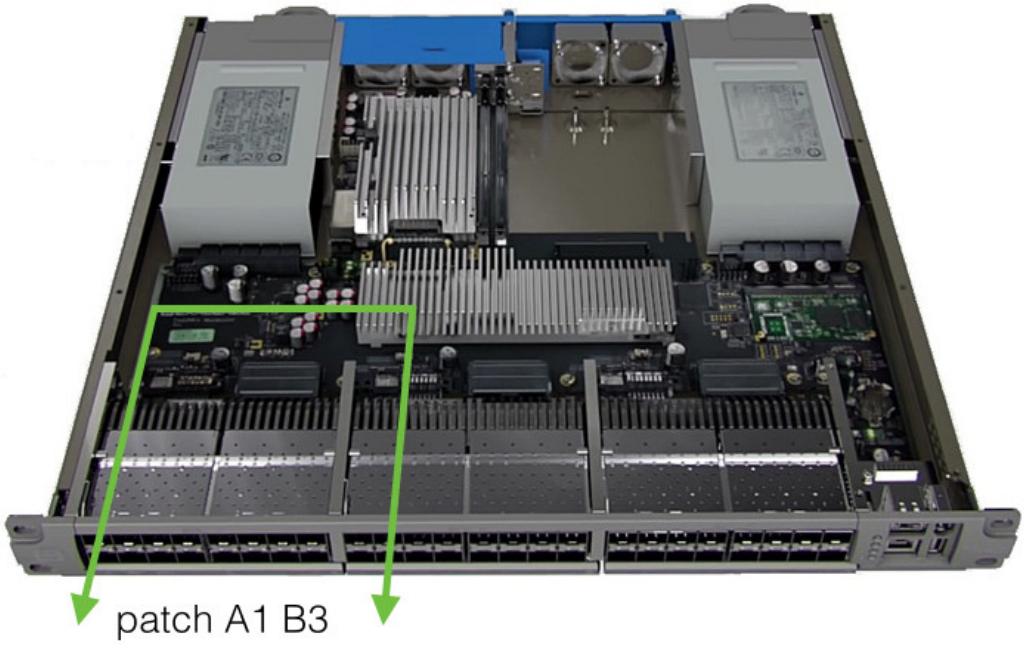
Note: that as patches and taps operate at layer 1 only, rate conversion between different speeds is not possible using these objects. Data will not flow successfully between ports of differing line rates using patch or tap objects.

Patching

Two ports can be connected together at the physical layer by creating a `patch` object. To create a patch object, first enter config mode, then use the `patch` command as shown:

```
admin@EXALINK-FUSION(config)> patch A1 B3
Patch created between port "A1" and port "B3"
```

A layer 1, physical connection has now been created between port A1 and B3. The port to port latency in this configuration will be under 5ns.



Patching port A1 to port B3

To remove an existing patch, use the `no` form of the patch command. For example, to remove the patch we created above:

```
admin@EXALINK-FUSION(config)> no patch A1 A2
Patch deleted between port "A1" and port "A2"
```

Note: that tab completion can be used on the port numbers. This is especially useful when specifying the second port in a patch to be removed, as the command line interface will resolve the second port automatically.

Setting up a tap

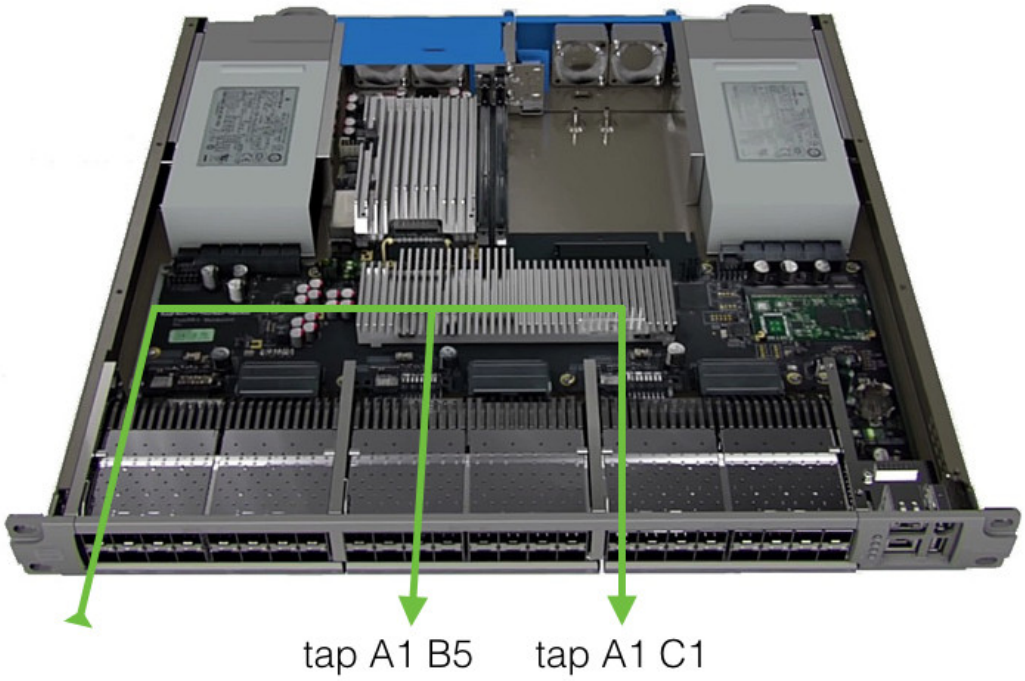
A tap object is analogous to an optical tap. Within the ExaLINK Fusion, tapping is performed electronically, which has a number of benefits when compared with optical taps whilst providing the same low and deterministic latency.

By default, tap objects replicate data received on a source port to an output port. This default is useful for a number of situations, including replication of market data to multiple hosts. To tap one port to another port, enter config mode and use the `tap` command to create a tap object:

```
admin@EXALINK-FUSION(config)> tap A1 A2
Added input tap on port "A1", send to port "A2"
```

An input can be tapped to any number of outputs. To create multiple taps with one command, use a range specifier. As an example, to replicate any traffic received on port A1 out A10, A11, and A12, use:

```
admin@EXALINK-FUSION> config tap A1 A10-A12
Added input tap on port "A1", send to port "A10"
Added input tap on port "A1", send to port "A11"
Added input tap on port "A1", send to port "A12"
```



Tapping one port to multiple outputs

Sometimes it is necessary to tap the data that is transmitted out of a given port, as opposed to data received on that port. One potential application is to replicate data sent to an exchange out of a logging port. To configure an output tap, simply add the `output` modifier to the tap command:

```
admin@EXALINK-FUSION(config)> tap output A1 A2
Added output tap on port "A1", send to port "A2"
```

To remove any tap use the `no` form of the tap command:

```
admin@EXALINK-FUSION(config)> no tap A1 A2
Removed input tap on port "A1", send to port "A2"
```

Similarly, for an output tap:

```
admin@EXALINK-FUSION(config)> no tap output A1 A2
Removed output tap on port "A1", send to port "A2"
```

FPGA module

The ExaLINK Fusion uses reconfigurable FPGA technology to provide Layer 2 functionality. Different FPGA firmwares can be loaded onto each of the internal modules in order to modify the functionality. To change the firmware function, use the `configure module X function` command. For example, to configure the mux firmware onto module X (the default module installed at the factory):

```
admin@EXALINK-FUSION> configure module X function mux
Module X function set to mux
WARNING: Module X is initializing
```

On the ExaLINK Fusion, the available firmware types are `fastmux`, `mux`, `switch` and `custom`.

On the ExaLINK Fusion HPT, the available firmware types are `hpt` and `hpt-40g`.

The mux firmware is optimized for use with mux objects and mirror objects but does not support switch objects. The switch firmware provides full layer 2 switching functionality and supports all object types. The fastmux firmware uses advanced features to provide the lowest latency. The custom function should be set by users who are running their own firmware on one of the FPGA modules - please refer to the FPGA Development section for more information.

The `show module` command can be used to see information as to the current state and config of a module:

```
admin@EXALINK-FUSION> show module X
Module     : X
Function   : mux
Power      : on
State      : running
FPGA state : configured
```

Please see this List of FPGA Firmware Differences for further details of the differences between the `fastmux`, `mux` and `switch` firmware types.

Switch objects

Switch overview

The switch object provides the functionality of a normal layer 2 switch. Ports that are members of the switch object can communicate with any other member ports. The switch implements MAC address learning to route traffic based on the destination MAC address.

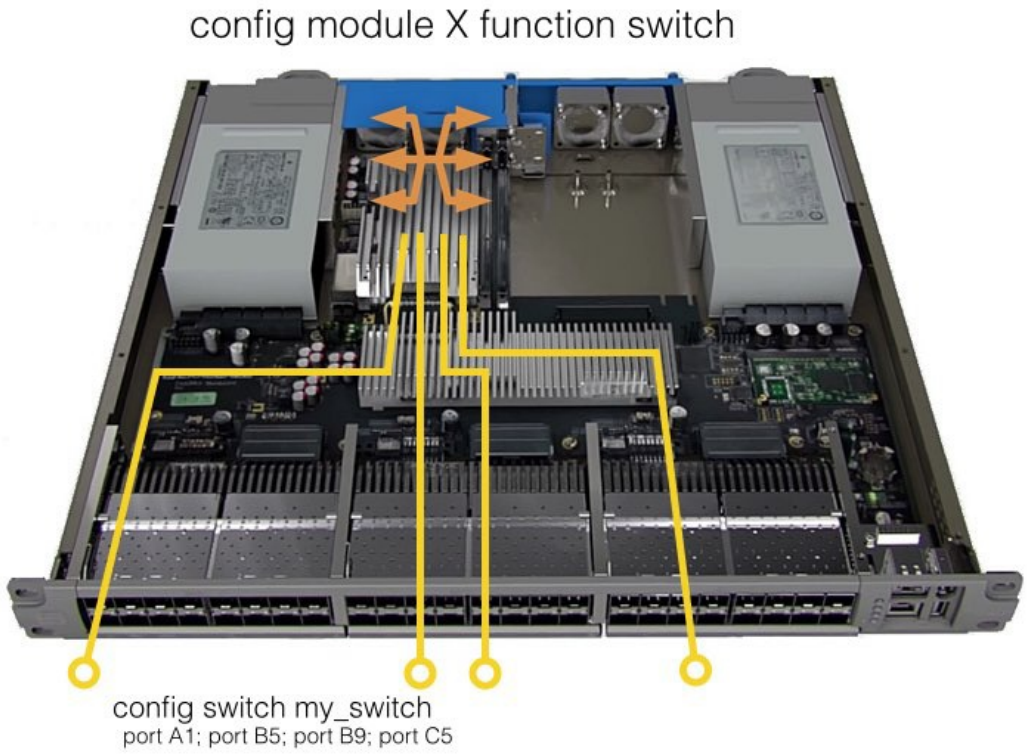
Multiple, independent switch objects can be created on the ExaLINK Fusion. Each switch object is logically an independent layer 2 switch with its own broadcast domain. There is no restriction on the number of switch objects that can be created.

Warning

The switch object requires the `switch` firmware to be loaded on the FPGA module.
Refer to this section for details on selecting firmware.

Note

Your Fusion must have a valid license in order to create `switch` objects. Refer to Licensing for further details.



Data is routed through the switch based upon the destination MAC address. For details on managing the MAC address table, see the MAC address table section.

Configuring a switch object

The switch configuration is managed via an instance of the switch object. To create a switch object use the `switch` command, followed by a name for the instance:

```
admin@EXALINK-FUSION> config switch myswitch
Switch name "myswitch" created
```

After creating the switch object it is possible to add any number of ports. To add a front panel port to the switch, use the `port` command:

```
admin@EXALINK-FUSION> config switch myswitch port A1
Added port "A1" to switch "myswitch"
```

Use the `no` form of the port command to remove a port from the switch:

```
admin@EXALINK-FUSION> config switch myswitch no port A1
Removed port "A1" from switch "myswitch"
```

When no longer in use, a switch object can be removed using the `no` form of the switch command:

```
admin@EXALINK-FUSION> config no switch myswitch
Switch "myswitch" deleted
```

Unknown unicast flooding

By default, unicast frames with an unknown destination MAC address will be broadcast to all ports in the switch object.

The switch object can be configured to block unknown unicast traffic from being transmitted on selected ports by using the `no unknown-unicast` command.

```
admin@EXALINK-FUSION(config-switch:myswitch)> no unknown-unicast B2
Port "B2" of switch "myswitch" drops unknown unicast frames
```

Note This setting will be reset to default if a port is removed from an object.

Blocking traffic between ports

This feature requires version 1.9.0 or later

By default all ports can communicate with all other ports in the same switch object. However, using the `block` command, the switch object can be configured to block all traffic between selected ports:

```
admin@EXALINK-FUSION(config-switch:myswitch)> block A1 B2
Blocking traffic from port "A1" to port "B2" on switch "myswitch"
```

Blocks are unidirectional, so the above example will block all traffic from port A1 to port B2, but allow traffic to flow from port B2 to port A1. However, because unicast requires ARP in both directions, in practice this would only allow multicast and broadcast traffic to flow from port B2 to port A1.

Mux objects

Mux overview

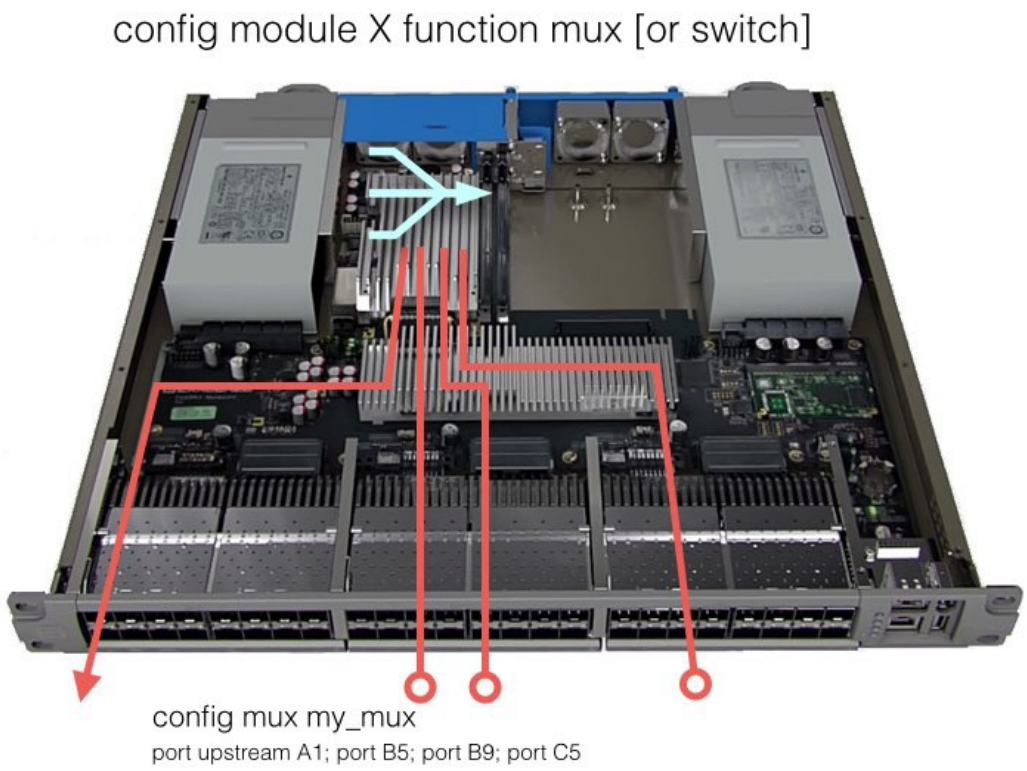
The mux object can be thought of as a specialized switch which is optimized for many-to-one connectivity. The advantage of using a mux over a switch is that no packet inspection or routing decisions are required. This results in significant latency reductions.

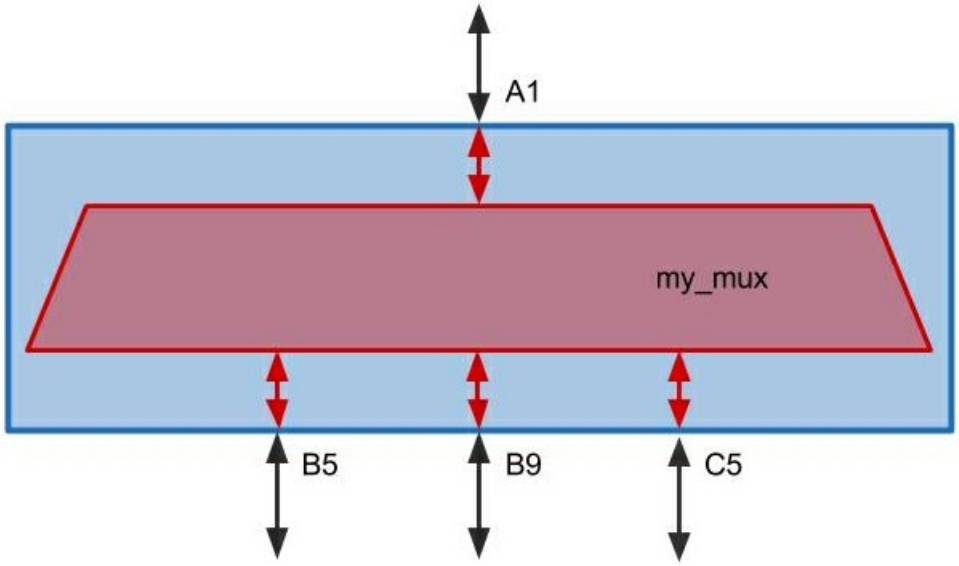
A mux object has many *downstream* ports which are connected to one *upstream* port. The *upstream* port is typically connected to a shared connection - for example, an order line to an exchange. The *downstream* ports are usually connected to devices that need to share a connection, for example local trading servers.

The supported number of mux objects, upstream ports, and downstream ports varies depending on the currently running firmware mode (see Multiple mux objects for more details). The `mux` object requires the `switch`, `mux`, or `fastmux` firmware to be loaded and running.

Note

A switch license is *not* required to create `mux` objects in the `switch` firmware mode. A license is only required to create `switch` objects. Refer to Licensing for further details.





A typical Mux setup, showing a single upstream WAN connection (A1) to be shared amongst three local downstream servers (B5, B9, C5)

Note that traffic cannot be sent between downstream ports within a mux object. In the example depicted above B5 is not able to communicate with B9 or C5.

The primary function of a mux is to forward traffic from downstream ports up to the upstream port. However, most use-cases require a bidirectional connection, which means that traffic must be returned from the upstream to the downstream ports. Mux objects can be configured in several different modes that affect how return traffic is routed from the upstream port to the downstream ports.

- In `layer2` mode, traffic that arrives at the upstream port is routed to a downstream port according to the destination MAC address. The upstream port to downstream port latency in this configuration is approximately 107ns. In this mode downstream hosts will only see traffic destined for their MAC address, as well as broadcast traffic.
- In `raw` mode, traffic that arrives at the upstream port is broadcast via the layer 1 cross-point device to the downstream ports. This allows return traffic to be routed to hosts in around 5 ns. However, all downstream hosts will see all return data from the upstream connection. Usually the network card on the downstream host will perform this filtering transparently, so `raw` mode can provide significant latency improvements. One consideration in the use of `raw` mode is network security. In shared setups, where multiple independent parties are sharing an ExaLINK Fusion, it may be undesirable for all local hosts to have visibility into all return data from the upstream port. In such setups, `layer2` mode should be used. Note: when using a `mux` object with `fastmux` firmware, the `layer2` forwarding is not supported.
- The `fast-vlan` mode can be used on VLAN enabled mux objects to slightly improve the latency by not inspecting the VLAN tag of packets entering a downstream port. Refer to VLAN support for more details.

Note that when raw mode is used, all ports within the mux object operate at the same link speed (e.g. 10Gb/s). This is because all there is no opportunity to do rate conversion at layer 1. The CLI will issue warnings if ports of different speeds are added to a mux object in raw mode.

Configuring a mux object

A mux object must first be created and assigned a name. This must be done from within the configuration mode, or by prefixing the command with `config`.

```
admin@EXALINK-FUSION(config)> mux my_mux
Mux "my_mux" created
```

A common next step is to add front panel ports to the mux. Any front panel port can be assigned as the upstream port, but currently only one upstream port can be assigned per mux. As an example, to assign port A1 as an upstream port:

```
admin@EXALINK-FUSION(config-mux:my_mux)> port up A1
Added upstream port "A1" to mux "my_mux"
```

Any number of downstream ports can then be assigned to the mux. To add a downstream port simply omit the 'up' modifier to the port command:

```
admin@EXALINK-FUSION(config-mux:my_mux)> port A2
Added downstream port "A2" to mux "my_mux"
```

Both upstream and downstream ports can be removed from the mux using the `no` form of the port command. It is recommended to remove unused ports from a mux when they are no longer required.

```
admin@EXALINK-FUSION(config-mux:my_mux)> no port A2
Removed port "A2" from mux "my_mux"
```

The mode of the mux can be changed between `layer2` (the default setting) and `raw` using the mode command:

```
admin@EXALINK-FUSION(config-mux:my_mux)> mode raw
Set mux "my_mux" to raw mode
```

The mode can be changed back to `layer2` using:

```
admin@EXALINK-FUSION(config-mux:my_mux)> mode layer2
Set mux "my_mux" to layer2 mode
```

Note A `mux` object using `fastmux` firmware can only operate in `raw` mode.

To display the current configuration of a mux object use the `show` command:

```
admin@EXALINK-FUSION(config-mux:my_mux)> show
Mux name      : my_mux
Mode          : layer2
VLAN tagging  : disabled
IGMP snooping : disabled

Port Side
---- ----------
A1   upstream
A2   downstream
```

The mux object should be deleted when no longer in use. This is done using the `no` form of the mux command:

```
admin@EXALINK-FUSION(config)> no mux my_mux
Mux "my_mux" deleted
```

Multiple mux objects

Mux objects can be created when running any of the three standard firmware images (`switch` , `mux` and `fastmux`). The supported number of mux objects, upstream ports, and downstream ports varies depending on the firmware mode. These differences are summarized in the table below:

Firmware	Mux Objects	Upstream Ports	Downstream Ports
Switch	Unlimited	Unlimited	Unlimited
Mux	Unlimited	4	Unlimited
FastMux	4	4	15, 11, 11, 7

Switch firmware

When running `switch` firmware, there is no restriction on the number of mux objects that can be created. The number of downstream ports for each mux object is also unlimited. For example, a single mux with 47 downstream ports can be created (48 ports total). Similarly, one can create 6 muxes each with 7 downstream ports (48 ports total).

Mux firmware

The `mux` firmware also supports any number of mux objects, however, at most 4 individual upstream ports are supported. Multiple mux objects can use the same upstream port if the VLAN ID of the upstream port is different. Please refer to VLAN support for further details. As with the `switch` firmware, all mux objects in `mux` firmware support an unlimited number of downstream ports. For example, a single mux with 47 downstream ports can be created (48 ports total). Similarly one could create 3 muxes each with 15 downstream ports (48 ports total). However, 6 muxes with 6 different upstream ports is not supported in the `mux` firmware mode.

Fastmux firmware

In the `fastmux` firmware mode the underlying FPGA firmware has 4 different fixed sized mux resources in it. This means that the `fastmux` firmware can create up to 4 mux objects each with its own upstream port. The number of downstream ports supported by each mux object depends on the hardware resource allocated to it. It may be:

- 15 ports to 1
- 11 ports to 1
- 11 ports to 1
- 7 ports to 1

The management processor will attempt to automatically map mux objects to one of the 4 muxes within the FPGA, depending on the number of ports in the configuration. For example, if a 14:1 mux and a 5:1 mux are created, the software will map the 14:1 mux to the 15:1 FPGA mux resource and the 5:1 mux to the 7:1 FPGA mux resource.

The allocation logic will wait until an upstream port is specified before making a decision on which FPGA mux to use.

If additional ports are added to a mux object, causing it to exceed the number of ports that the underlying hardware can support, the following error will be displayed: `WARNING: Failed to allocate switch resources for port C3`. In this case, `config optimize` needs to be run to reallocate resources within the system. If any mux object has more than 15 downstream ports, this command will not succeed.

Warning

Running the `optimize` command may cause traffic flows to be be momentarily interrupted.

As an example, assume that a single 5:1 mux is required and there are no other mux objects defined. In this case, the following steps would be taken:

```
admin@EXALINK-FUSION> config mux my_mux
Created mux "my_mux"
WARNING: Mux object "my_mux" disabled: Mux objects must be set to raw mode when usi
admin@EXALINK-FUSION(config-mux:my_mux)> mode raw
Set mux "my_mux" to raw mode
admin@EXALINK-FUSION(config-mux:my_mux)> port a1-a5
Added downstream port "A1" to mux "my_mux"
Added downstream port "A2" to mux "my_mux"
Added downstream port "A3" to mux "my_mux"
Added downstream port "A4" to mux "my_mux"
Added downstream port "A5" to mux "my_mux"
admin@EXALINK-FUSION(config-mux:my_mux)> port up b1
Added upstream port "B1" to mux "my_mux"
```

This mux has 5 downstream ports so the allocation logic will map this mux to the 7:1 hardware mux resource within the FPGA.

If, at a later stage, an additional 3 ports needs to be added:

```
admin@EXALINK-FUSION(config-mux:my_mux)> port c1-c3
Added downstream port "C1" to mux "my_mux"
Added downstream port "C2" to mux "my_mux"
Added downstream port "C3" to mux "my_mux"
WARNING: Failed to allocate switch resources for port C3
WARNING: The "optimize" command may help to resolve allocation failures
admin@EXALINK-FUSION(config-mux:my_mux)>
```

When the 8th port is added (ie C3), an *"allocation"* error results. The `optimize` command can then be issued to instruct the software to swap ports and objects around to fit the requested config.

```
admin@EXALINK-FUSION(config-mux:my_mux)> exit
WARNING: Failed to allocate switch resources for port C3
WARNING: The "optimize" command may help to resolve allocation failures
admin@EXALINK-FUSION(config)> optimize
This may cause an interruption in network traffic
Are you sure? y
Optimizing switch resources...
Operation completed
admin@EXALINK-FUSION(config)>
admin@EXALINK-FUSION(config)>
```

Note

`Fastmux` mode has traded off some management features for latency, such as LLDP, BGP and IGMP.

MAC address table

The switch and mux of the ExaLINK Fusion makes use of a MAC address table. The table is built dynamically from the source address of received frames, and can also be populated with static entries by the user.

When a frame is received on a port in a switch object, or on the upstream port of a mux object, the destination address is looked up in the MAC address table, and the frame is sent out on the ports stored in the table. If the address is not in the table, the frame is broadcast on all other ports of the switch object, or on all downstream ports of the mux object.

Frames received on downstream ports of a mux object bypass the MAC address table and are always sent out the upstream port.

Display the MAC address table

The `show mac-address-table` command can be used to show the contents of the MAC address table. For example:

```
admin@EXALINK-FUSION(switch:my_switch)> show mac-address-table
Address      Input                                          Output Type
------------ ---------------------------------------------- ------ ----
643F5F0114A1 A13 A14 B1 B2 B5 B6 B9 B10 B13 B14 C1 C2 C5 C6  A10    L
643F5F0114A2 A10 A14 B1 B2 B5 B6 B9 B10 B13 B14 C1 C2 C5 C6  A13    L
643F5F0114A3 A10 A13 B1 B2 B5 B6 B9 B10 B13 B14 C1 C2 C5 C6  A14    L
643F5F01159C A10 A13 A14 B2 B5 B6 B9 B10 B13 B14 C1 C2 C5 C6 B1     L
643F5F01159D A10 A13 A14 B1 B5 B6 B9 B10 B13 B14 C1 C2 C5 C6 B2     L
643F5F01159E A10 A13 A14 B1 B2 B6 B9 B10 B13 B14 C1 C2 C5 C6 B5     L
643F5F01159F A10 A13 A14 B1 B2 B5 B9 B10 B13 B14 C1 C2 C5 C6 B6     L
```

- The `address` field contains the MAC address for this entry
- The `input` field indicates the input ports for which this entry applies.
- The `output` field indicates the ports on which a matching frame will be output.
- The `type` field indicates whether the entry is learned (L) from received frames or if it is a static (S) entry added by the user.

MAC address learning

By default a switch or mux object will learn routes via MAC source addresses from incoming packets and note the ports they come from. This route learning behaviour can be disabled via:

```
admin@EXALINK-FUSION(switch:my_switch)> no mac-address-learning
```

If required, can be enabled again via:

```
admin@EXALINK-FUSION(switch:my_switch)> mac-address-learning
```

Clearing the MAC address table

The MAC address table for an object can have all learned/dynamic entries cleared using the following command:

```
admin@EXALINK-FUSION(switch:my_switch)> clear mac-address-table dynamic
Cleared dynamic MAC address table entries on switch "my_switch"
```

Note this will not impact any static entries that have been added.

Managing static routes

The `mac-address-table static` command can be used to add static entries to the MAC address table. For example, suppose we want to configure a mux object to send traffic for the MAC address 64:3F:5F:01:23:45 to port A2:

```
admin@EXALINK-FUSION(config-mux:my_mux)> mac-address-table static 643F5F012345 port
Added static MAC address "643F5F012345" port "A2" on mux "my_mux"
```

To remove the static entry, use the `no` form of the command:

```
admin@EXALINK-FUSION(config-mux:my_mux)> no mac-address-table static 643F5F012345 p
Removed static MAC address "643F5F012345" port "A2" on mux "my_mux"
```

Note that all packets received on a downstream port of a mux object are transmitted out the upstream port. Static entries in the MAC address table for mux objects only apply to packets received on an upstream port.

Disabling MAC address learning

The `no mac-address-learning` command disables MAC address learning for a switch or mux object:

```
admin@EXALINK-FUSION(config-mux:my_mux)> no mac-address-learning
Disabled MAC address learning on mux "my_mux"
```

To re-enable MAC address learning, use the `mac-address-learning` command:

```
admin@EXALINK-FUSION(config-mux:my_mux)> mac-address-learning
Enabled MAC address learning on mux "my_mux"
```

IGMP and multicast support

The ExaLINK Fusion includes support for IGMP snooping to optimize delivery of IPv4 multicast traffic.

Enabling IGMP snooping

IGMP snooping is available on a per switch/mux object basis. IGMP snooping is disabled by default, but can be enabled using the `igmp snooping` command within the switch or mux configuration mode:

```
admin@EXALINK-FUSION(config-switch:my_switch)> igmp snooping
Enabled IGMP snooping on switch "my_switch"
```

You may review your IGMP snooping settings via 'show igmp snooping', note that by default IGMP snooping is disabled (an error will display to reflect this):

```
admin@EXALINK-FUSION(config-switch:my_switch)> show igmp snooping
IGMP snooping           : disabled
IGMP querier            : disabled
Multicast router ports  :
Flood unknown multicast : disabled
Fast leave              : disabled

Error: IGMP snooping not enabled
```

Once enabled, it can be disabled using the `no` form of the command:

```
admin@EXALINK-FUSION(config-switch:my_switch)> no igmp snooping
Disabled IGMP snooping on switch "my_switch"
```

IGMP groups

Groups that object ports are subscribed to can be reviewed at any time via the 'show igmp groups' command:

```
admin@EXALINK-FUSION(config-switch:my_switch)> show igmp groups
Group      Ports Version Timeout
---------- ----- ------- -------
224.2.0.52 B2    v2      259
224.2.0.55 C9    v3      63
```

IGMP querier

The ExaLINK Fusion can also be configured as an IGMP querier to periodically send IGMP queries originating from a particular IP address. Use the `igmp snooping querier` command to enable IGMP querier functionality:

```
admin@EXALINK-FUSION> configure switch my_switch igmp snooping querier 192.168.1.1
Configured IGMP querier on switch "my_switch" at address 192.168.1.1
```

The ExaLINK Fusion will now periodically send IGMP queries from the designated address.

To disable the periodic sending of IGMP queries, use the `no` form of the command:

```
admin@EXALINK-FUSION> configure switch my_switch no igmp snooping querier
Disabled IGMP querier on switch "my_switch"
```

Note: that the IGMP querier is enabled only if IGMP snooping is enabled.

IGMP Version

By default the ExaLINK Fusion supports IGMPv3, however it can be configured to use IGMPv1 or IGMPv2. Use the `igmp snooping version` command to set this:

```
admin@EXALINK-FUSION> configure switch my_switch igmp snooping version 2
Set IGMP version 2 on switch "my_switch"
```

Multicast router ports

For IGMP snooping to function correctly, the switch needs to know which ports are connected to multicast routers. A port can be configured to be a multicast router port by using the `igmp mrouter` command:

```
admin@EXALINK-FUSION(config-switch:my_switch)> igmp mrouter A1
Configured port "A1" on switch "my_switch" as multicast router port
```

Multicast router ports can be removed by using the `no` form of the command:

```
admin@EXALINK-FUSION(config-switch:my_switch)> no igmp mrouter A1
Removed port "A1" on switch "my_switch" as multicast router port
```

Unknown multicast groups

By default, all unknown multicast traffic will be suppressed on a switch or mux object. This behaviour can be changed using the `igmp flood-unknown` command.

```
admin@EXALINK-FUSION(config-switch:my_switch)> igmp flood-unknown
Flooding unknown multicast traffic on switch "my_switch"
```

When this setting is enabled, traffic for a particular multicast group will be flooded before a host joins the group. Once a host joins the group, the group is no longer unknown and traffic will no longer be flooded for that group.

Use the `no` form of the `igmp flood-unknown` command to disable it:

```
admin@EXALINK-FUSION(config-switch:my_switch)> no igmp flood-unknown
Suppressing unknown multicast traffic on switch "my_switch"
```

IGMP fast leave

IGMP snooping on the ExaLINK Fusion supports "fast leave" functionality to reduce the time it takes for a host to leave a multicast group. When fast leave is enabled, a port is immediately removed from a multicast group when a IGMPv2 Leave message is received on the port.

Warning

At most one host can be connected to each non-router port when fast leave is enabled.

Ports with more than one host connected may be incorrectly dropped from multicast groups.

Use the command `igmp snooping fast-leave` to enable fast leave mode, and the `no` form of this command to disable fast leave mode.

```
admin@EXALINK-FUSION(config-switch:my_switch)> igmp snooping fast-leave
Enabling IGMP fast leave mode on switch "my_switch"

admin@EXALINK-FUSION(config-switch:my_switch)> no igmp snooping fast-leave
Disabling IGMP fast leave mode on switch "my_switch"
```

Manually setting multicast filters

It is possible to manually add multicast filters instead of/as well as using IGMP snooping:

```
admin@EXALINK-FUSION(config-switch:my_switch)> igmp static-group 225.1.1.1 port B1
Added static multicast group "225.1.1.1" for port "B1" on switch "my_switch"
```

VLAN support

The ExaLINK Fusion supports traffic separation using VLAN tags, and adding, removing or rewriting VLAN tags of packets passing through the device.

Concepts

The concept of switch objects on the ExaLINK Fusion already allows network segmentation without using VLANs.

The VLAN feature on the ExaLINK Fusion allows VLAN tagging ports to be added to switch or mux objects. A VLAN tagging port can be shared between more than one switch or mux object, so that the port can be used as a trunking port.

VLAN tag rewriting can be achieved by adding VLAN tagging ports with different VLAN IDs to the same object.

The Fusion supports up to 256 different VLANs, and the VLAN ID can be up to 4093.

Enabling VLAN support

VLAN support is enabled or disabled per switch and mux object. To enable VLAN support on an object, use the following command:

```
admin@EXALINK-FUSION(config-switch:my_switch)> vlan-enable
Enabled VLAN support on switch "my_switch"
```

When VLAN support is enabled on an object, ports already in the object will become untagged ports, and will only accept untagged packets.

VLAN support can be disabled on an object using the `no` form of the command:

```
admin@EXALINK-FUSION(config-switch:my_switch)> no vlan-enable
Disabled VLAN support on switch "my_switch"
```

Adding ports to a VLAN enabled object

The following `port` command can be used to add an untagged port to a VLAN enabled object:

```
admin@EXALINK-FUSION(config-switch:my_switch)> port A1
Added port "A1" to switch "my_switch"
```

This variant of the `port` command can be used to add a VLAN tagged port to a VLAN enabled object:

```
admin@EXALINK-FUSION(config-switch:my_switch)> port A2 vlan 10
Added port "A2" with VLAN ID 10 to switch "my_switch"
```

Note: A port can only be added to an object once. This means that the same port can not be both tagged and untagged in an object, or be added with two different VLAN IDs.

The `show` command can be used to see the ports added to the object, and the VLAN IDs used for each port:

```
admin@EXALINK-FUSION(config-switch:my_switch)> show
Switch name   : my_switch
VLAN tagging  : enabled
IGMP snooping : disabled

Port VLAN ID
---- --------
A1   untagged
A2   10
```

Mux VLAN modes

The mux object supports two distinct VLAN modes, described below. If a mux object has VLAN tagging enabled, the use of `raw` mode is not compatible.

- `fast-vlan`: A packet that arrives at a downstream port in the mux will be forwarded to the upstream port of that mux, irrespective of whether the packet has a VLAN tag, the VLAN ID, or if the packet is untagged. When this packet is transmitted out of the associated upstream port, it will have the tag associated with that upstream port (or will be untagged, if the upstream port is untagged). Since a lookup is not required, this mode only incurs the added latency of inserting, removing, or modifying a tag. The latency from downstream to upstream port in this mode is approximately 107ns.
- `layer2`: The VLAN tag of a packet that arrives at a downstream port in the mux will be inspected. The upstream port to forward the packet to depends on the VLAN tag (or lack thereof) in the packet. The packet is transmitted out of the associated upstream port with the tag associated with the upstream port. This mode can be used to select between multiple upstream ports, or translate VLAN IDs between the downstream and upstream ports. The latency from downstream to upstream port in this mode is approximately 125ns.

Sharing physical ports between objects

Ports can be shared between multiple objects provided that the VLAN ID of that port is different for each object. Switch ports and mux upstream ports have no further restrictions on use.

Physical mux downstream ports can be shared between multiple mux objects, provided that the mux is configured in `layer2` mode. Downstream ports that are members of a mux configured in `fast-vlan` mode can only be used in one object.

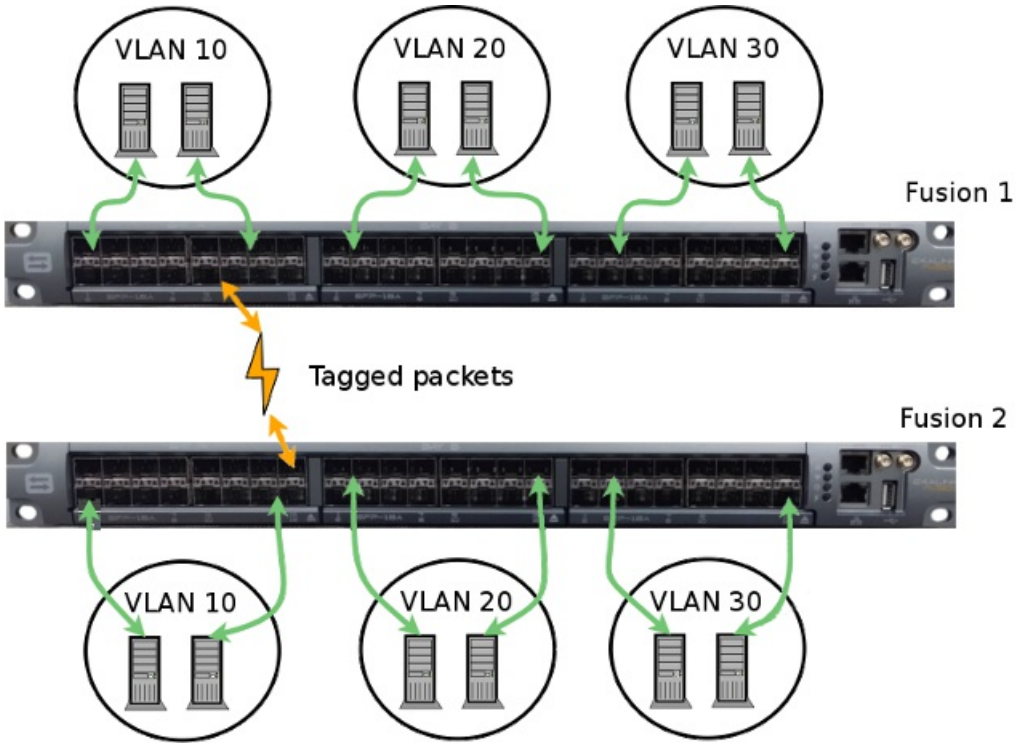
Example configurations

Fast trunking of a common WAN connection

In this example, three clients share a common WAN connection, with traffic forcibly separated using VLAN tags. Segregation is done in fast-vlan mode, which means that when forwarding a packet from downstream to upstream ports, the VLAN tag of the frame is not inspected. This means that, for example, if a client in mux m1 (VLAN 10) sends an untagged packet into a downstream port, it will leave the upstream port tagged with VLAN ID 10. Also, if a client in mux m1 sends a tagged packet into a downstream port, that tag will be translated to VLAN ID 10 prior to leaving the upstream port.

When a packet arrives at the upstream ports in this configuration, forwarding is conducted based upon VLAN ID. For example:

- When a packet arrives at port A10 with tag 10, it will only be forwarded to downstream ports in mux m1. When this packet leaves these downstream ports it will be untagged.
- When a packet arrives at port A10 with no tag it will be dropped. This is because no mux objects include port A1 as an untagged upstream port.



Fusion 1 configuration:

```
mux m1
  vlan-enable
  mode fast-vlan
  port up A10 vlan 10
  port A1
  port A11

mux m2
  vlan-enable
  mode fast-vlan
  port up A10 vlan 20
  port B1
  port B13

mux m3
  vlan-enable
  mode fast-vlan
  port up A10 vlan 30
  port C1
  port C15
```

Fusion 2 has the same configuration, with the downstream ports replaced appropriately.

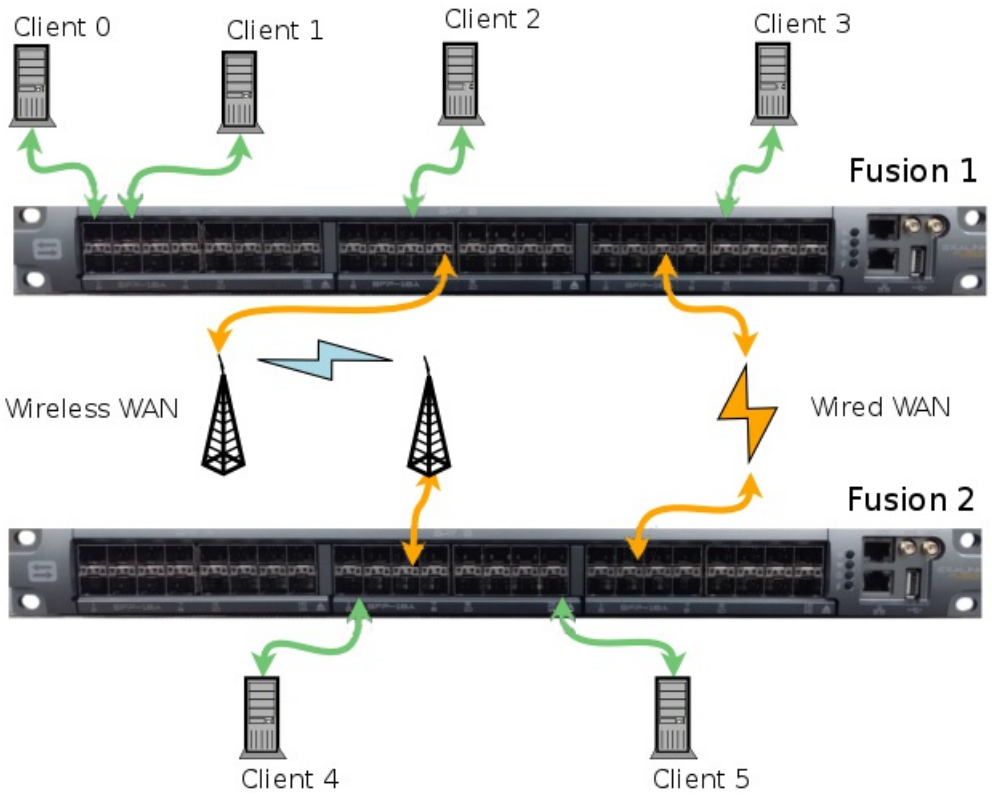
In this configuration the latencies at 10GbE are as follows:

- Downstream to upstream: 107 ns
- Upstream to downstream: 125 ns

Sharing of upstream ports with tag translation

In this example two upstream WAN ports are shared between multiple clients, with translation of tags between downstream and upstream ports. Clients can choose to send packets over the Wireless or Wired WAN link based upon VLAN ID. Under this config:

- We define two groups of clients at each end, group 1 and group 2.
- We segregate these clients so that whichever WAN link they use, we prevent group 1 from communicating with group 2.
- When a client transmits an untagged packet, it will be transmitted with a VLAN tag over the wireless link. The VLAN ID is the group number of the client.
- When a client transmits a packet tagged with VLAN ID 30, it will be transmitted over the wired WAN. The VLAN ID will be translated to the group ID of the client.
- Any other VLAN IDs used by the client will cause the packets to be dropped (not forwarded).
- When the wireless WAN upstream port receives a packet, it will forward it to group 1 or group 2 clients based on the VLAN tag. These packets will be transmitted out of the downstream port with no tag.
- When the wired WAN upstream port receives a packet, it will forward it to group 1 or group 2 clients based on the VLAN tag. These packets will be transmitted out of the downstream port with tag 30.



Fusion 1 configuration:

```
mux wireless_group_1
  vlan-enable
  mode layer2
  port up B4 vlan 1
  port A1
  port A3

mux wired_group_1
  vlan-enable
  mode layer2
  port up B16 vlan 1
  port A1 vlan 30
  port A3 vlan 30

mux wireless_group_2
  vlan-enable
  mode layer2
  port up B4 vlan 2
  port B1
  port C1

mux wired_group_2
  vlan-enable
  mode layer2
  port up B16 vlan 2
  port B1 vlan 30
  port C1 vlan 30
```

Fusion 2 configuration:

```
mux wireless_group_1
  vlan-enable
  mode layer2
  port up B1 vlan 1
  port A16

mux wired_group_1
  vlan-enable
  mode layer2
  port up B13 vlan 1
  port A16 vlan 30

mux wireless_group_2
  vlan-enable
  mode layer2
  port up B1 vlan 2
  port B10

mux wired_group_2
  vlan-enable
  mode layer2
  port up B13 vlan 2
  port B10 vlan 30
```

In this configuration the latencies at 10GbE are as follows:

- Downstream to upstream: 125 ns
- Upstream to downstream: 125 ns

Switch trunking ports

This configuration sets up port A1 as a trunking port for two switch objects:

```
switch my_switch_1
  vlan-enable
  port A1 vlan 10
  port A2
  port A3

switch my_switch_2
  vlan-enable
  port A1 vlan 20
  port A4
  port A5
```

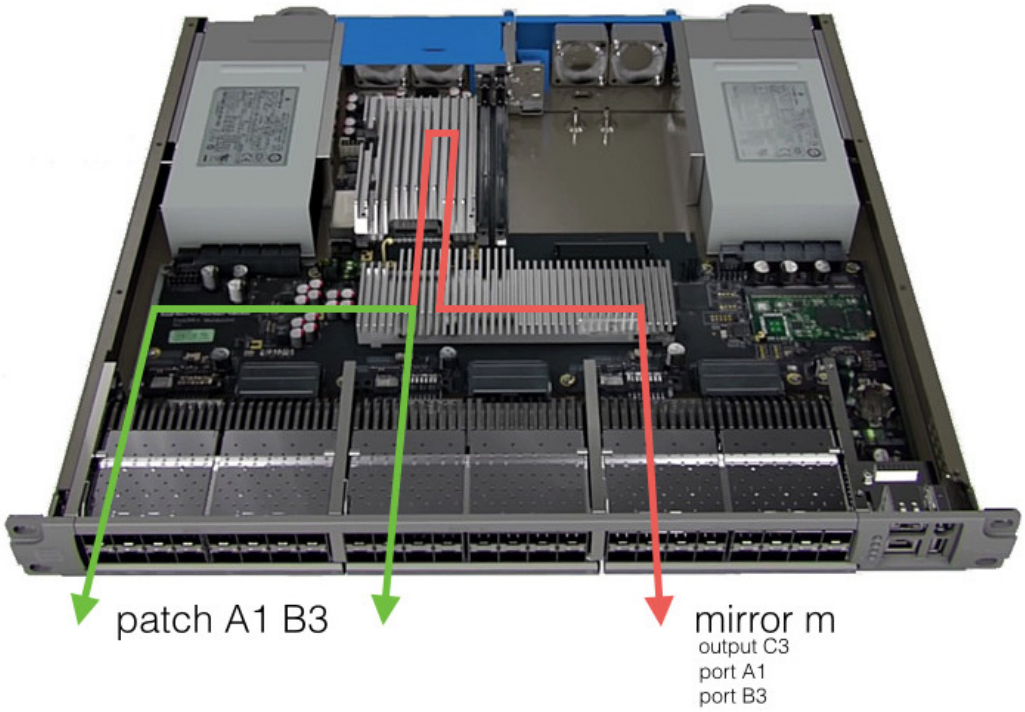
Mirror and timestamping - ExaLINK Fusion

Note

This page describes the operation of mirror objects for the **ExaLINK Fusion**. For details on the operation of mirror objects for the ExaLINK Fusion HPT, please refer to this page.

The mirror configuration provides a mechanism for configuring the logging and timestamping functionality within the ExaLINK Fusion. A mirror replicates any data traversing its member objects out of another port. Ports, mux and switch objects can be added to a mirror to replicate traffic that traverses them.

The mirror can also be configured to enable timestamping. All packets that enter the ExaLINK Fusion are timestamped on arrival at the ingress port. This timestamp is then inserted into the packet body when packets arrive at the mirror egress port.



Example use of a mirror, showing port A1 and port B3 mirrored to output port C3.

Creating a mirror

Port mirrors can be created after specifying a name, completed in either configuration mode, or prefixing the command with the `config` keyword.

```
admin@EXALINK-FUSION(config)> mirror mymirror
Mirror "mymirror" created
```

Creating a mirror will place the command line into a config-mirror state. Within this state an output

port for the mirror can be assigned using the `output` command. The output port can be any of the available front panel ports.

```
admin@EXALINK-FUSION(config-mirror:mymirror)> output C3
Added output port "C3" to mirror "mymirror"
```

Note: It is not possible to use a port that is already in use for the output port.

Note: There are only four timestamping/mux outputs available in the FPGA. These can be assigned arbitrarily to front-panel ports. Adding an output port to a mirror will consume one of these outputs. If you need a lot of outputs from a mirror, it's worth considering a mirror with only one output port, and using a tap object from that output port to the other ports.

The mirror will replicate the traffic of any of its member objects. To assign an object to the mirror use the object type followed by the name. Objects that can be mirrored are `port`, `mux` and `switch` objects. For example, to replicate all traffic received on ports A1 and B3:

```
admin@EXALINK-FUSION(config-mirror:mymirror)> port A1; port B3
Added port "A1" to mirror "mymirror"
Added port "B3" to mirror "mymirror"
```

In addition to replicating specific ports, it is also possible to replicate all traffic received by a given mux or switch object. For example, to replicate all of the traffic passing through an existing mux object, use the `mux` command:

```
admin@EXALINK-FUSION(config-mirror:mymirror)> mux mymux
Added mux "mymux" to mirror "mymirror"
```

Note: It is not possible to explicitly mirror outputs to a port using a mirror object, only inputs to ports or mux/switch objects. In order to take a copy of all traffic leaving a port, it is suggested to use a tap object and tap the output port.

Note: Mirroring is not supported when using `fastmux` firmware.

To remove an object from the mirror, use the `no` form of the commands:

```
admin@EXALINK-FUSION(config-mirror:mymirror)> no mux mymux
Removed mux "mymux" from mirror "mymirror"
admin@EXALINK-FUSION(config-mirror:mymirror)> no port A1
Removed port "A1" from mirror "mymirror"
```

Removing a mirror

When a port mirror is no longer needed it is possible to remove the mirror. Using the `no` form of the mirror command:

```
admin@EXALINK-FUSION> config no mirror mymirror
Mirror "mymirror" deleted
```

Timestamping

The ExaLINK Fusion timestamps packets upon arrival using a 350MHz 64 bit timestamp counter. Mirror output ports can be configured to transmit the timestamp along with packet in a number of

different modes using the `timestamp` command. Timestamps are taken at the moment the first bit of the incoming packet arrives.

Exablaze has a utility available on GitHub that allows reception and decoding of these timestamped streams. This can be used to write out a pcap file with nanosecond accurate timestamps.

In order to replace the Ethernet FCS (Frame Check Sequence) with the lower 32 bits of the timestamp counter, the timestamp mode should be set to `fcs`:

```
admin@EXALINK-FUSION> config mirror mymirror timestamp fcs
Set timestamp mode "fcs" on mirror "mymirror"
admin@EXALINK-FUSION> config mirror mymirror no timestamp
Timestamping disabled on "mymirror"
```

Each increment of the 32 bit counter represents approximately 2.86 ns of time. The lower 32 bits of this counter wrap every 12.3 seconds. To allow the capturing host to reconstruct the absolute UTC (Universal Time Coordinated) time of each packet, the mirror port will send periodic keyframes that are not part of the capture stream. These keyframes contain the full 64 bit counter and the UTC time in nanoseconds associated with the counter value.

These keyframes and FCS timestamps allow the remote host to reconstruct the absolute time of arrival of all packets in the captured feed.

The ExaLINK Fusion also supports an FCS Compatibility mode, whereby instead of replacing the FCS with the lower 32 bits of the timestamp counter the FCS is replaced with a timestamp format compatible with Arista devices and systems or applications designed to recognize and process the timestamp format provided by Arista. Similar to the FCS mode, the `timestamp` command can be used again:

```
admin@EXALINK-FUSION> config mirror mymirror timestamp fcs-compat
Set timestamp mode "fcs-compat" on mirror "mymirror"
```

In this format the timestamp counter's lower 31 bits are written over the FCS, with a gap found in the highest order bit of the lower order byte is a zeroed padding (i.e. 31 bits of data in a 32-bit format). Note that even in FCS Compatibility mode, timestamps are taken immediately upon packet arrival at an ingress port, and not upon arrival of the FCS field.

For instances where it is desirable for the timestamped frame to include a valid FCS, another two modes are supported where the timestamp is inserted in place of the original FCS bytes, and a new FCS for the whole packet is calculated and transmitted, therefore increasing the size of each transmitted frame by 4 bytes. Again, the selection between standard and compat modes are available:

```
admin@EXALINK-FUSION> config mirror mymirror timestamp append
Set timestamp mode "append" on mirror "mymirror"
admin@EXALINK-FUSION> config mirror mymirror timestamp append-compat
Set timestamp mode "append-compat" on mirror "mymirror"
```

Note

When a mirror object is configured to append a new FCS, the traffic path through the mirror changes from "cut through" to "store and forward". This does *not* impact the latency of traffic flow for mux/switch objects (or indeed any layer 1 paths).

Important

Accurate time synchronization of the ExaLINK Fusion is important when using a mirror with timestamping enabled. Synchronization allows the ExaLINK Fusion to provide keyframes that accurately correlate the internal timestamp counter with absolute "wall clock" time. Refer to System Time for details on setting and maintaining accurate time synchronization.

Mirror and timestamping - ExaLINK Fusion HPT

Note

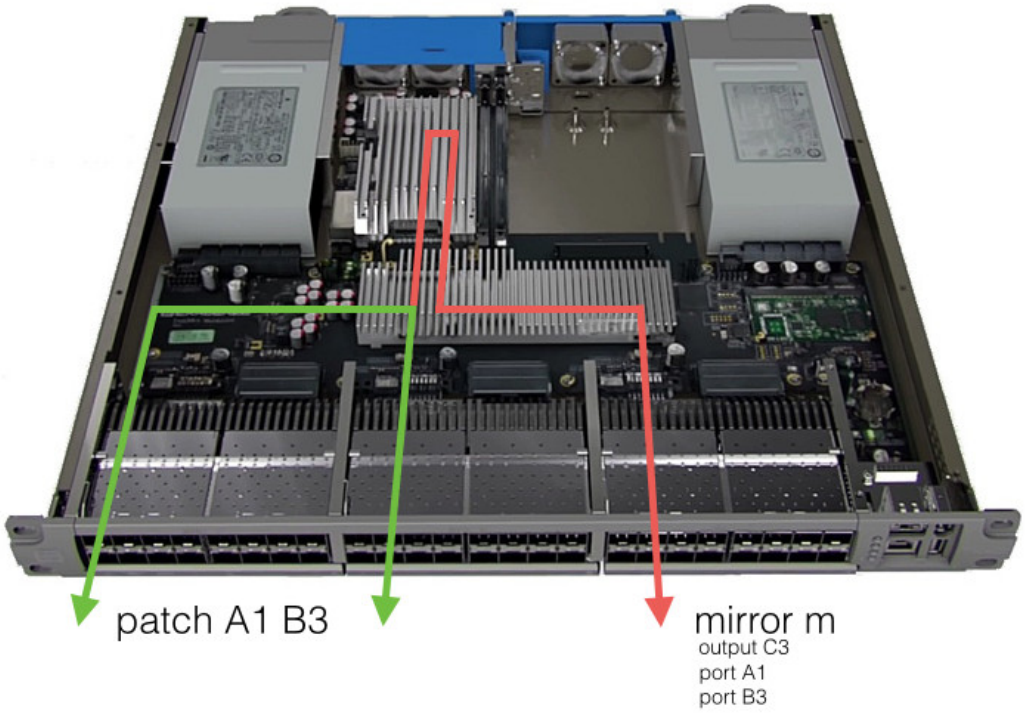
This page describes the operation of mirror objects for the **ExaLINK Fusion HPT**. For details on the operation of mirror objects for the ExaLINK Fusion, please refer to this page.

The mirror object provides a mechanism for configuring the logging and timestamping functionality within the ExaLINK Fusion HPT.

All packets that enter the mirror object are timestamped on arrival at the ingress port to a resolution of 100ps, along with other metadata such as port number and a device ID. This metadata is then appended to the incoming packet and sent to one or more output ports.

The supported port speeds varies depending on the currently selected firmware mode. The `hpt` firmware supports 1G and 10G ingress ports, and the `hpt-40g` firmware supports 10G and 40G ingress ports. Refer to this section for details on selecting firmware.

Support for 1G and 40G ingress ports requires version 1.12.0 or later.



Example use of a mirror, showing port A1 and port B3 mirrored to output port C3.

Creating a mirror

A mirror object is created using the `mirror` command and giving the object a name. The command must be completed in configuration mode, or prefixed with the `configure` command:

```
admin@EXALINK-FUSION(config)> mirror mymirror
Mirror "mymirror" created
```

Creating a mirror will place the command line into a config-mirror state.

Note

Only a single mirror object is supported on the Fusion HPT.

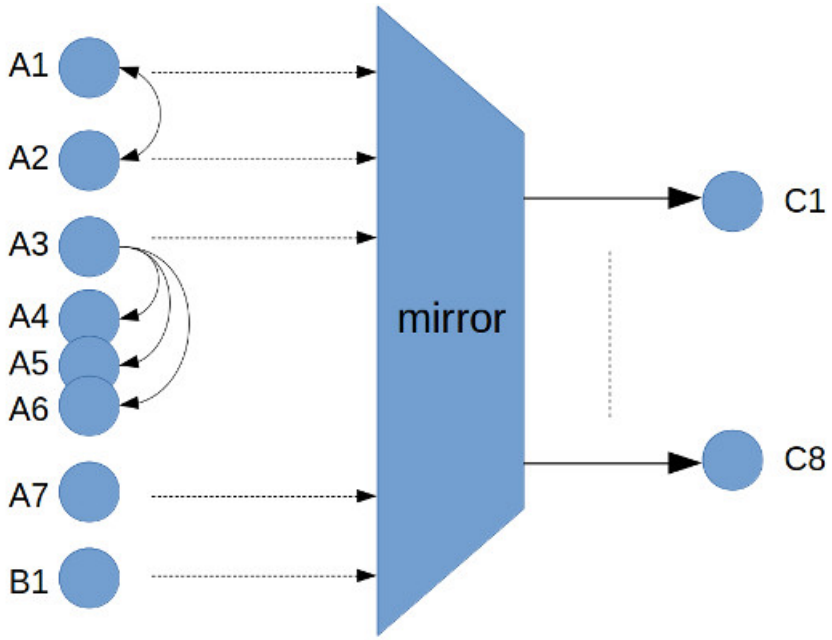
Input Ports

The mirror will aggregate and timestamp traffic entering any of its ports. For example, to aggreate all traffic received on ports A1 and B3:

```
admin@EXALINK-FUSION(config-mirror:mymirror)> port A1; port B3
Added port "A1" to mirror "mymirror"
Added port "B3" to mirror "mymirror"
```

Note: It is not possible to explicitly mirror the output of a port using a mirror object, only inputs to ports. In order to take a copy of all traffic leaving a port, it is suggested to use a tap object and tap the output port, or work out what the source port for this output port is, and add the source port to the mirror.

Input ports to a mirror object can also be part ofpatch or tap objects. For example, the following configuration is supported:



An example of a mirror with a number of inputs also used for patches and taps.

To remove a port from the mirror, use the `no` form of the commands:

```
admin@EXALINK-FUSION(config-mirror:mymirror)> no port A1
Removed port "A1" from mirror "mymirror"
```

Output Port(s)

One or more `output` ports needs to be assigned to the mirror object. To add a single output port, use the following command:

```
admin@EXALINK-FUSION(config-mirror:mymirror)> output C3
Added output port "C3" to mirror "mymirror"
```

Note: A port cannot be used for both an input and an output at the same time.

Multiple output ports can be used to share the output load across several ports, and is required in cases where the **sustained** aggregate input bandwidth is greater than 10Gb/s in order to prevent packet loss. This is done by creating a `bond` object, adding ports to the bond object, and assigning the bond object to be the mirror output. For example:

```
admin@EXALINK-FUSION(config)> bond mybond
Bond "mybond" created
admin@EXALINK-FUSION(config-bond:mybond)> port C1-C3
Added port "C1" to bond "mybond"
Added port "C2" to bond "mybond"
Added port "C3" to bond "mybond"
admin@EXALINK-FUSION(config-bond:mybond)> exit
admin@EXALINK-FUSION(config)> mirror mymirror output mybond
Added output "mybond" to mirror "mymirror"
```

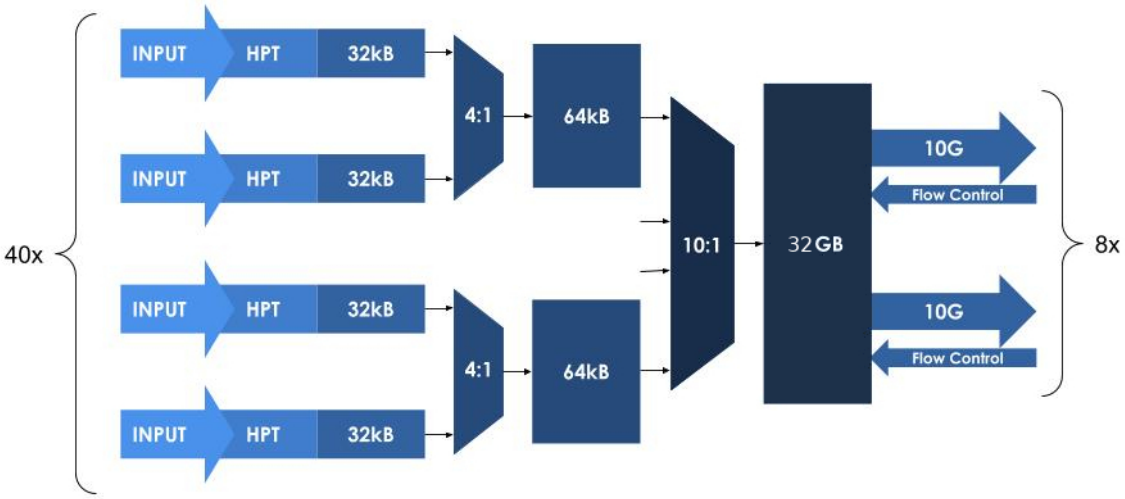
Removing a mirror

When the mirror object is no longer needed it can be removed with the `no` form of the mirror command:

```
admin@EXALINK-FUSION> config no mirror mymirror
Mirror "mymirror" deleted
```

Input Buffering Architecture

A multi-layered buffer design is implemented on the Fusion HPT to minimize the chance of dropping packets arriving into or traversing the device due to buffer overflow. The following diagram describes this design:



An overview of the buffer architecture used on the Fusion HPT.

The `show port` can be used to see if there have been any drops on a port, for example:

```
admin@EXALINK-FUSION> sh port c13
Port name          : C13
Address            : 643F5F8284BC
...
...
Link status        : down
Link up/down count : 0
Packets received   : 0
Bytes received     : 0
Receive errors     : 0
Packets sent       : 0
Bytes sent         : 0
Packets dropped    : 0
```

Note: By design, packet loss will only ever occur at a port's first input. The packet loss counter will therefore account for all loss for any given port.

The `show stats switch` command can be used to see how much of the 32GB buffer is being used:

```
admin@EXALINK-FUSION> show stats switch
Used memory : 655107840 bytes
```

Note: This is an instantaneous view of the buffer usage.

Output Format

All frames aggregated through the Fusion HPT mirror have a 16 byte trailer appended. This trailer includes the following information:

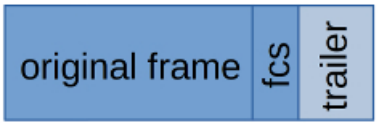
- Device identifier
- Port identifier
- Timestamp
- Flags
- Recalculated CRC for the whole frame

Note

Runt frames of less than 64 bytes are padded up to 64 bytes before the trailer is added.

Frames longer than 1600 bytes are truncated to 1600 bytes before the trailer is added.

The output frame format of the Fusion HPT is as follows:



The trailer format is:

Byte	Description
1	Device ID
2	Port ID (Refer Device and Port ID)
3-11	Timestamp
12	Flags
13-16	CRC (FCS)

As packets flow out of the DDR4 buffers shown in the buffering diagram above, a scheduler is used to select which output port packets should be sent (in the case when multiple output ports are configured). This scheduler will keep the egress bandwidth across these ports even.

Exablaze provides example software for the consumption of traffic coming out of the Fusion HPT. Plese refer to the timestamp-decoder software project for further details.

An example pcap file containing several output packets from a Fusion HPT can be downloadedhere.

Timestamp ordering

All packets arriving on any single input port are guaranteed to be output in timestamped order. However no guarantee is made between ports.

For example, consider a mirror configuration with a single input port (P1) and a single output port. In this case, if two packets arrive at P1 in the order P1.1, P1.2, then all packets will be output in timestamped order i.e. P1.1, P1.2

Now consider a configuration with 2 input ports (P1 and P2) and a single output port. In this case packets may arrive in order following order P1.1, P2.1, P1.2, P2.2. However, they may be output from the mirror in the following order: P2.1 P2.2 P1.1 P1.2. In this case, the arrival order of all packets on P1 is maintained (P1.1, P1.2), and the arrival order of all packets P2 is also maintained (P2.1, P2.2). However, the relative order of packets between port P1 and P2 has not been maintained.

Consumers of this data that require all packets to be in timestamp order (irrespective of the source port) will need to sort the packets in timestamp order.

Timestamp Format

The timestamp added to each packet is 9 bytes wide. The upper 32 bits are the seconds since epoch, and the lower 40 bits are the fractional seconds. For example, if the 9 byte timestamp is `0x5B7DEEAC_12FBD45A2E`, the time is:

```
0x5B7DEEAC == August 22, 2018 11:15:56 PM
0x12FBD45A2E == (81534409262 * 2^-40) = 0.07415511323597457
```

In other words, `20180822T231556.07415511323597457`

Flow control

The output ports of Fusion HPT mirror objects are able to accept IEEE 802.3x pause frames. This flow control signal is issued by some NICs to slow the rate of the incoming traffic. e.g in cases where

the capture device cannot keep up with the incoming traffic rate.

Device and Port ID

The device and port ID fields allow a consumer of traffic from the Fusion HPT to determine where a packet originated when multiple ports across multiple Fusion HPT's are used in an environment. There is a fixed mapping of input port name to the port ID value used. This is:

Port Name	Port ID
A1	1
A2	2
...	...
A16	16
B1	17
B16	32
C1	33
C16	48

The device ID is an integer between 0 and 255 can be assigned by the user as follows:

```
admin@EXALINK-FUSION(config-mirror:mymirror)> device-id 29
Set device ID to 29 on mirror "m1"
```

Flags

The following table describes what different bits in the flags field in the trailer format indicate.

Bit	Flag
0	Aborted frame
1	Truncated jumbo frame (larger than 1600B)
2	Frames dropped on this port (Not yet implemented)
3	Time synchronization lost (Not yet implemented)
4-7	Unused

Important

Accurate time synchronization of the ExaLINK Fusion is important when using a mirror with timestamping enabled. Synchronization allows the ExaLINK Fusion to provide keyframes that accurately correlate the internal timestamp counter with absolute "wall clock" time. Refer to System Time for details on setting and maintaining accurate time synchronization.

Virtual Ports

The ExaLINK Fusion has the concept of *virtual ports*. A virtual port pair can be thought of as a pipe with an A side and a B side, where traffic flows in both directions between A and B. Each side of a virtual port can be connected to another Fusion object, enabling a range of network and data flow possibilities.

Virtual ports are created using the `virtual-port` command, and must be named in the format V*number*, for example:

```
admin@EXALINK-FUSION> config virtual-port V1
Created virtual port pair V1A and V1B
```

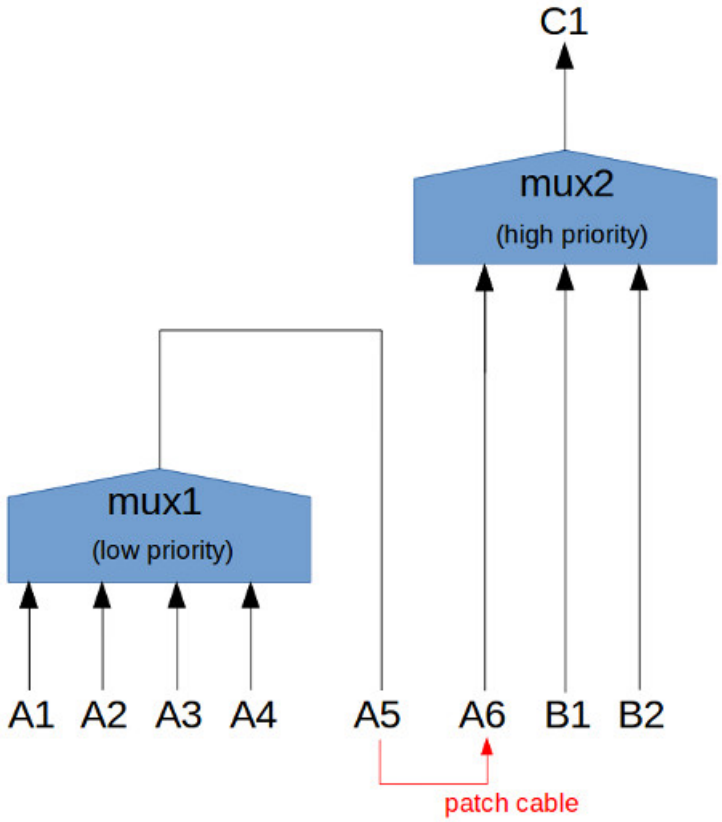
As with other commands, virtual ports can be removed with the `no virtual port` command.

Once created, each side of the virtual port can be used like any other port. The following contrived example shows a simple use of virtual ports:

```
admin@EXALINK-FUSION> config
admin@EXALINK-FUSION(config)> patch A1 V1A
Patch created between port "A1" and port "V1A"
admin@EXALINK-FUSION(config)> patch A2 V1B
Patch created between port "A2" and port "V1B"
```

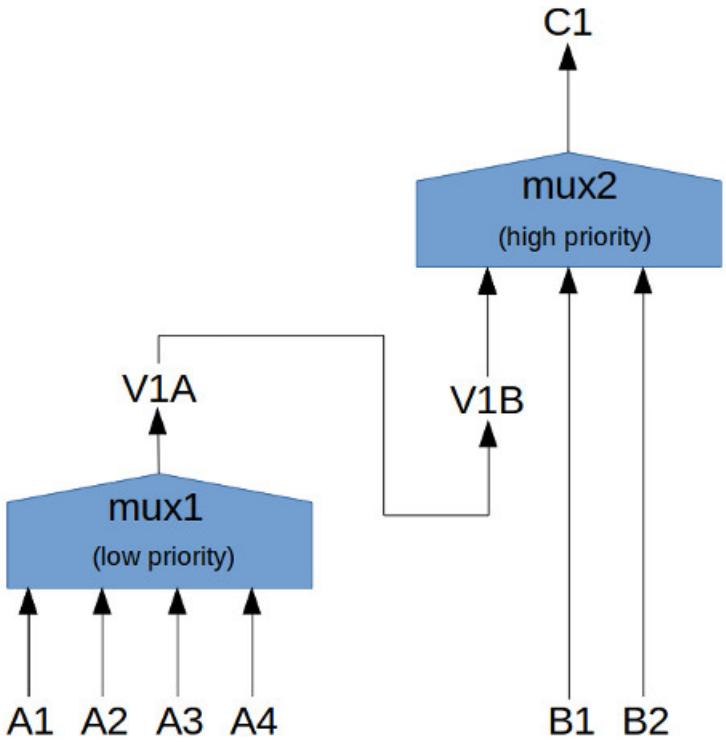
This would effectively create a patch between A1 and A2. Note this could have been done with `patch A1 A2`, however the intention here is to describe the use of virtual ports.

A more realistic example use of these objects would be if a user wanted to aggregate traffic from a number of servers towards a single destination, where some servers are considered higher priority than others. This could be implemented by the use of cascaded mux objects as follows:



Cascaded mux objects joined by an external patch cable

The use of virtual ports allows this cascading to be done without the external patch cable, for example:



Cascaded mux objects joined by virtual ports

The config to implement this would be:

```
admin@EXALINK-FUSION> config
admin@EXALINK-FUSION(config)> virtual-port V1
Created virtual port pair V1A and V1B
admin@EXALINK-FUSION(config)> mux mux1
Created mux "mux1"
admin@EXALINK-FUSION(config-mux:mux1)> port A1; port A2; port A3; port A4
Added downstream port "A1" to mux "mux1"
Added downstream port "A2" to mux "mux1"
Added downstream port "A3" to mux "mux1"
Added downstream port "A4" to mux "mux1"
admin@EXALINK-FUSION(config-mux:mux1)> port upstream V1A
Added upstream port "V1A" to mux "mux1"
admin@EXALINK-FUSION(config-mux:mux1)> exit
admin@EXALINK-FUSION(config)> mux mux2
Created mux "mux2"
admin@EXALINK-FUSION(config-mux:mux2)> port V1B, port B1; port B2
Added downstream port "V1B" to mux "mux2"
Added downstream port "B1" to mux "mux2"
Added downstream port "B2" to mux "mux2"
admin@EXALINK-FUSION(config-mux:mux2)> port upstream C1
Added upstream port "C1" to mux "mux2"
```

LLDP

This section contains information related to the Link Layer Discovery Protocol (LLDP) features on the ExaLINK Fusion. Fusions purchased as layer 1 only devices (ie those which do not have an FPGA module installed) are not capable of receiving or transmitting LLDP information, nor are Fusions running `fastmux` firmware.

Displaying LLDP neighbors

LLDP messages will always be received on LLDP capable ports, and the `show lldp neighbors` command can be invoked at any time to see what entries are in the ExaLINK Fusion's database, for example:

```
admin@@EXALINK-FUSION> show lldp neighbors
Local Port Chassis ID        Port ID           System Name
---------- ----------------- ----------------- --------------------
B1         F4:6D:04:8D:F6:41 64:3F:5F:01:14:A0 MyNetworkDevice
B2         F4:6D:04:8D:F6:41 64:3F:5F:01:14:A1 MyNetworkDevice
B3         F4:6D:04:8D:F6:41 64:3F:5F:01:14:A2 MyNetworkDevice
B4         F4:6D:04:8D:F6:41 64:3F:5F:01:14:A3 MyNetworkDevice
```

Additional information can be shown by issuing the same command from within the port modal state, for example:

```
admin@EXALINK-FUSION> port B1
admin@EXALINK-FUSION(port:B1)> show lldp neighbors
Port                  : B1

Chassis ID type       : MAC address
Chassis ID            : F4:6D:04:8D:F6:41
Port ID type          : MAC address
Port ID               : 64:3F:5F:01:14:A0
System name           : MyNetworkDevice
System description    : An LLDP enabled device on my network

Management address    : 124.15.3.133
Interface number type : Unknown
```

Configuring LLDP

The broadcast of LLDP frames which would allow remote parties to detect the identity of the ExaLINK Fusion is configurable on a port-by-port basis, allowing for less congestion on ports that do not require LLDP. Enabling of LLDP transmit must be invoked through the config mode for a specific port:

```
admin@EXALINK-FUSION> config port B1
admin@EXALINK-FUSION(config-port:B1)> lldp transmit
Port B1 LLDP transmit enabled
```

As long as the specified port forms part of a mux or switch object, it will transmit LLDP frames periodically out that port. Note that this configuration can be done when a port is not a part of an object, however transmission of LLDP frames will not begin until the port is part of a mux or switch object.

Note also that if a mux object is in `raw` mode, ie where the traffic path from the upstream port to the downstream ports is broadcast via layer 1 only, it's not possible for LLDP transmits to go out a downstream port.

Transmission of LLDP frames can be disabled with the `no lldp transmit` command:

```
admin@EXALINK-FUSION> config port B1
admin@EXALINK-FUSION(config-port:B1)> no lldp transmit
Port B1 LLDP transmit disabled
```

SNMP

This section covers the ExaLINK Fusion's support for SNMP v2c. The MIB file for the ExaLINK Fusion can be downloaded here.

Show configuration

The current status and configuration for SNMP can be obtained by issuing the following command:

```
admin@EXALINK-FUSION> config show snmp
SNMP status    : disabled
Location       :
Contact        :
Community name :
Listen port    : 161 (default)
SNMP traps     : disabled
```

To view just the status of all services:

```
admin@EXALINK> show services
Service        Status
-------------- --------
telnet         enabled
remote-logging disabled
snmp           disabled
snmptrap       enabled
http           enabled
```

Configuring SNMP

Before SNMP can be used, the shared read community phrase should be set. This is shared between the ExaLINK Fusion and the system making the SNMP requests. It can be set, for example:

```
admin@EXALINK> config snmp read community public
SNMP configuration updated
```

After which, the service may be enabled

```
admin@EXALINK> config snmp enable
SNMP enabled
```

The location of the ExaLINK Fusion can be set in the SNMP configuration. This is available to SNMP through the OID SNMPv2-MIB::sysLocation.0. Similarly, the configured contact details are available through SNMPv2-MIB::sysContact.0

```
admin@EXALINK> config snmp location "server room, 6th floor"
SNMP configuration updated

admin@EXALINK> config snmp contact "The Sys Admin <sysadmin@company.org>"
SNMP configuration updated

admin@EXALINK-FUSION> config show snmp
SNMP status    : enabled
Location       : server room, 6th floor
Contact        : The Sys Admin <sysadmin@company.org>
Community name : public
Listen port    : 161 (default)
SNMP traps     : disabled
```

SNMP traps

The ExaLINK Fusion has the ability to send SNMP notifications on important events. To enable this feature, use the command:

```
admin@EXALINK> config snmp trap enable
SNMP Trap enabled
```

To specify where the SNMP notifications should be sent, one or more trap targets must be configured using the `snmp trap target` command:

```
admin@EXALINK> config snmp trap target 192.168.220.11 public
SNMP Trap configuration updated
```

To clear SNMP trap targets use the `no` form of the command:

```
admin@EXALINK> config no snmp trap
Reset SNMP Trap configuration
```

The current SNMP trap configuration is visible via the `show snmp` command:

```
admin@EXALINK-FUSION> config show snmp
SNMP status    : enabled
Location       : server room, 6th floor
Contact        : The Sys Admin <sysadmin@company.org>
Community name : public
Listen port    : 161 (default)
SNMP traps     : enabled

Target address Community
-------------- ---------
192.168.220.11   public
192.168.220.12   public
```

SNMP v3

SNMP v3 can be configured to operate on the ExaLINK Fusion. If an administrator chooses to specify SNMP v3 users then this will change the behaviour of the SNMP system, providing the ability to authentic users,encrypt traffic and generate secure traps.

SNMP v3 requires users to be configured as part of the SNMP configuration. When the operator is adding these users they can decide whether the user should require no authentication, authentication only or authentication and encryption.

A SNMP v3 user can be added to the ExaLINK Fusion with the `configure snmp user` command. This follows the usage:

```
admin@EXALINK> configure snmp user
Usage: configure snmp user <name> [{md5-auth|sha-auth} <auth_password> [{aes-encryp
```

For example to add a user with no authentication and no encryption run:

```
configure snmp user tim
```

To configure a user with MD5 authentication but not encryption run:

```
configure snmp user tom md5-auth mypassword1
```

To configure a user with SHA authentication and AES encryption run:

```
configure snmp user jim sha-auth mypassword2 aes-encrypt mypassphrase3
```

To view the configured SNMP users run the `show snmp` command which will include a table of the configured users:

```
User auth-type auth-password encrypt-type encrypt-phrase
---- --------- ------------- ------------ --------------
tim  (none)    (none)        (none)       (none)
tom  MD5       mypassword1   (none)       (none)
jim  SHA       mypassword2    AES          mypassphrase3
```

If no SNMP v3 users are configured `show snmp` will instead contain:

```
No SNMP v3 users configured
```

If an administrator has configured SNMP v3 users then when a TRAP is generated the SNMP v3 TRAP functionality will be enforced. The TRAP notifications will be sent to the specified targets using the configured authentication and encryption methods. The TRAP target will need to be configured in a similar manner to observe or act on the SNMP v3 TRAPS.

Before a TRAP target can configure the valid users, the operator will need to identify the engine ID of the SNMP source. This can be done by querying the SNMP OID for the `snmpEngineID`

```
$ snmpget -v 3 -l noAuthNoPriv -u tim EXALINK SNMP-FRAMEWORK-MIB::snmpEngineID.0
SNMP-FRAMEWORK-MIB::snmpEngineID.0 = Hex-STRING: 80 00 A9 20 03 64 3F 5F 80 C4 00
```

An example `snmptrapd.conf` file that would allow an operator to view SNMP v3 TRAPs raised from the users defined on the Fusion above would then be:

```
createUser -e 0x8000A92003643F5F80C400 tim
authuser log tim
createUser -e 0x8000A92003643F5F80C400 tom MD5 mypassword1
authuser log tom
createUser -e 0x8000A92003643F5F80C400 jim SHA mypassword1 AES mypassphrase3
authuser log jim
```

TACACS+

TACACS+ can be used with the ExaLINK Fusion for authentication, authorization and accounting. Care should be taken when configuring TACACS+ as it is possible to block access to the device.

Configuring TACACS+ client

Minimally, a TACACS+ server and a passphrase must be supplied, before this can be enabled. Any number of servers can be configured and these are checked in order, until a TACACS+ server is found to which a connection can be established. Depending on your TACACS+ server side configuration, you may need to set the service key as well.

```
admin@EXALINK-FUSION> configure tacacs server 192.168.220.14 192.168.220.15
Using TACACS+ server 192.168.220.14, 192.168.220.15
admin@EXALINK-FUSION> configure tacacs secret m1-s3cr3t
Updated TACACS+ secret key
admin@EXALINK-FUSION> configure tacacs service system
TACACS+ service name set to "system"
admin@EXALINK-FUSION> show tacacs
Servers    : 192.168.220.14 192.168.220.15
Secret     : (hidden)
Service    : system (default)
Timeout    : 5.0 s (default)
Periodic   : disabled
Accounting : disabled
Status     : disabled
```

In this example, `192.168.220.14` is the primary server, and `192.168.220.15` the secondary. The timeout is the time required for the full authentication and authorization to complete.

TACACS+ uses TCP port 49 by default, however this can be changed by specifying the port number after the server's IP address, for example:

```
admin@EXALINK-FUSION> configure tacacs server 192.168.220.14:1111
Using TACACS+ server 192.168.220.14:1111
```

Accounting may be disabled, set to standard or set to verbose.

```
admin@EXALINK-FUSION> configure tacacs accounting standard
TACACS+ accounting level set to "standard"
admin@EXALINK-FUSION> show tacacs
Servers    : 192.168.220.14 192.168.220.15
Secret     : (hidden)
Service    : system (default)
Timeout    : 5.0 s (default)
Periodic   : disabled
Accounting : standard
Status     : disabled
```

The EXALINK Fusion may be configured to periodically check the authorization level of logged on users. If the TACACS+ server returns a different privilege level for the user, then the roles of the user will be updated appropriately.

```
admin@EXALINK-FUSION> configure tacacs periodic 120
TACACS+ periodic reauthorization set to 120.0 s
admin@EXALINK-FUSION> show tacacs
Servers    : 8192.168.220.14 192.168.220.15
Secret     : (hidden)
Service    : system (default)
Timeout    : 5.0 s (default)
Periodic   : 120.0 s
Accounting : standard
Status     : disabled
```

Before enabling TACACS+, ensure the server side is configured, and there is a user with administrator rights (privilege level 15). Enable TACACS+

```
admin@EXALINK-FUSION> configure tacacs enable
TACACS+ enabled
```

Now, before this session times out, try to start a second session logging on with the remote administrator credentials. If this session is able to log on, verify that this user has administrator rights, for example, by trying to change the accounting level:

```
admin@EXALINK-FUSION> configure tacacs accounting verbose
TACACS+ accounting level set to "verbose"
```

If this succeeded, then all is well, and you have a working system that is using the remote TACACS+ server to authorize users. However, if this second session failed, then there is a problem in the client or server's configuration and it is recommended to disable TACACS+ on the client until the configuration is adjusted.

```
admin@EXALINK-FUSION> configure tacacs disable
TACACS+ disabled
```

Note

When TACACS+ is enabled any locally defined users will not be able to login using their local credentials. Any local accounts that have the same username as an account on the server will take roles on permissions as defined on the server.

Recovery

If no TACACS+ servers are available, then the system falls back to local authentication. This means that if access to the ExaLINK Fusion is blocked by TACACS+, it is possible to recover by disabling access to the TACACS+ server(s). If the TACACS+ servers can be turned off, then access can be regained using the local administrator, `admin`. Alternatively, the network cable for the management interface on the ExaLINK Fusion can be removed, and the serial port used to logon as the local administrator.

TACACS+ server side configuration

The ExaLINK Fusion uses the privilege level returned by the initial authorization to determine which permissions to grant the remote user. A description of how to configure TACACS+ servers is beyond the scope of this document, but the key points are described here.

1. The ExaLINK Fusion may need to be configured on the TACACS+ server side as an recognized client, for example by adding its IP address.
2. The service type configured on the client ('system' in the example above), must match the configuration on the server side.
3. Users must be configured to return default privilege levels. Depending on the TACACS+ server, this may be done on a per user level, or on the group that the user belongs to, or as a custom attribute of the form `priv-lvl=XX`, where XX is the privilege level.
4. At least one user must have administrator rights, that is, privilege level 15.

Users of the tac_plus TACACS+ server should note that a service of `shell` requires additional parameters not currently supported by the Fusion. A service of `system` is known to work well with the Fusion and tac_plus. A simple config script for tac_plus can be downloaded here.

TACACS+ user permissions

The privilege level as returned by the TACACS+ server is used to map the user to a role. Please refer to User Management for more information on roles and their permissions and capabilities.

Privilege	Role	Permissions
0-2	guest	limited to reading system information
3-4	monitor	can read all information
5-8	user	can configure ports
9-14	operator	full access to non-security related settings
15	admin	full access

TACACS+ accounting

The command line interface on the ExaLINK Fusion is a thin layer, through which calls are made to exalinkd, a daemon service. The interface to the service is through a JSON RPC API, which is accessible to other clients and user interfaces.

In order to capture the requests from all clients, the JSON API calls are accounted. A distinction is made between calls that request information and those that change state. The setter functions are accounted when the accounting level is set to standard. When the level is set to verbose, then the getter functions are also accounted. Calls that fail, for example due to bad parameters, are not accounted.

Access control

Access to the ExaLINK Fusion management interface can be controlled through access control rules, which can be used to allow or deny specific IP address ranges.

Care should be taken when specifying access control rules, otherwise it is possible to block all access. If services are used that require access to other servers, such as SNMP or TACACS+, remember to add rules to allow access to those machines.

Configuring access control

Typically rules are set to allow a certain range of addresses and block all others. Add the `allow` rules before the `deny` rules.

To grant access to connections originating from IP addresses `192.168.220.*` and `192.168.7.1`:

```
admin@EXALINK> configure management access-list allow 192.168.220.0/24
Access control rules updated
admin@EXALINK> configure management access-list allow 192.168.7.1
Access control rules updated
admin@EXALINK> show management access-list
Policy Address
------ -------------
allow  192.168.220.0/24
allow  192.168.7.1
```

To deny access from all other addresses:

```
admin@EXALINK> configure management access-list deny 0.0.0.0/0
Access control rules updated
admin@EXALINK> show management access-list
Policy Address
------ -------------
deny   0.0.0.0/0
allow  192.168.220.0/24
allow  192.168.7.1
```

To reset the rules:

```
admin@EXALINK> configure no management access-list
Access control rules reset
```

Recovery

If the rules are entered in the wrong order, or are entered incorrectly, you can block your own access. To recover from this, simply use the serial port of the ExaLINK Fusion for access to log on and change the rule set.

Statistics

The ExaLINK Fusion collects data traffic statistics when a FPGA module with `mux`, `switch` or `fastmux` firmware is installed.

Sent and received packet counters can be viewed using the `show port` command. Please refer to the port page for more information.

Latency statistics

Note: This feature is for the `mux` firmware only.

The ExaLINK Fusion records aggregated statistics on the time it takes for packets to traverse the device when using switch or mux objects. The latency statistics can be viewed using the `show latency` command:

```
admin@EXALINK-FUSION> port B1 show latency
Total packets : 5858442

Percentile Latency
---------- -------
maximum    107 ns
99.99      107 ns
99.90      107 ns
99.00      101 ns
95.00      101 ns
90.00      101 ns
75.00      101 ns
50.00      95 ns
25.00      95 ns
minimum    95 ns
```

The latency number is the time it takes from the packet entering the ingress port at the front panel, to the time it leaves the egress port at the front panel.

By default, the `show latency` command shows latency statistics for the last minute. To show the latency statistics for a different length of time, the `last` argument can be used to specify the number of minutes of statistics to show. For example, to show the latency statistics for the last 10 minutes:

```
admin@EXALINK-FUSION> port B1 show latency last 10
```

Because of memory limitations, older latency data is aggregated, so the statistics shown using this command may include more than the requested length of time.

BGP Peering

This section contains information related to the Border Gateway Protocol (BGP) features on the ExaLINK Fusion. Fusions purchased as layer 1 only devices (ie those which do not have an FPGA module installed) are not capable of receiving or transmitting BGP information, nor are Fusions running `fastmux` firmware.

The Border Gateway Protocol (BGP) is an inter-Autonomous System routing protocol. The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems.

The Fusion can enable limited BGP support on a router object to allow the Fusion to connect to a remote BGP peer and advertise routes. Note that currently no routing is done by the Fusion.

Router Objects

The router object is used to define layer 3 functionality on the Fusion. Since routing is currently not supported by the Fusion, a router object can have at most one port.

Router objects are created using the `router` command from within configuration mode.

```
admin@EXALINK-FUSION(config)> router Primary
Created router "Primary"
```

A front panel port can then be added to the router object, and an IP address assigned.

```
admin@EXALINK-FUSION(config-router:Primary)> port A16
Added port "A16" to router "Primary"
admin@EXALINK-FUSION(config-router:Primary port:A16)> ip-address 10.30.1.10 255.255
Set IP address on port "A16" on router "Primary"
```

The router object can now be configured to speak BGP on this port.

Note: The router port can be the same as the upstream port of a mux object. This allows layer 2 aggregation of BGP traffic from a router object with traffic from the downstream ports of a mux object.

Note

Router objects currently have **only** 1 port and are **only** used for connecting to a BGP peer. There is no routing functionality implemented at the moment.

Displaying BGP Configuration

When configuring a router object the `show bgp` command can be invoked to see the current BGP configuration. The initial empty configuration on a router object called "Primary" should look as follows:

```
admin@EXALINK-FUSION(config-router:Primary)> show bgp
BGP       : disabled
Router ID : default
Password  : disabled

No networks are configured

No neighbors are configured
```

Configuring BGP

The AS (Autonomous System) number is one of the essential element of BGP. The AS number is a two octet value, ranging in value from 1 to 65535. The AS numbers 64512 through 65535 are defined as private AS numbers. Private AS numbers must not be advertised in the global Internet.

Configuring the BGP AS number:

```
admin@EXALINK-FUSION(config-router:Primary)> bgp as-number 65123
Configured BGP AS number on router "Primary"
```

Removing the BGP AS number:

```
admin@EXALINK-FUSION(config-router:Primary)> no bgp as-number
Removed BGP AS number on router "Primary"
```

The router ID indicates the BGP Identifier of the sender of BGP messages. A given BGP speaker sets the value of its BGP Identifier to an IP address assigned to that BGP speaker.

Configuring the BGP router ID:

```
admin@EXALINK-FUSION(config-router:Primary)> bgp router-id 10.30.1.10
Configured BGP router ID on router "Primary"
```

Removing the BGP router ID:

```
admin@EXALINK-FUSION(config-router:Primary)> no bgp router-id
Using default BGP router ID on router "Primary"
```

Networks

The Fusion BGP client may want to announce its own networks to other neighbors. To do this the `bgp network` command is used.

```
admin@EXALINK-FUSION(config-router:Primary)> bgp network
Usage: bgp network <address> <prefix>
Add BGP network
```

For example the following configuration adds the network 172.18.10.0/24 to be announced to all neighbors.

```
admin@EXALINK-FUSION(config-router:Primary)> bgp network 172.18.10.0 24
Added BGP network "172.18.10.0" / "24" on router "Primary"
```

To remove the announced network:

```
admin@EXALINK-FUSION(config-router:Primary)> no bgp network 172.18.10.0 24
Removed BGP network "172.18.10.0" / "24" on router "Primary"
```

Neighbors

This command adds new neighbors, which allows a user to configure the peers that the Fusion BGP client can peer with. The AS number is the unique identifier of the remote peer and the address is the peer IPv4 address.

```
admin@EXALINK-FUSION(config-router:Primary)> bgp neighbor
Usage: bgp neighbor <AS number> <address> [<password>]
Add BGP neighbor
```

Many service providers use a pre-shared key and MD5 checksum for protecting their BGP sessions. In a protected BGP session, a transmitting BGP router generates a MD5 hash value using the pre-shared key and portions of the packet. This checksum is included within the transmitted packet as a TCP option field. Upon receipt of the packet, a receiving router uses the same method to generate and validate the received checksum with its version of the MD5 checksum.

Enabling a protected BGP session is optional and is implemented as an optional password for neighbor configuration. Configuring the BGP neighbor with a password (pre-shared key):

```
admin@EXALINK-FUSION(config-router:Primary)> bgp neighbor 65456 10.30.1.20 myneighb
Added BGP neighbor "65456" via address "10.30.1.20" with password "myneighborpasswo
```

Configuring the BGP neighbor without a password:

```
admin@EXALINK-FUSION(config-router:Primary)> bgp neighbor 65789 10.30.1.30
Added BGP neighbor "65789" via address "10.30.1.30" with no password on router "Pri
```

Removing the BGP neighbor:

```
admin@EXALINK-FUSION(config-router:Primary)> no bgp neighbor 65789 10.30.1.30
Removed BGP neighbor "65789" via address "10.30.1.30" on router "Primary"
```

Enabling / Disabling BGP

To initiate the BGP peering, the BGP system should be enabled as follows:

```
admin@EXALINK-FUSION(config-router:Primary)> bgp enable
Enabled BGP on router "Primary"
```

To show the full BGP details after it has been configured, re-run `show bgp`:

```
admin@EXALINK-FUSION(config-router:Primary)> show bgp
BGP       : enabled
AS number : 65123
Router ID : 10.30.1.10
Password  : disabled

Network
---------------
172.18.10.0/24
172.18.11.0/24
172.18.12.0/24

Neighbor   AS number Password
---------- --------- ------------------
10.30.1.20 65456     myneighborpassword
10.30.1.30 65789
```

To protect from accidental disconnections from peers the BGP system will not allow primary functions from being changed while BGP is enabled. A user will also be prompted when disabling to prevent accidental loss of announced networks.

Example of disconnection protection:

```
admin@EXALINK-FUSION(config-router:Primary)> bgp as-number 123456
Error: BGP must be disabled before editing.

admin@EXALINK-FUSION(config-router:Primary)> bgp disable
Are you sure you want to disable BGP for this router? yes
Disabled BGP on router "Primary"
```

The Router password allows for setting the localhost link password. This can be left as default for standard operation. Configuring the BGP router password:

```
admin@EXALINK-FUSION(config-router:Primary)> bgp password mypassword
Configured BGP password on router "Primary"
```

Removing the BGP Router password:

```
admin@EXALINK-FUSION(config-router:Primary)> no bgp password
Removed BGP password on router "Primary"
```

Displaying BGP Status

To observe the current BGP status of the Fusion BGP peer, run `show bgp status`:

```
admin@EXALINK-FUSION(config-router:Primary)> show bgp status
Router Primary BGP status:
BGP table version is 0, local router ID is 10.30.1.10
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  10.30.0.0/16     10.30.1.20               1             0 65456 ?
*>                  0.0.0.0                  1         32768 ?
*> 172.18.10.0/24   0.0.0.0                  0         32768 i
*> 172.18.11.0/24   0.0.0.0                  0         32768 i
*> 172.18.12.0/24   0.0.0.0                  0         32768 i
*> 192.168.10.0/25  10.30.1.20               0             0 65456 i

Total number of prefixes 5
```

To observe the current state of known BGP neighbors, run `show bgp neighbors`:

```
admin@fusion5(config-router:Primary)> show bgp neighbors
Router Primary neighbor status:
BGP neighbor is 10.30.1.20, remote AS 65456, local AS 65123, external link
  BGP version 4, remote router ID 10.30.1.20
  BGP state = Established, up for 00:01:41
  Last read 22:39:03, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
                         Sent       Rcvd
    Opens:                  2          0
    Notifications:          0          0
    Updates:                2          2
    Keepalives:             3          2
    Route Refresh:          0          0
    Capability:             0          0
    Total:                  7          4
  Minimum time between advertisement runs is 30 seconds

 For address family: IPv4 Unicast
  Community attribute sent to this neighbor(both)
  2 accepted prefixes

  Connections established 1; dropped 0
  Last reset never
  External BGP neighbor may be up to 64 hops away.
Local host: 10.30.1.10, Local port: 179
Foreign host: 10.30.1.20, Foreign port: 57011
Nexthop: 10.30.1.10
Nexthop global: ::
Nexthop local: ::
BGP connection: non shared network
Read thread: on  Write thread: off
```

To observe the current summary of the BGP client, run `show bgp summary`:

```
admin@fusion5(config-router:Primary)> show bgp summary
Router Primary BGP summary:
BGP router identifier 10.30.1.10, local AS number 65123
RIB entries 9, using 576 bytes of memory
Peers 1, using 2524 bytes of memory

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.30.1.20      4 65456       5       8        0    0    0 00:02:50        2

Total number of neighbors 1
```

Bash Shell

This feature requires version 1.8.0 or later

The ExaLINK Fusion command line interface provides access to a bash shell, which supports many standard UNIX commands and the ability to run scripts.

Accessing the Bash shell

To access the bash shell from the CLI, use the `bash` command:

```
admin@EXALINK-FUSION> bash

ExaLINK Fusion shell

admin@EXALINK-FUSION:~$
```

The `bash` command is only available to users with the `admin` role.

Directories

- `/media/userfs` contains the startup configuration and other files which need to persist across updates. This is the only part of the filesystem that is preserved on a firmware update.
- `/media/userfs/debug` is a temporary directory for debug dumps. Its contents does not persist across reboots.
- `/media/userfs/update` is a temporary directory for firmware updates. This also does not persist across reboots.

Note that home directories are also temporary and no user files may be placed in them. Files which need to persist should be placed in `/media/userfs`.

Shell scripting

Shell scripts should be placed in `/media/userfs` or in a subdirectory and must have `#!/bin/bash` or `#!/bin/sh` as the first line. The executable bit will automatically be set for these files.

Using ExaLINK Fusion commands from the shell

The ExaLINK Fusion command interpreter can be accessed from the Bash shell using the `cli` command. Commands can be provided to the command interpreter using the `-c` argument, for example:

```
admin@EXALINK-FUSION:~$ cli -c 'show port'
Port Status Description
---- ------ -----------
...
```

This is useful if you want to use unix shell commands such as `grep` together with ExaLINK Fusion commands, for example:

```
admin@EXALINK-FUSION:~$ cli -c 'show port' | grep 'down'
```

It can also be used to run commands from shell scripts.

Python scripting

The ExaLINK Fusion has a Python interpreter installed.

```
admin@EXALINK-FUSION:~$ python
Python 2.7.3 (default, Jun  5 2017, 12:52:39)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Bindings are provided for access to the JSON-RPC API. This can be accessed by importing the module `exalink`.

```
>>> import exalink
```

This creates an `exalink` object in the current namespace. Fusion JSON-RPC API calls can be accessed as methods of this object, for example:

```
>>> print exalink.get_ports()
```

Python scripts should be placed in `/media/userfs` or in a subdirectory and must have `#!/usr/bin/python` as the first line. The executable bit will automatically be set for these files.

Cron jobs

The ExaLINK Fusion supports cron for running user-defined tasks at scheduled times.

The Cron configuration file is called *crontab*. This file can be edited by running the command `crontab -e` in the bash shell:

```
admin@EXALINK-FUSION:~$ crontab -e
```

Note that the `crontab` utility on the Fusion is specially modified to always edit the *admin* user's crontab. This means any users with access to the bash shell can edit the same crontab, and that all cron jobs will run as the user *admin*.

Here is an example crontab for running a script every 5 minutes:

```
*/5 * * * * /media/userfs/hello.py
```

Each line consists of 5 fields followed by a command. The 5 fields specify the time and day the command is to be run. In order, they are:

- Minute (0 to 59)
- Hour (0 to 23)
- Day of month (1 to 31)
- Month (1 to 12)
- Day of week (0 to 6)

Each field can be a number, a comma-separated list (eg. `5,10,15`), or `*`. The `/` syntax is also supported, so for example `*/5` in the first field means every 5 minutes.

Use the `crontab -l` command to list the current cron configuration:

```
admin@EXALINK-FUSION:/media/userfs$ crontab -l
*/5 * * * * /media/userfs/hello.py
```

Automatic Deployment and Configuration

This feature requires version 1.9.0 or later

This section contains information related to the automatic deployment and configuration features on the ExaLINK Fusion.

The automatic deployment and configuration functionality allows a user to setup their network in such a way that new Fusion devices added to that network will query a DHCP server to download and execute a script defined by the user. Executing this script means a user can configure or communcaite to and from the Fusion remotely at the time of deployment.

Displaying Automatic Configuration State

When inspecting the Fusion the `show auto-config` command can be invoked to see the current enabled or disabled state of the automatic configuration functionality.

For Fusion devices the initial empty configuration should look as follows:

```
admin@EXALINK-FUSION> show auto-config
Automatic startup configuration enabled
```

Note

Newly purchased Fusion devices will have this feature enabled to allow for streamlined deployment. Once configured or if not desired a user can disable this feature.

Configuring Automatic Configuration

First enter configuration mode of the Fusion.

```
admin@EXALINK-FUSION> configure
```

When in configuration mode a user can enable this feature with the `auto-config enable` command.

```
admin@EXALINK-FUSION(config)> auto-config enable
Enabled auto-configuration
```

To disable this feature

```
admin@EXALINK-FUSION(config)> auto-config disable
Disabled auto-configuration
```

or

```
admin@EXALINK-FUSION(config)> no auto-config
Disabled auto-configuration
```

Running `show auto-config` will now show it is in the disabled state

```
admin@EXALINK-FUSION> show auto-config
Automatic startup configuration disabled
```

The automatic configuration feature will only execute when the Fusion is connecting to a DHCP server (which is the factory default mode). This can either happen at start up or when the user configures the Fusion for DHCP mode. To do this run `management address dhcp`

```
admin@EXALINK-FUSION(config)> management address dhcp
Enabled DHCP on management interface
```

Remember to save the running configuration to the startup configuration if you want these settings to take remain over a reboot.

```
admin@EXALINK-FUSION(config)> copy running-config startup-config
Saved running config to startup config
```

Configuring a DHCP Server

If desired when a DHCP client starts, and after it receives a DHCPOFFER response from a DHCP server, the client can communicate directly with a boot server (instead of the DHCP server) to download a `boot file`.

The Fusion auto-config feature relies on this behaviour, that is its DHCP client being able to contact a DHCP server on a local LAN. The DHCP server can then be configured to provide a `boot file` address to the Fusion. The Fusion will fetch this file and apply it once downloaded. Note the `boot file` may have various names depending on the DHCP server e.g. `boot-file`, `filename` etc

Example script for dhcpd

```
#
# DHCP Server Configuration file.
#
option domain-name "autoconfig.exablaze.com";
default-lease-time 600;
max-lease-time 7200;
log-facility local7;

filename "http://192.168.111.11/fusion_autoconf.sh";

subnet 192.168.111.0 netmask 255.255.255.0 {
   range 192.168.111.111 192.168.111.130;
}
```

The Fusion can download a boot file via HTTP:

```
filename "http://192.168.111.11/fusion_autoconf.sh";
```

or TFTP:

```
filename "tftp://192.168.111.11/fusion_autoconf.sh";
```

Note

When downloading via HTTP boot file names should use HTTP encoding for special characters such as spaces. For example:

```
filename "http://192.168.111.11/my%20new%20pythontest.py";
```

Example script files

The script files will execute as if executed by the admin user on the Fusion. Therefore can operate standard tools like `ip a`, `journalctl` or Fusion tools such as `cli`. Calling `cli` allows execution of commands from bash that you would normally execute through the Fusion command line interpreter (`cli`)

A simple bash example to configure basic management details:

```
#!/bin/sh
cli -c 'configure management name-server 10.11.12.13'
cli -c 'configure hostname MYFUSION'
cli -c 'configure timesync gps'
cli -c 'configure copy running-config startup-config'
```

A simple bash example to configure the data path:

```
#!/bin/sh
cli -c 'conf patch A1 A2'
cli -c 'conf tap B1 B15'
cli -c 'conf tap B1 B16'
cli -c 'conf copy running-config startup-config'
```

A bash example configuring remote logging and snmp and then turning off auto-config once complete:

```
#!/bin/sh
cli -c 'conf snmp read community public'
cli -c 'conf snmp enable'
cli -c 'conf remote-logging target udp my-logging-server all all'
cli -c 'conf remote-logging enable'
cli -c 'conf auto-config disable'
cli -c 'conf copy running-config startup-config'
```

A Python example utilising the Fusion API which fetches a firmware update and applies it:

```
#!/usr/bin/python

# Example python script for running on Fusions
# Uses python bindings for the Fusion API
# Refer to https://fusion.exablaze.com/api/ for full API details

import exalink

exalink.set_hostname(hostname='MYFUSION')
exalink.set_management_address_ipv4(mode="static", static={"address":"172.16.0.153"
exalink.save_startup_config()
exalink.update_tftp(server="172.16.0.160", file="exalink_fusion_1.9.0.tar")
# note that when update_tftp() completes it will automatically reboot the box
```

A Python example configuring the data path:

```
#!/usr/bin/python

import exalink

exalink.create_patch(ports=["A10","C1"])
exalink.create_tap(port="A16", src_port="A10", direction="input")
exalink.create_object(type="mux", name="my_mux", mode="layer2")
exalink.add_object_port(object={"type":"mux","name":"my_mux"}, port="B1", side="up"
exalink.add_object_port(object={"type":"mux","name":"my_mux"}, port="B10", side="do
exalink.save_startup_config()
```

FPGA Development

The documentation for the ExaLINK Fusion FPGA development kit (FDK) has been moved to https://exablaze.com/docs/fdk-guide/

Note

Users will require an access code in order to view this documentation.

For users with an Exablaze SFTP account you can use your password, otherwise to obtain an access code please contact us at Exablaze support.

X86 Module Development

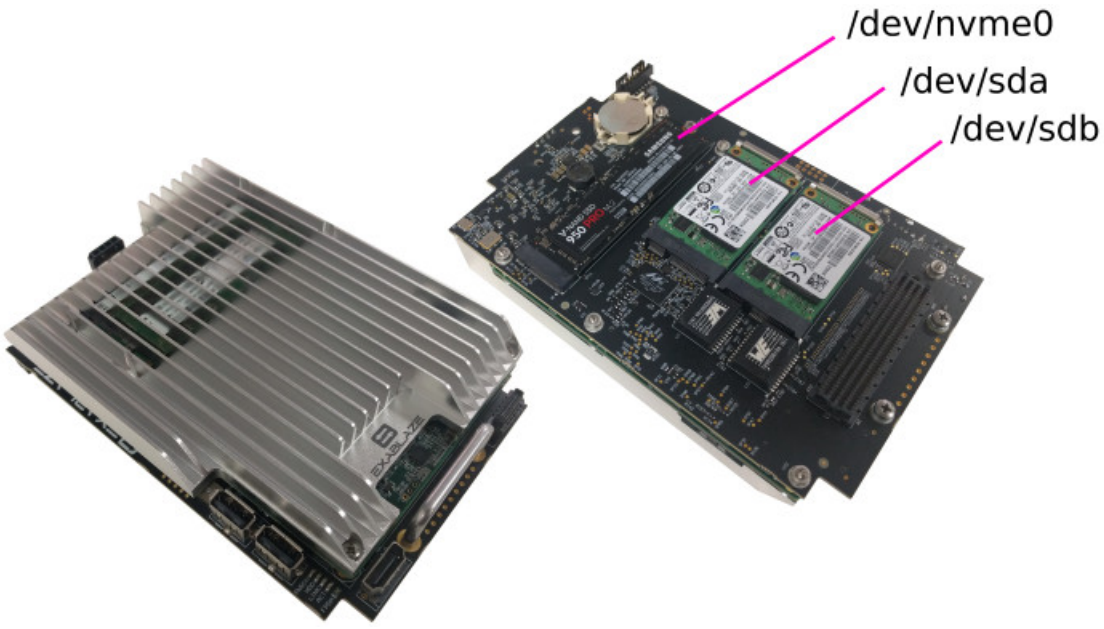
Overview

An Intel Skylake x86 Processor Module is available for installation into one of the internal module bays within the ExaLINK Fusion. This processor is available for you to run your own applications on, allowing construction and deployment of any number of custom appliances. The CPU is not required by Exablaze for running or management of the rest of the Fusion, so you are not virtualized or sandboxed in any way - you have complete control over the processor.

Specifications

The specifications of the processor module are as follows:

- Intel Core-i7 Skylake 6820EQ CPU, 4 cores @ 2.8GHz, 3.5GHz turbo, QM170 chipset
- 32GB DDR4-2133MT/s (PC4-17000)
- Dual mSATA SSDs, typically shipped with Samsung 850 EVO 250GB as RAID1 for operating system
- M.2 PCIe NVMe SSD, typically shipped with Samsung 950 PRO 512GB capable of sustained write at 1,500MByte/s
- Onboard low latency ExaNIC X40, providing 8x10Gbe (2x40G via firmware update) interfaces which can be connected to other Fusion objects (mux, switch, front panel ports etc)
- Onboard 1Gb LAN, typically patched through to front panel
- Trusted Platform Module (TPM)
- Intel AMT allows remote KVM access, including operating system install and BIOS config
- Multiple serial ports available
- Multiple FPGA development options available
- Access to PPS from front panel or Fusion GPS receiver for highly accurate time synchronization



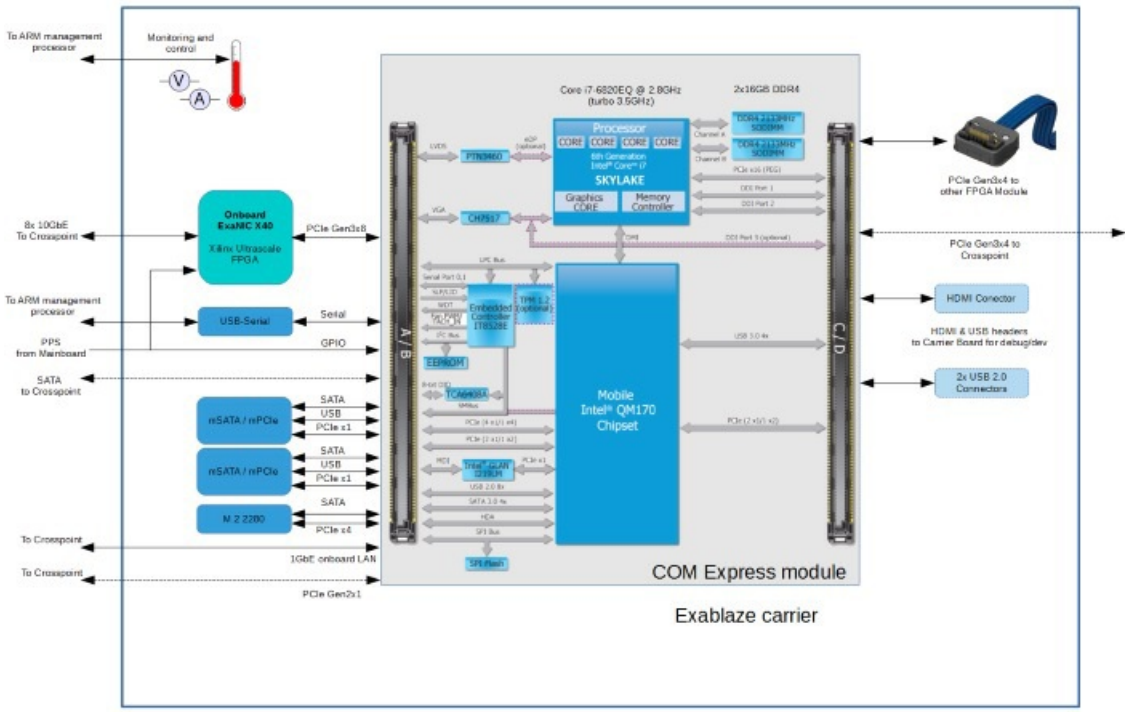
Top and bottom view of x86 Processor Module, showing drive bays

Architecture

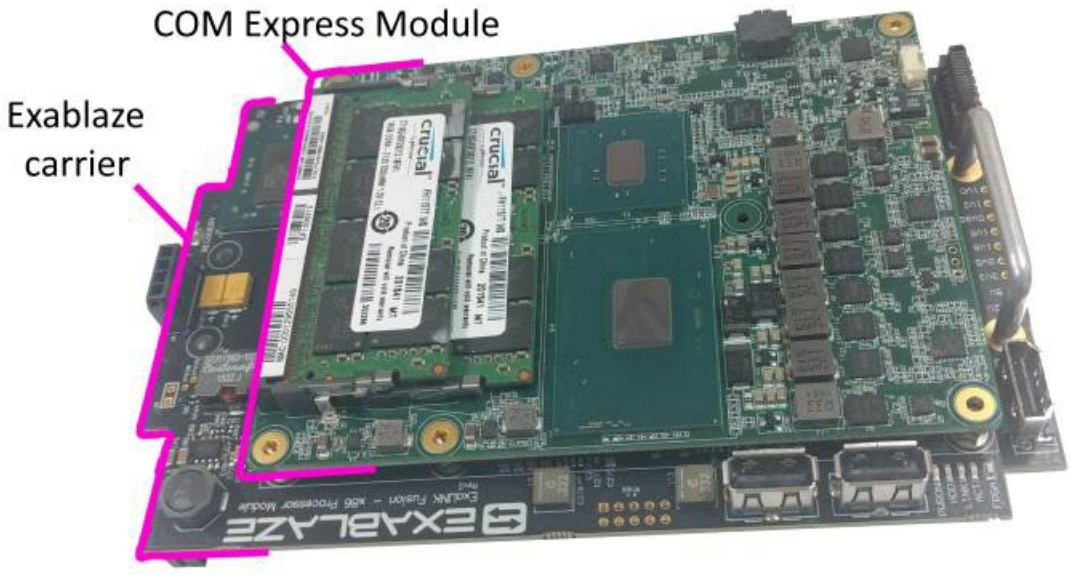
This module consists of a *carrier board* designed by Exablaze, and a 3rd party x86 processor board in a COM Express form factor. This design allows for potential upgrading of the x86 processor board to newer microarchitectures - for example Intel's Kaby Lake or Cannonlake, without changing the underlying carrier board.

The 3rd party processor board currently shipping is the SH960 from DFI.

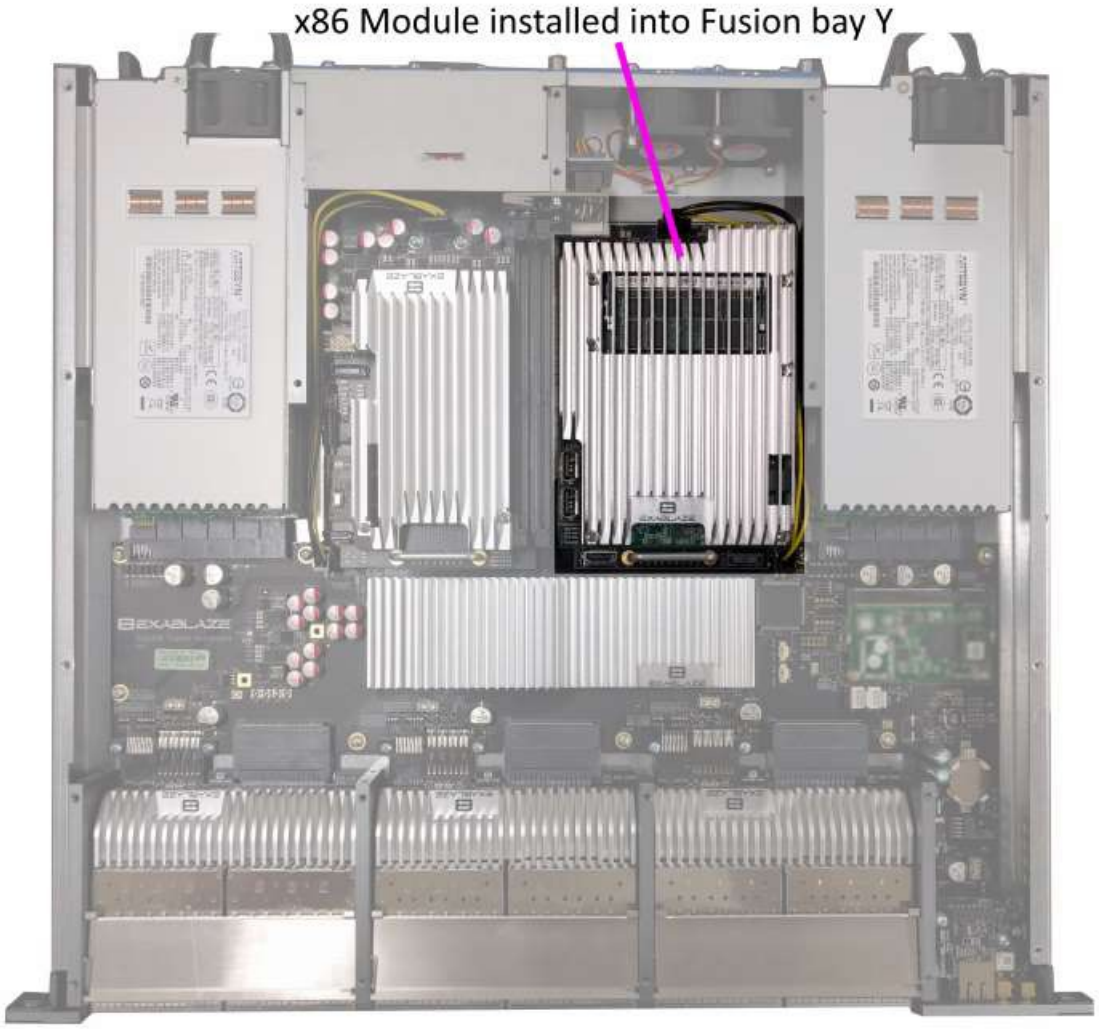
As can be seen below the carrier board features the ExaNIC X40, connectors for mounting mSATA's and the M.2 drive, along with all the required power supply, support and management hardware.



Block diagram of x86 Processor Module



x86 Processor Module consists of a COM Express module installed on Exablaze's carrier board



x86 Processor Module installed into Fusion

System Power Management

Power State

The Fusion CLI is used to enable/disable power to the processor module. Assuming the processor is installed into module Y, the module can be powered up/down using the following commands:

```
admin@EXALINK-FUSION> configure module Y power on
Module Y powered on
admin@EXALINK-FUSION> configure module Y power off
Module Y powered off
```

This is a hard power off, ie the equivalent of removing all power feeds to a server.

When operating, the power consumption of the processor module can be displayed using the `show power` command as follows:

```
admin@EXALINK-FUSION> show power
             Voltage  Current
             -------  -------
Line card A  12.3 V   0.3 A
Line card B  12.3 V   0.3 A
Line card C  12.2 V   0.5 A
Module X     12.3 V   0.9 A
Module Y     12.2 V   2.7 A
```

The temperature of the processor module can be read using the `show temperature` command, as follows:

```
admin@EXALINK-FUSION> sh temp
             Temperature
             ----------------------------
Crosspoint   36.9 C  36.9 C  34.3 C  34.3 C
Line card A  34.0 C
Line card B  35.4 C
Line card C  35.8 C
Mainboard    30.9 C  34.9 C
Module X     34.0 C  33.0 C
Module Y     52.0 C  51.0 C
```

The first temperature listed above for module Y is the carrier board temperature. The second is the temperature of the ExaNIC X40 FPGA. There is no way to directly read the CPU temperature from the Fusion CLI, however the CPU and FPGA are thermally coupled by the heatsink. The CPU temperature can be read by logging into the processor module itself and running an application such as lm_senors, i7z etc.

Installed Operating System

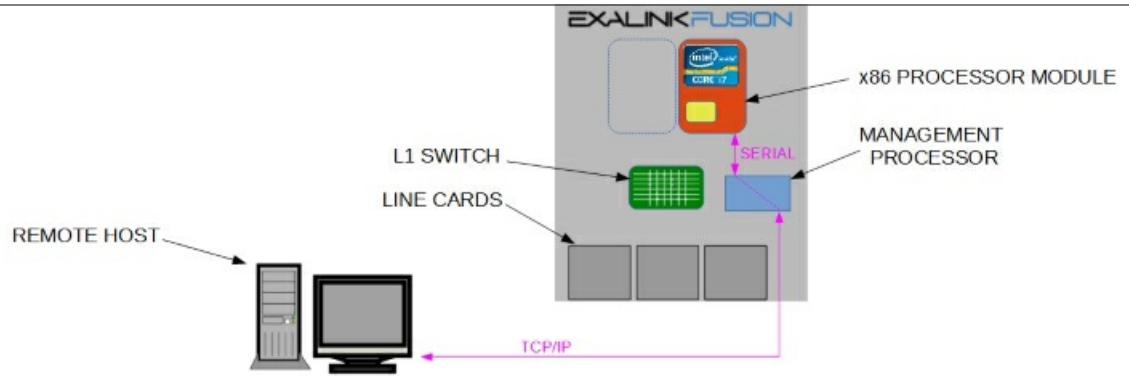
The default configuration ships with Linux CentOS 7 installed onto a RAID1 partition created from the dual mSATA drives. The M.2 NVMe drive is mounted at /data. The login for this system as it ships is:

```
username: root
password: exablaze
```

Refer below for console access and Network Interfaces connectivity options for logging into the operating system.

Primary Serial Port

There are 2 serial ports provided on the x86 processor module, both of which are connected through to the Fusion management processor. One of these can be used to gain console access to the x86 processor module. This is done by enabling a `serial-server` on the Fusion and connecting to it remotely via the Fusion management port. Using this feature allows you to access the x86 processor module for management purposes without sacrificing a high speed front panel port



As can be seen in the diagram above, the remote host connects to a serial server running on the Fusion management processor. Traffic received/transmitted on this connection is routed to the x86 processor module's serial port, where a getty session would typically be running, enabling console access to the system. Commands to configure this would be as follows:

```
admin@EXALINK-FUSION> configure module Y serial-server port 1444
Serial server enabled for module Y on TCP port 1444
```

Once the serial server is enabled on the Fusion, various applications can be used to pass traffic to/from the connection. For example, telnet can be used on the remote host to login to the x86 processor module as follows:

```
$ telnet EXALINK-FUSION 1444
Trying 192.168.220.11...
Connected to EXALINK-FUSION.
Escape character is '^]'.

CentOS Linux 7 (Core)
Kernel 4.6.2-1.el7.elrepo.x86_64 on an x86_64

fusion_x86 login: root
Password:
Last login: Mon Aug  8 13:22:02 on tty1
[root@fusion_x86 ~]#
```

To disable the serial-server, the standard `no` form of the command can be issued:

```
admin@EXALINK-FUSION> config module Y no serial-server
Serial server disabled for module Y
```

Secondary Serial Port

The second serial port on the x86 processor module can be used to gain access to the CLI of the Fusion the processor module resides in.

To enable this, the Fusion must first be instructed to enable CLI access via this serial port:

```
admin@EXALINK-FUSION> configure module Y serial-console
Serial console access enabled for module Y
```

We can then use a standard terminal application on the x86 module to open the serial port and gain access to the CLI login on the Fusion:

```
[root@fusion_x86 ~]# picocom /dev/ttyS0 --baud 9600
picocom v1.7

port is        : /dev/ttyS0
flowcontrol    : none
baudrate is    : 9600
parity is      : none
databits are   : 8
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv
imap is        :
omap is        :
emap is        : crcrlf,delbs,

Terminal ready

EXALINK-FUSION login: admin
Password:
admin@EXALINK-FUSION>
admin@EXALINK-FUSION>
```

The `no` form of the command can be used to disable to console server:

```
admin@EXALINK-FUSION> config module Y no serial-console
Serial console disabled for module Y
```

Network Interfaces

ExaNIC

There is an onboard ExaNIC X40 embedded into the processor module. Whilst this is not in the same physical form factor as the ExaNIC X40 (ie it's not a PCIe card), the functionality and performance is exactly the same. The ExaNIC X40 provides 8 ultra low latency network interfaces to the host processor module, each of which can be configured for operation at 10G, 1G or 100M (with 40G firmware in development). The standard features of all ExaNICs apply, including hardware timestamping, flow steering and kernel bypass for low latency transparent acceleration of sockets applications.

When the Fusion detects an installed processor module, nine new ports are added to the system, for example Y1-Y9 assuming the module was installed into bay Y.

These interfaces/ports can be patched directly through to the front panel, or used to monitor & capture data flowing elsewhere throughout the Fusion. The ports can be displayed with the `show port` command, for example we can see the following on the Fusion CLI:

```
admin@EXALINK-FUSION> sh port Y*
Port Status  Description
---- ------- -----------
Y1   unused
Y2   unused
Y3   unused
Y4   unused
Y5   unused
Y6   unused
Y7   unused
Y8   unused
Y9   unused
```

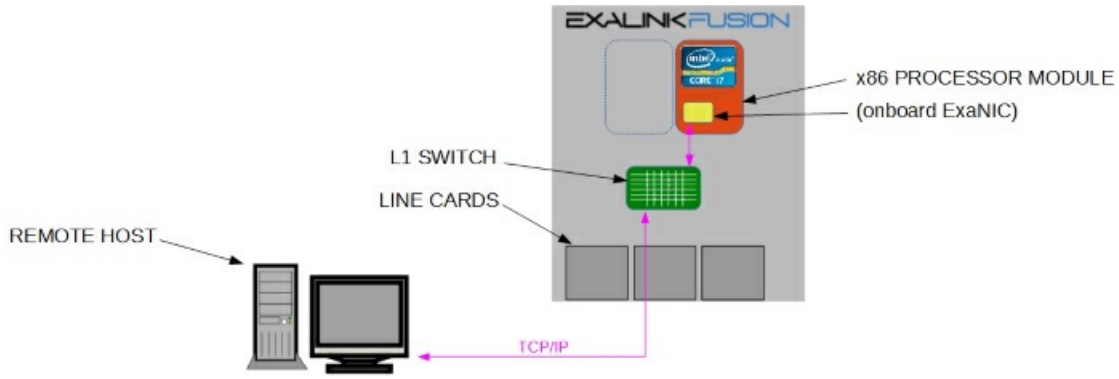
Port Y9 is the onboard LAN (described below), and Y1-Y8 are the 8 ExaNIC interfaces, from exanic0:0 through to exanic0:7. As with any other port, they can have an alias and description set, get patched to another port, be added to a mux object etc.

Note: Setting the speed of a port in this case just configures the signal recovery circuitry on the Fusion for optimal performance, it does not change the speed the ExaNIC is configured for on the Host - the user must ensure this is done independently.

In order to patch one of these ExaNIC ports to the front panel, the `patch` command should be used:

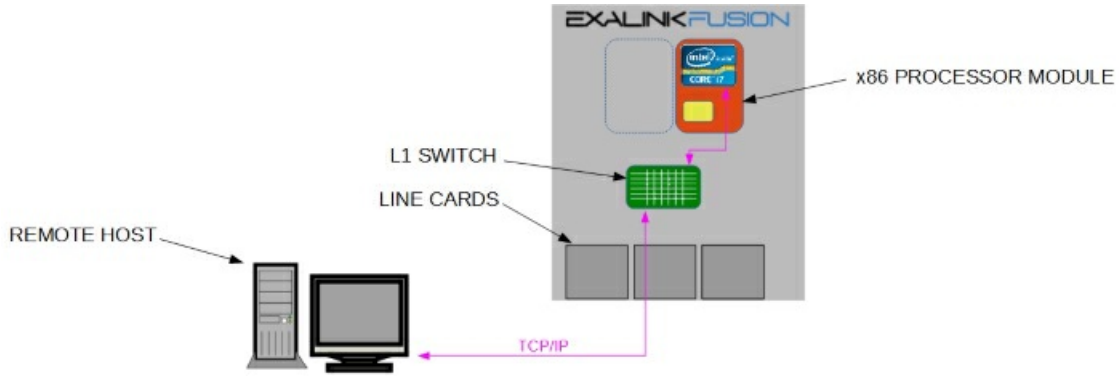
```
admin@EXALINK-FUSION> conf patch Y1 B2
Patch created between port "Y1" and port "B2"
```

This would allow, for example, a remote host to connect to the processor module as follows:



Onboard LAN

The motherboard within the processor module has an Intel(R) i217 Ethernet controller capable of running at 1G and 100M. This is exposed as port Y9, and would typically be patched out to allow for remote connection and management:



The onboard Ethernet controller supports Intel Active Management Technology (AMT), which allows for a number of handy IPMI /remote management capabilities. Refer to the section on AMT for further details.

Intel AMT

The onboard Ethernet controller supports Intel Active Management Technology (AMT), which allows for a number of handy IPMI /remote management capabilities. For example, you are able to establish a remote KVM over IP session with the processor, so you can interact with the system as a local user, for example reboot, enter the BIOS, install operating systems etc. Users might be familiar with similar technologies from Dell (iDRAC), HP (iLO) etc.

Exablaze ships the processor modules with AMT enabled with a password set which is `Exablaze1!`. AMT is only available via the onboard LAN (ie port Y9), not any of the exanic interfaces. It can be disabled via the KVM methods described below.

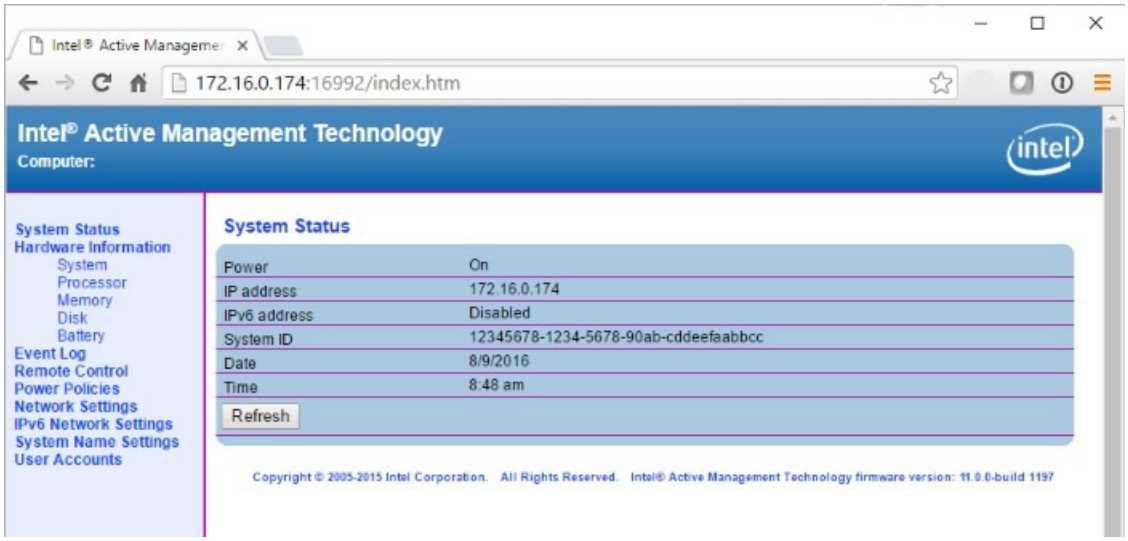
Intel AMT can be connected to via a number of different methods. It is configured to be assigned an IP address from a DHCP server on the network. Once the IP has been assigned, it can be discovered by searching for open ports on port 16992, for example:

```
$ sudo nmap --open -sT -p 16992 192.168.220.1/24
[sudo] password for donald:

Starting Nmap 6.40 ( http://nmap.org ) at 2016-08-09 18:17 AEST
Nmap scan report for 192.168.220.174
Host is up (-0.088s latency).
PORT      STATE SERVICE
16992/tcp open  amt-soap-http
MAC Address: 00:01:29:5D:EA:6B (DFI)

Nmap done: 256 IP addresses (122 hosts up) scanned in 8.14 seconds
```

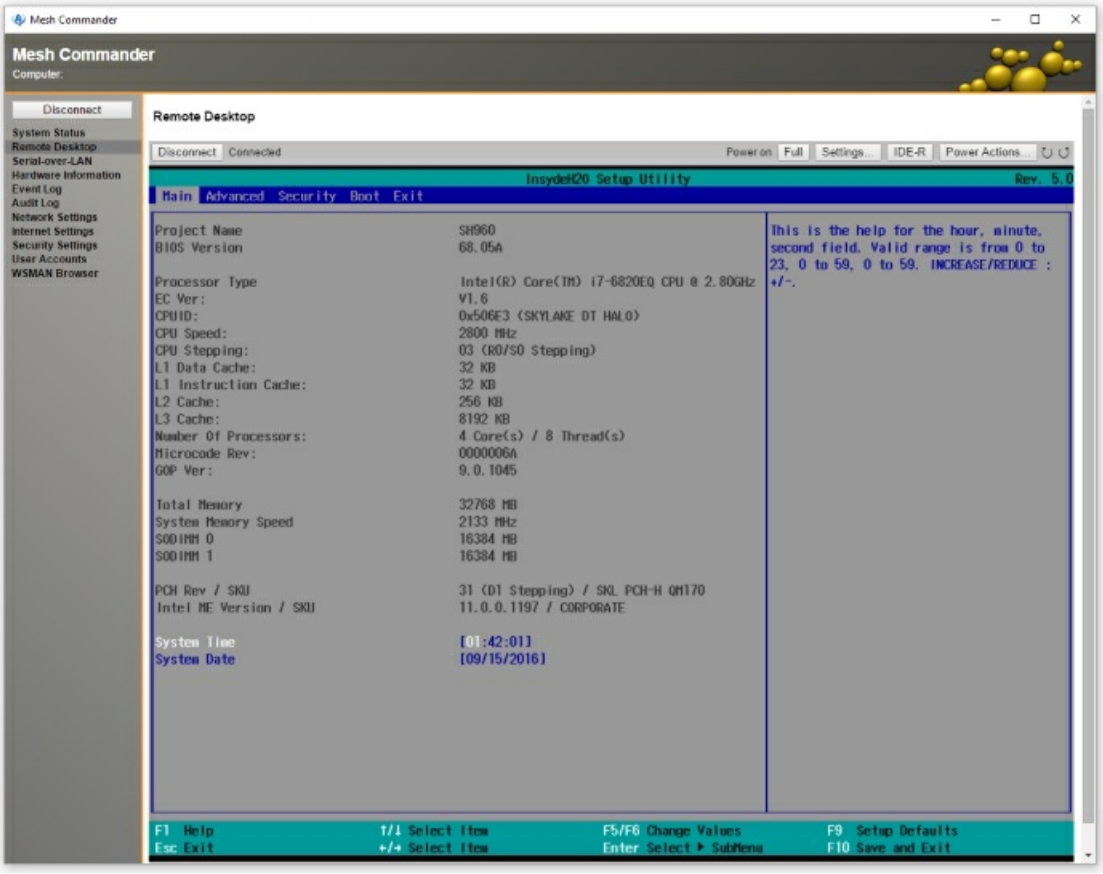
As can be seen above, the local DHCP server has assigned the module an IP address of 192.168.220.174. AMT has a fairly simple web interface for interacting with the system, so browsing to http://192.168.0.174:16992 and logging in as user `admin` with password `Exablaze1!` yields the following:



The default admin password can be changed by selecting User Accounts, and clicking the Change Admin... button.

A number of software utilities are available the allow for greater control over the system, including event & access logs, full KVM access (including into the BIOS), storage redirection, serial-over-lan etc.

Mesh Commander (compatible with connecting from Windows, Linux and Mac hosts) is known to work well, and the somewhat older Intel Platform Solution Manager is another option.



Editing the x86 Module BIOS using Mesh Commander from a remote PC

Intel AMT can be completely disabled in the BIOS, however users should be aware that once this is done, the only way to re-enable it is to remove the lid of the Fusion chassis and connect an HDMI monitor and USB keyboard to the connectors provided on the x86 module, and entering the MEBX extension within the BIOS menus.

Firmware Updates

Please see updating the firmware for firmware update instructions.

If you would like to be informed when new features or firmware releases are available, please add yourself to the ExaLINK Fusion mailing list here.

Version 1.13.0

Release date: 8th November, 2019

Build date: 8th November, 2019

Required support date: 20th October, 2019 or later

Fusion Download link: Download (156 MB, md5sum 5e9471be577bf1b8a8bfff273bd7404b)

Fusion HPT Download link: Download (125 MB, md5sum 25bf1de5bc2112364e0ca76cf889b173)

Changelog

- Link generation is on by default for ports that would otherwise be without a source of data (link-gen config option has been removed)
- Improved SNMP compatibility including additional MIBs
- Workaround for incorrectly coded Solid Optics SFP+ modules
- Workaround for incorrect "link up" detection depending on the nature of the remote end
- Add the SKU to `show version` output
- Add feature to optionally enable PPS input termination
- Add module power information to debug dump

Version 1.12.0 (Fusion HPT only)

Release date: 11th July, 2019

Build date: 10th July, 2019

Required support date: 1st July, 2019 or later

Fusion HPT Download link: Download (125 MB, md5sum ffda83eb151b454b55382957f7d6b591)

Changelog

- Link generation is on by default for ports that would otherwise be without a source of data (link-gen config option has been removed)
- There are now two HPT firmware images: `hpt` for 1G and 10G capture (replacing the old 10G only image) and `hpt-40g` for 10G and 40G capture.
- Link generation is now enabled automatically when a SFP is inserted on ports that have no output data. This replaces the configuration option.

Version 1.11.3

Release date: 14th May, 2019

Build date: 14th May, 2019

Required support date: 29th April, 2019 or later

Fusion Download link: Download (156 MB, md5sum 03f31969bb73a0e05dd0549ebe8b5deb)

Fusion HPT Download link: Download (86 MB, md5sum 75b3ff562bf39a36cf7ad4cbfb5943d5)

Changelog

- Add packet counters to debug dump output
- Fix FSN-824: Limit Fastmux output corruption when input link is unstable
- Fix FSN-892: Fix bug which could result in a missing VLAN tag in some IGMP queries
- Fix FSN-893: Fusion HPT timestamps may occasionally be invalid on some ports
- Fix FSN-899: Improve reliability of TX packet counters
- Fix FSN-911: Fix incorrect values for msgAuthoritativeEngineBoots and msgAuthoritativeEngineTime in SNMPv3 traps

Version 1.11.2

Release date: 14th February, 2019

Build date: 13th February, 2019

Required support date: 28th January, 2019 or later

Fusion Download link: Download (156 MB, md5sum 1cb76e98a009c652c05655dfe8592703)

Changelog

- Fastmux firmware: limit output link flapping in rare cases when input link is unstable

Version 1.11.1

Release date: 29th January, 2019

Build date: 29th January, 2019

Required support date: 14th January, 2019 or later

Fusion Download link: Download (157 MB, md5sum 89547503c1b3765cec837f1a055b1b3f)

Fusion HPT Download link: Download (85 MB, md5sum 9c30d075c5ef5429e2204d8ae5cd5883)

Changelog

- Add buffer usage statistic (Fusion HPT only)
- SFTP read-only access now requires `operator` role or higher

- Fix FSN-876: Fix bogus counter values in some situations where additional counters are unavailable
- Fix FSN-882: Fix bug which could result in frames being corrupted or dropped (Fusion HPT only)

Version 1.11.0

Release date: 18th December, 2018

Build date: 18th December, 2018

Required support date: 10th December, 2018 or later

Fusion Download link: Download (157 MB, md5sum f058300555401b5703edf9f6541393e4)

Fusion HPT Download link: Download (86 MB, md5sum d9bcd0e31f12e84faf61b3327b7cdd94)

Changelog

- Add support for SNMPv3
- Add support for CLI pipelines (ie for filtering with grep)
- Improve packet dump output, including ability to pipe into `tcpdump`
- Add additional counters, including breakdown of packets sent/received by frame size and type (unicast, multicast, etc).
- Add layer1 source port to `sh port` output where applicable
- Add support for changing the averaging parameters for PPS time sync (Fusion HPT only)
- Expose PPS time sync information available via CLI/API and InfluxDB (Fusion HPT only)
- Show PPS cable delay on `show timesync` output
- When `show log` uses the `contains` keyword, make the test case insensitive
- Add some clarifying text to the error condition resulting when a user tries to load Fusion HPT firmware to a Fusion and vice versa
- Fix FSN-868: Fix bug which could result in bit errors after a port is reconfigured
- Fix FSN-866: Fusion HPT timestamps could be incorrect after experiencing excessive link flaps
- Fix FSN-867: Improve robustness around power supply PMBus accesses
- Fix FSN-870: Incorrect statistics when connected to some network cards in 1G mode

Version 1.10.2 (Fusion HPT only)

Release date: 16th October, 2018

Build date: 16th October, 2018

Required support date: 17th September, 2018 or later

Fusion HPT Download link: Download (86 MB, md5sum b23ea1700228c78f997ed1875806d5ac)

Changelog

- Add support for devices which are now shipping with 32GB DRAM

Version 1.10.1

Release date: 4th October, 2018

Build date: 4th October, 2018

Required support date: 17th September, 2018 or later

Fusion Download link: Download (156 MB, md5sum bbd55d7b947ac2ae0bbfdc73c983da97)

Fusion HPT Download link: Download (86 MB, md5sum ef5168b4da1decadf2ce23d34f1324b5)

Changelog

- Fix FSN-853: CPLD update process may fail, affecting line card detection, LEDs, and PPS functionality

Version 1.10.0

Release date: 21st September, 2018

Build date: 21st September, 2018

Required support date: 17th September, 2018 or later

Fusion Download link: Unavailable (156 MB, md5sum a880a5b8e395276668c74e8b4a6eb70e)

Fusion HPT Download link: Unavailable (86 MB, md5sum b01c84093fb5a32a887c22377e012f87)

Changelog

- Add support for the Fusion HPT device with enhanced mirror object functionality
- Add support for link generation on ports
- Add support for configuring a cable delay/time offset for PPS inputs.
- Add support for setting system timezone.
- Add "end" command to drop out of all modal states
- The keyword "interface" (or int) can be used in place of "port" throughout the system
- Add debug command to set the minimum speed fans will run at
- Reduce the target temperature of x86 modules from 50C to 40C
- Fix FSN-798: Resolve high CPU load if FPGA module firmware function is changed whilst FPGA is still being configured
- Fix FSN-799: Automatically run optimize command after FPGA firmware function is changed to improve allocation
- Fix FSN-847: In some circumstances, VLAN tags may not be applied to mux objects in fast-vlan mode

Known issues

#	Issue	Description
FSN-853	CPLD update failure	CPLD update process may fail, affecting line card detection, LEDs, and PPS functionality upon *power cycling* after upgrade to 1.10.0

Version 1.9.0

Release date: 23nd March, 2018

Build date: 23nd March, 2018

Required support date: 1st March, 2018 or later

Download link: Download (157 MB, md5sum 2aa6b8bcc7d3ecdc35a2c4a024818afa)

Changelog

- Add support for time series statistics logging to InfluxDB
- Add support for auto configuration
- Allow selective blocking of traffic from one port to another port in switch objects
- SHA-256 and SHA-512 HMACs are now supported for ssh connections
- Show management interface link status and MAC address on "show management address" command output
- Fix FSN-747: "Large" mux objects when running fastmux firmware may have link issues passing traffic, particularly at elevated temperatures.

Version 1.8.0

Release date: 9th March, 2018

Build date: 9th March, 2018

Required support date: 1st February, 2018 or later

Download link: Download (157 MB, md5sum f8a6c8a2d06b5034db84b08f520659cf)

Changelog

- Reduce aggregation latency of fastmux firmware down to 39ns
- Expose user access to the underlying Linux BASH shell
- Add support for user python scripting and CRON jobs
- Add support for tunable DWDM SFP modules
- Report an error if a valid license is not found
- Preliminary support for QSFP line cards
- Expand info added into the debug dump to assist support
- Clarify warnings regarding port speed mismatch and port allocation errors
- Fix FSN-652: Report the correct timestamp mode in keyframes
- Fix FSN-494: Improve robustness around module initialization at power-up

Known issues

#	Issue	Description
FSN-747	"Large" mux objects when running fastmux firmware may have link issues passing traffic, particularly at elevated temperatures.	

Version 1.7.0

Release date: 17th October, 2017

Build date: 17th October, 2017

Required support date: 1st September, 2017 or later

Download link: Download (155 MB, md5sum 7757e77697725a68d517bbe63ebf95f1)

Changelog

- Check the end of support date in the license against the required support date
- Add router objects as a way to configure layer 3 functionality - for now, this is only used for configuring BGP
- Add BGP support to allow the Fusion to connect to a remote BGP peer and advertise routes (note that no routing is done by the Fusion)
- Add *no unknown-unicast* command to control where packets go on a failed MAC table lookup
- Improve handling of line card removal or insertion whilst device is in operation
- Reduce the frequency of "CDR interrupt stuck" warning messages
- Power down line cards if there is a critical temperature alert
- Increase MTU to 1599 bytes on mux and switch firmware
- Reduce mux latency on fastmux firmware
- Fix FSN-573: PPS distribution to internal x86 module
- Fix FSN-539: Add extra information to error messages for line card failures
- Fix FSN-678: Glitches in system time when using GPS time synchronization and the exalinkd management process is busy
- Fix FSN-680: Avoid potential deadlock when restarting services

Version 1.6.3

Release date: 5th September, 2017

Build date: 1st September, 2017

Download link: Download (155 MB, md5sum 29b52e470807e419ff1a9bf5062326d3)

Changelog

- Add support for multiple mux objects when running fastmux firmware
- Fix FSN-642: Max MTU size was previously 1527, now updated such that 1528 byte long frames pass through FPGA

Version 1.6.2

Release date: 27th July, 2017

Build date: 26th July, 2017

Download link: Download (147 MB, md5sum 6b760eacd32ebcfa61e9e42766611275)

Changelog

- Add support for updated protocol used in inbuilt GPS receiver

Version 1.6.1

Release date: 13th July, 2017

Build date: 6th July, 2017

Download link: Download (147 MB, md5sum b3d540fb15aa8bb8f499497a2ffbfbf8)

Changelog

- Various bug fixes relating to the fastmux firmware
- Fixed a bug where the chassis ID and port ID subtype fields in transmitted LLDP frames were incorrect
- Fixed a bug where the non-volatile log could be unreadable
- Fix FSN-592: Fastmux firmware now supports line rate aggregation

Version 1.6.0

Release date: 20th April, 2017

Build date: 7th April, 2017

Download link: Download (141 MB, md5sum a5c9859f4704aaa77512028338c85171)

Changelog

- Reduced minimum latency for 10G aggregation down to 49ns port-port when using fastmux firmware
- Add support for latest model Line Cards that support packet statistics and limited packet capture
- Add support for adding SSH keys for user login via public key authentication
- Add support for updating via http/s
- Add support for pressing "?" to get help for available command syntax, removed "help" as a command
- Add support for adding management interface name servers
- Add support for pinging hosts via name, not just IP
- Add support for exposing port statistics via SNMP - this requires an update to the MIB
- Extend license file format, a single lincense file can now be used for all devices
- Change license key format, now includes "end of support date" and a human readable description
- Show the end of support date in the output of "show version"
- Restrict Fusions from accepting a firmware update that's released after the end of support date for that Fusion
- Add cause for last reset to non-volatile log
- Add command to show non-volatile log on CLI
- Add command to disable MAC address learning
- Improve clarity for some CLI warning messages
- Increased maximum session timeout to 1 hour
- Updated underlying ptpd version used for time syncronization
- Add internal module serial numbers to the startup log when available
- Add support for Rev3 Internal Module FPGA boards, aka "KU115-04"
- When there's a fan fault, log which fan has failed
- Add extra detail to the log regarding change of power supply state, eg insertion, removal, power restored, etc
- **JSON API CHANGE**: The presence or absence of a port stat is now indicated by whether that particular field is present or not in the JSON object
- Fix FSN-191: Fan speeds could incorrectly reported as 0 on the CLI
- Fix FSN-461: Include power supply info in the debug dump
- Fix FSN-478: Malformed JSON API requests could cause internal process to restart
- Fix FSN-470: Command completion for module names is slow
- Fix FSN-517: Packets could be corrupted after going through mirror object with timestamping enabled
- Fix FSN-531: Damaged SFP's could lock up parts of the system

- Fix FSN-594: TACACS+ logins can gain direct access to shell

Known issues

#	Issue	Description
FSN-592	Line rate traffic support for the fastmux	The fastmux firmware image cannot sustain full line rate on a single port (i.e. minimum IFG).

Version 1.5.0

Release date: 5th September, 2016

Build date: 5th September, 2016

Download link: Download (163 MB, md5sum 2d7d70067ad6f9a828f7f8d98a2d73c0)

Changelog

- Reduce latency of switch and mux firmware images
- Add support for production x86 modules
- Add command to clear port counters
- Add command to clear bitfile when custom FPGA image is used
- Log tamper events to non-volatile log
- Improve robustness to SNMP responder
- Hostname is now reset to factory default after erasing config
- Clarified error message when unable to allocate upstream ports
- Disabled ports are now displayed in running-config
- Dont show latency stats for ports that dont have this capability
- Allow PTP domain numbers up to 127
- Users with insufficient roles are now prevented from entering config modal state. NB this was not previously a security issue as these users could not effect a change after entering the config state, but it's now clearer as they can't enter the config state in the first place
- Fix FSN-358: 1G mirror output stalls with timestamp FCS mode.

Known issues

#	Issue	Description
FSN-478	Malformed JSON requests can cause server crash	Malformed JSON requests to the API can cause a crash of the internal exalinkd server, resulting in a restart of this server. NB this would not reboot the system, just restart the underlying server application.

Version 1.4.2

Release date: 14th June, 2016

Build date: 8th June, 2016

Download link: Download (163 MB, md5sum 9da8fcd2685fa4bee278a944837efab8)

Changelog

- Fix FSN-406: Can't add multiple ports to a mirror object.
- Fix FSN-404: Link flapping can result in concatenated packet when going from 1G to 10G.

Known issues

#	Issue	Description
FSN-358	1G mirror output stals with timestamp FCS mode	When all ports for a mirror object are configured for 1G, enabling timestamping with FCS can cause the mirror output to lock up under high load conditions. Note this issue does not affect 10G mirror outputs.

Version 1.4.1

Release date: 17th May, 2016

Build date: 6th May, 2016

Download link: Download (163 MB, md5sum b3f571c554abce5f8e639bb6ad3c1d1f)

Changelog

- Add support for multiple NTP servers
- Add support to periodically poll the TACACS+ server for changes in user roles/permissions
- Improved power supply communication robustness

Known issues

#	Issue	Description
FSN-406	Can't add multiple ports to a mirror object	When a mirror object is created, if the output port is assigned first the number of input/source ports for that object is restricted. A temporary workaround is to add input ports before adding the output port.
FSN-404	Link flapping can result in concatenated packet when going from 1G to 10G	Mux objects with VLANs enabled, where there is a rate conversion from 1G to 10G and the 1G side of the link flaps (specifically during the middle of a frame), can result in concatenated frames being sent out 10G ports with a recalculated, and valid FCS for the concatenated frame.
FSN-358	1G mirror output stalls with timestamp FCS mode	When all ports for a mirror object are configured for 1G, enabling timestamping with FCS can cause the mirror output to lock up under high load conditions. Note this issue does not affect 10G mirror outputs.

Version 1.4.0

Release date: 7th April, 2016

Build date: 4th April, 2016

Download link: Download (163 MB, md5sum e9bd3239f43aff536f4a1788a682aa43)

Changelog

- Added support for Fusion chassis that include inbuilt GPS Receiver
- Added support for Fusion chassis that are able to output and input PPS

- Added commands to erase running-config
- Added command to do a delayed reboot
- Added commands to add/remove/manage local users with different roles
- Added *license erase* event to logs
- Added more status info on timesync when using PTP
- Added port latency stats for switch firmware (up to 4 ports)
- Improved PTP timesync accuracy
- Improved sanity checking for PTP and NTP parameters
- Improved sanity checking when setting port aliases
- Fix FSN-355: Issues with mirror objects and fcs-append mode on switch firmware

Known issues

#	Issue	Description
FSN-358	1G mirror output stalls with timestamp FCS mode	When all ports for a mirror object are configured for 1G, enabling timestamping with FCS can cause the mirror output to lock up under high load conditions. Note this issue does not affect 10G mirror outputs.

Version 1.3.0

Release date: 11th March, 2016

Build date: 11th March, 2016

Download link: Download (163 MB, md5sum 0df623f80aea5c9c4d9c61a954ce4f06)

Changelog

- Added licensing requirement to implement switch objects
- Added port latency statistics (currently only supported on mux firmware)
- Added support for Xilinx XVC server, used for customer FPGA development on Fusion
- Added support for setting power on/off for FPGA modules when running custom firmware, plus other custom FPGA enhancements
- Switch objects now support up to 48 total VLAN-enabled ports (was 16)
- Port allocator now supports 47 downstream ports on a single mux object (was 44)
- Packet count stats available for layer 1 objects (eg patch), when an FPGA module is installed
- LLDP receive now available for layer 1 objects (eg patch), when an FPGA module is installed
- Added command to clear learned MAC address table entries from an object
- Added *config optimize* command to optionally defragment port allocation in the FPGA, potentially improving latency
- Can now display summary of known LLDP neighbors for all ports at once
- Added support for new timestamping format - FCS append
- Added *show uptime* command
- Internal hardware RTC (real time clock) now updated by PTP
- Preliminary support for Fusion Internal x86 Module
- Add code to power down internal modules on critical temperature alert to prevent possible damage
- Debug dump changes - now generate a tarball containing running config and system logs
- JSON-RPC API is now available over WebSocket
- Fix FSN-327: CLI crash on *show port* for layer 1 only systems
- Fix FSN-347: SFP+ ports not being reconfigured properly with media change

- Fix FSN-338: Packet corruption with 1G mirror output or 1G to 1G VLAN enabled ports.
- Fix FSN-56: Corrupt retransmitted checksum when received checksum is invalid in VLAN enabled object
- Fix FSN-331: Timestamps value can be latched incorrectly under high load.

Known issues

#	Issue	Description
FSN-358	1G mirror output stalls with timestamp FCS mode	When all ports for a mirror object are configured for 1G, enabling timestamping with FCS can cause the mirror output to lock up under high load conditions. Note this issue does not affect 10G mirror outputs.
FSN-355	Issues with mirror objects and fcs-append mode on switch firmware	When running switch firmware, mirror outputs with timestamping enabled in fcs-append mode have a valid FCS, but no timestamp inserted.

Version 1.2.0

Release date: 18th December, 2015

Build date: 16th December, 2015

Download link: Download (162 MB, md5sum 8d557bbf201a987b9aa579a7e7884e58)

Changelog

- Add support for LLDP on ports that are part of mux/switch objects
- Bridge filtered addresses are now blocked through mux/switch objects
- Increased range of allowable VLAN IDs up to 4094
- Added support for virtual ports
- Added *ping* command
- Added *set time* command
- SFP RX/TX power now also shown in dBm
- Added support for FPGA module serial server in prep for release of FPGA dev kit
- Added more information into debug dump (top, ps, df)
- Added confirmation prompt for *reboot* command

Known issues

#	Issue	Description
FSN-327	CLI crash when showing ports on layer 1 only systems	Performing a "show port" on a system without an FPGA module can result in a CLI crash where the user is logged out. Note the underlying system remains unaffected.
FSN-338	Packet corruption with 1G mirror output or 1G to 1G VLAN enabled ports	When all ports are configured in 1G mode in a mux and VLAN tagging is enabled, or when a mirror output is set to 1G, the transmitting port may send some corrupt packets.
FSN-347	SFP+ ports not being reconfigured properly with media change	After a port is brought up, a change of media (eg SR optics to twinax) does not reconfigure the port for optimal signal performance. This could result in packet errors. Until 1.3.0 is released, short term fixes include rebooting the system, or remove the port from the object and re-add after insertion of the desired media.
FSN-56	Corrupt retransmitted checksum when received checksum is invalid in VLAN enabled object	VLAN enabled objects modify the received frame to insert, remove or modify the received VLAN tag. As part of this process the Fusion calculates a new frame check sequence (FCS) for the transmitted packet, which can result in a previously corrupt packet having an FCS that indicates the packet is OK. This issue will be fixed in 1.3.0.
FSN-331	Timestamps are not monotonic	With a small gap between packets the timestamping logic on the ingress to each port may latch a timestamp in which the low order bits are incorrect.

Version 1.1.1

Release date: 16th November, 2015

Build date: 16th November, 2015

Download link: Download (161 MB, md5sum e5644b91c23e014dba576057d4037817)

Changelog

- Fix issue 253 - Running config incorrectly applied to mux object in raw mode

Version 1.1.0

Release date: 13th November, 2015

Build date: 13th November, 2015

Download link: Unavailable (161 MB, md5sum 858806bfd97880ea6203fa21274a7661)

Changelog

- Introduce initial support for VLANs on switch and mux objects (support for VLANs 1 through 511)
- Added TACACS+ support for authentication, authorization and accounting
- Added management interface access control
- Expanded the amount of information that can be obtained via SNMP, for example load averages
- Expanded the number of SNMP traps that are sent on various events
- Expanded the number of internal events that are logged, for example link up/down
- Update event log to be more closely aligned with internal API, rather than client command line
- Login banner is now displayed pre-login
- Services are now restarted when the management IP address or hostname changes

- Added support for post-login MOTD banner
- Added RX FCS error counter for mux and switch ports
- Added link up/down counters on FPGA module
- Command line session timeout is now user configurable
- Added support for enabling/disabling ports, whilst leaving their use in objects untouched
- Reduced alert thresholds for onboard temperature sensors
- Update CLI show config output for IGMP changes
- Show in output of "show mac-address-table" if MAC address entry lock status is wrong
- Show static multicast groups in "show igmp groups" command
- Add commands to manipulate static multicast groups
- Clarifications/corrections to help command text
- Fix issue 217 - Static MAC entries could be overwritten
- Fix issue 241 - System log allowed to grow too large before rolling over
- Fix issue 243 - Add validity check to specified port used for SNMP

Version 1.0.8.1

Release date: 16th October, 2015

Build date: 16th October, 2015

Download link: Download (154 MB, md5sum 289ae21951b2fc9cffd3156fc1c8ebac)

Changelog

- Add support for FPGA modules with production (XCKU115-2FFVD1924C) silicon. Previous firmware releases only supported -ES chips.

Version 1.0.8

Release date: 16th September, 2015

Build date: 16th September, 2015

Download link: Download (154 MB, md5sum 339596bc6efcd944353639c208b84484)

Changelog

- Fix issue 198 - FPGA module sometimes in invalid state during rapid power cycling
- Fix issue 202 - Occasional packet loss when under high traffic load
- Fix issue 203 - IGMP messages could be ignored at high rates

Version 1.0.7

Release date: 21st August, 2015

Build date: 20th August, 2015

Download link: Download (154 MB, md5sum 27c9be548fabdb4631ccd5a17726afd1)

Changelog

- Sending of SNMP notifications on boot up and on power loss
- Command line sessions are now automatically logged out after 10 minutes of inactivity
- Customizable login banner
- Added command to monitor the status of time synchronization using NTP or PTP
- Enabled API for configuring the ExaLINK Fusion using JSON-RPC over HTTP
- Fix issue 106 - Port ranges do not work in mux/switch object

Known issues

#	Issue	Description
202	Occasional packet loss when under high traffic load	This problem was seen when the Fusion is transmitting a large amount of traffic out of a single port

Version 1.0.6

Release date: 31st July, 2015

Build date: 31st July, 2015

Download link: Download (154 MB, md5sum 9b2c440b0442a0f1cc308a6f995135d9)

Changelog

- SNMP read-only support
- Added commands for viewing system log in the CLI
- Fix issue 146 - Packet corruption with small back-to-back packets
- Fix issue 148 - 10GbE transmitter sends non-standard IPG

Known issues

#	Issue	Description
106	Port ranges do not work in mux/switch object	Trying to use port ranges (for example, A1-A5) to add ports to a mux or switch object does not work.

Version 1.0.5

Release date: 17th July, 2015

Build date: 17th July, 2015

Download link: Download (154 MB, md5sum d918425737352e67446b7aaff94afbfb)

Changelog

- Expanded IGMP functionality including configurable handling of unknown multicast traffic and fast-leave support.
- Optimized fan speed profile to reduce noise and increase fan MTBF
- Fix issue 121 - Limited IGMP config settings
- Fix issue 122 - Limited IGMP visibility
- Fix issue 125 - Missed received frames with certain 10G peers

Known issues

#	Issue	Description
106	Port ranges do not work in mux/switch objects	Trying to use port ranges (for example, A1-A5) to add ports to a mux or switch object does not work.
146	Packet corruption with small back-to-back packets	When a single port receives many small packets back-to-back at maximum line rate, some packet corruption or loss may occur. This issue is most notable when the receive interpacket gap is very small.
148	10GbE transmitter sends non-standard IPG	The 10GbE transmitter will return to idle for longer than necessary between packets. This reduces maximum throughput, particularly for small packets. Under particularly high throughputs this may result in dropped packets.

Version 1.0.4

Release date: 10th July, 2015

Build date: 9th July, 2015

Download link: Download (154 MB, md5sum 392736acb0afc7f735b0ee16d476e9fb)

Changelog

- Fix issue 124 - Invalid source address on IGMP queries

Known issues

#	Issue	Description
106	Port ranges do not work in mux/switch object	Trying to use port ranges (for example, A1-A5) to add ports to a mux or switch object does not work.
121	Limited IGMP config settings	IGMP version, timeout, and other variables cannot currently be configured by end-users.
122	Limited IGMP visibility	Currently, end users have only limited visibility into IGMP state. (Manually inspecting MAC address tables gives some insight but is a limited workaround.)
125	Missed received frames with certain 10G peers	Under high throughput and when connected to transmitters that produce small interframe gaps (IFG), some packets can be lost.

Version 1.0.3

Release date: 8th July, 2015

Build date: 8th July, 2015

Download link: Download (154 MB, md5sum 398e1a364d7f09219a0e35321a75aa73)

Changelog

- Fix issue 101 - Showing large MAC address tables no longer causes CLI to hang

- Initial release of support for IGMP snooping.

Known issues

#	Issue	Description
106	Port ranges do not work in mux/switch object	Trying to use port ranges (for example, A1-A5) to add ports to a mux or switch object does not work.
121	Limited IGMP config settings	IGMP version, timeout, and other variables cannot currently be configured by end-users.
122	Limited IGMP visibility	Currently, end users have only limited visibility into IGMP state. (Manually inspecting MAC address tables gives some insight but is a limited workaround.)
124	Invalid source address on IGMP queries	In certain cases, IGMP queries are sent with invalid source addresses. Specifically this occurs if IGMP is enabled on a pre-existing switch object

Version 1.0.2

Release date: 18th June, 2015

Build date: 17th June, 2015

Download link: Download (154 MB, md5sum c92e04f5730d70ac9c1071f10434baa5)

Changelog

- Fix issue 100 - switch firmware now supports timestamping.
- Change for switch firmware that helps with link stability.

Known issues

#	Issue	Description
106	Port ranges do not work in mux/switch object	Trying to use port ranges (for example, A1-A5) to add ports to a mux or switch object does not work.
101	Showing large MAC address tables causes CLI to hang	Showing very large MAC address tables causes CLI to hang. This causes no other side effects, however the user will need to terminate the session and log back into the device.

Version 1.0.1

Release date: 11th June, 2015

Download link: Download (154 MB, md5sum 34115e84d39b4ddd8b1c51523fa9f6bf)

Changelog

- Added "fcs-compat" timestamping mode, for Arista compatible timestamping mode.
- Software build date now shown in "show version".
- Implemented "show time" command
- Implemented "show fan-speed" command
- Fix for mux firmware that includes better link stability.

- Fix for mux firmware that fixes issue #99 - large frame sent out of mirror output port if keyframe sent.
- Both "show running-config" and "show startup-config" now available in root mode

Known issues

#	Issue	Description
100	Timestamping not supported in switch firmware	Switch firmware function does not support timestamping. Workaround is to use mux firmware for timestamping.
101	Showing large MAC address tables causes CLI to hang	Showing very large MAC address tables causes CLI to hang. This causes no other side effects, however the user will need to terminate the session and log back into the device.

Version 1.0.0

Release date: 4th June, 2015

Changelog

- Initial release version.

Known issues

#	Issue	Description
99	Mirror keyframes cause issues	Keyframes on mirror port can cause a large packet to be sent out of the mirror output port.
100	Timestamping not supported in switch firmware	Switch firmware function does not support timestamping. Workaround is to use mux firmware for timestamping.

Specifications

Type	Specification	
Physical	Dimensions	19.0x14.2x1.8""
	Rack Mount	1RU
Operating Conditions	Temperature	32 to 104F (0 to 40C)
	Voltage	(AC) 90-265V, (DC) 40-72V
	Frequency	50/60 Hz (for AC input)
	Power consumption	150W typical, 250W max
Connectivity	Switch Ports	48 x 10GbE SFP+ modules (MSA compliant)
	Management	1x RS232 9600,N,8,1 serial
		1x 10/100M Ethernet
	Timing	1x PPS timing port
	For firmware upgrade/file transfer	USB

Commands

Commands can be sent to the ExaLINK Fusion using a command line interface, of which the available commands are listed below.

Note

The CLI has a modal structure. User permissions/roles are not tested on entry to various operating modes, however they are tested when a command is executed within a mode. For example, all users can enter **config port** however only users with appropriate roles can execute a command such as setting the speed for that port.

Note

Commands that are preceded by `config` or `debug` do not need that prefix at the start when in the corresponding mode.

Name	Description	Min reqd. role
show version	Show software and hardware version.	guest
show uptime	Display how long the Fusion has been active for.	guest
show diagnostic	Show hardware diagnostics.	guest
show temperature	Show temperature measurements.	guest
show fan-speed	Show the speed (RPM) of every connected fan module.	guest
show power-supply	Show power supply diagnostics.	guest
show power	Show voltage and current measurements.	guest
show users	Display list of all users.	guest
config password <user>	Set password for the specified user. **Note:** a user will be able to change their own password, but not anyone else's without sufficient permission. If an argument is not provided to the command, it will default to changing the current user's password.	admin
exit	Exit current mode.	guest
logout	Terminate the current session.	guest
configure	Enters configuration mode (identical to `config`). **Note:** Refer to the note at the top of this table re entering modes.	N/A
ping <ip_address>	Use the management interface to ping a given IP address.	guest
show time	Show the current time.	guest
show port [<port>]	Displays port details or a summary of all ports.	monitor
port <port>	Select a port to operate on.	N/A

Name	Description	Min reqd. role
config port <port>	Enters configuration mode for specified port.	N/A
config port <port> alias <alias>	Assign an alias to a port.	user
config port <port> no alias	Remove an alias from a port.	user
config port <port> description <description>	Assign a description to port.	user
config port <port> no description	Remove a description from a port.	user
config port <port> speed <mbps>	Configure the speed of a port.	user
config port <port> autoneg <enable|disable>	Enable or disable autonegotiation on a port.	user
config port <port> link-gen	Enable simulated link generation on a port.	user
config port <port> no link-gen	Disable simulated link generation on a port.	user
config port <port> enable	Enable a port.	user
config port <port> disable	Disable a port.	user
config port <port> reset counters	Reset a port's packet statistics.	user
port <port> show latency	Generate a latency histogram of the port.	monitor
port <port> show latency last <n>	Generate a latency histogram of the port using the last *n* bins.	monitor
show lldp neighbors	Display a table of all neighboring devices found via LLDP.	monitor
config port <port> lldp transmit	Enable LLDP on the specified port.	user
config port <port> no lldp transmit	Disable LLDP on the specified port.	user
show patch	Display all currently patched ports.	monitor
config patch <port1> <port2>	Create a new patch between two ports.	operator
config no patch <port1> <port2>	Delete an existing patch between two ports.	operator
show tap	Display all currently tapped ports.	monitor
config tap <src_port> <destination>	Create a tap of incoming traffic from src_port to the destination.	operator
config no tap <src_port> <destination>	Remove a specified tap.	operator
config tap output <src_port> <destination>	Create a tap of outgoing traffic from src_port to the destination.	operator
config no tap output <src_port> <destination>	Remove a specified output tap.	operator
show mux	Show a summary of mux objects.	monitor
config mux <name> show	Show config for the selected mux.	operator
config mux <name>	Create/edit mux object with specified name.	operator

Name	Description	Min reqd. role
config no mux <name>	Delete mux object with specified name.	operator
config mux <name> mode <mode>	Set the operating mode of a mux object.	operator
config mux <name> port <port>	Add a downstream port on the selected mux.	operator
config mux <name> no port <port>	Remove port from the selected mux.	operator
config mux <name> port upstream <port>	Assign an upstream port for the selected mux.	operator
config mux <name> no port upstream <port>	Remove upstream port from the selected mux.	operator
config mux <name> igmp <flood-unknown mrouter | snooping | static-group>	Set the IGMP operating mode for the selected mux.	operator
config mux <name> <vlan-enable>	Enable VLAN operation for the selected mux.	operator
config mux <name> mac-address-table static <mac address> port <port>	Add a static route for the selected mux.	operator
show switch	Show a summary of switch objects.	monitor
config switch <switch> show	Show configuration of selected switch.	monitor
config switch <name>	Create/edit a switch object with specified name.	operator
config no switch <name>	Delete specified switch object.	operator
config switch <switch> port <port>	Add port to the selected switch.	operator
config switch <switch> no port <port>	Remove port from selected switch.	operator
show mirror	Show a summary of mirror objects.	monitor
config mirror <name> show	Show config for the selected mirror object.	monitor
config mirror <name>	Create/edit a mirror with specified name.	operator
config no mirror <name>	Delete a specified mirror.	operator
config mirror <name> output <port>	Assign port to be an output for selected mirror.	operator
config mirror <name> no output <port>	Remove output port from the selected mirror.	operator
config mirror <name> port <port>	Assign port as input for selected mirror.	operator
config mirror <name> no port <port>	Remove port as input for selected mirror.	operator
config mirror <name> switch <name>	Add switch object to selected mirror.	operator
config mirror <name> no switch <name>	Remove a switch object from the selected mirror.	operator

Name	Description	Min reqd. role
config mirror <name> timestamp <none fcs | fcs-compat | append | append-compat >	Configure timestamping on the selected mirror.	operator
config mirror <name> no timestamp	Disable timestamping on selected mirror.	operator
show log [last <n>] [reverse] [level <priority> [contains <string>] [all <id> ...]	Display the Fusion's log of activity.	monitor
show remote-logging	Displays remote-logging configuration and status.	monitor
config remote-logging target <type> <address> <facility> <level>	Add remote logging of specified setting.	admin
config remote-logging reset	Clear out and disable remote logging.	admin
config remote-logging enable	Enable remote logging.	admin
config remote-logging disable	Disable remote logging.	admin
debug dump	Create a .tar file detailing the current running state of the Fusion.	operator
config show snmp	Show SNMP configuration.	monitor
config no snmp	Clear out all SNMP settings and disable SNMP.	admin
config snmp disable	Disable SNMP, while retaining current settings.	admin
config snmp enable	Enable SNMP.	admin
config snmp read community <password>	Set the shared phrase used to identify a readonly community.	admin
config snmp contact <contact>	Set the contact details that will be displayed in SNMP object sysContact.	admin
config snmp location <location>	Set the location details that will be displayed in SNMP object sysLocation.	admin
config snmp port <port>	Set the SNMP port, or set to zero for the default.	admin
config no snmp trap	Clear out all SNMP Trap settings and disable SNMP Trap.	admin
config snmp trap disable	Disable the sending of SNMP Traps.	admin
config snmp trap enable	Enable the sending of SNMP Traps.	admin
config snmp trap target <address> <community>	Set an additional address to which SNMP Traps will be sent, using the given community string.	admin
show telnet	Display telnet status.	monitor
config telnet enable	Enable telnet access.	admin
config telnet disable	Disable telnet access.	admin
show tacacs	Show TACACS+ configuration.	monitor
config no tacacs	Reset all TACACS+ configuration.	admin
config tacacs disable	Disable TACACS+ authentication, authorization and accounting.	admin
config tacacs enable	Enable TACACS+ authentication, authorization and accounting.	admin

Name	Description	Min reqd. role
config tacacs server <server>...	Set list of TACACS+ servers to be used.	admin
config tacacs secret <secret>	Set the TACACS+ shared secret, used for encryption.	admin
config tacacs service <service>	Set the service name, used by the remote TACACS+ server.	admin
config tacacs timeout <seconds>	Set the TACACS+ request timeout.	admin
config tacacs periodic <seconds>	Configure TACACS+ to periodically check authorization levels of users.	admin
config tacacs no periodic	Disable periodic authorization level checks in TACACS+.	admin
config tacacs accounting <level>	Configure TACACS+ accounting to be disabled, standard or verbose.	admin
show running-config	Display the current configuration settings.	monitor
show startup-config	Display the startup configuration settings.	monitor
config copy running-config startup-config	Save running configuration to startup.	admin
config copy startup-config running-config	Load startup configuration.	admin
config erase startup-config	Erase all of the saved configuration files.	admin
config erase running-config <all management | module | data-plane>	Erase some or all of the currently running configuration.	admin
config optimize	Attempt to reallocate switch resources in a more optimized way.	operator
config license reload	Reload a previously entered license.	admin
config license set	Enter a license key for the Fusion.	admin
config license erase	Erase a current set license key for the Fusion.	admin
config time set <time>	Set the current time on the Fusion. **Note:** Accepted format is [YYYY-MM-DD] HH:MM:SS	operator
show timesync	Display the current time synchronization settings.	monitor
config timesync ntp <server>... ptp <domain> pps <edge> [cable-delay <delay>] ntp+pps <server>... [<edge>] [cable-delay <delay>] ptp+pps [<domain> [<edge>]] [cable-delay <delay>] gps [antenna-delay <nanoseconds>] none	Configure time synchronization settings.	operator
config timesync output pps <rising|falling>	Set the PPS output to either the rising or falling edge.	operator
config no timesync	Disable time synchronization.	operator
config session-timeout <seconds|default>	Set automatic logout to the specified number of seconds or a default value.	admin

Name	Description	Min reqd. role
show auto-config	Shows whether auto-config is enabled or not.	guest
config auto-config <enabled|disabled>	Enable or disable auto-configuration.	admin
config hostname <hostname>	Configure hostname of the device.	admin
show management address	Display IPv4 address for the management interface.	guest
config management address dhcp	Configure management interface using DHCP.	admin
config management address static <address> <netmask> [gateway]	Configure a static IPv4 address on the management interface.	admin
show management name-server	Shows the name server(s) on the management interface.	guest
config management name-server <name server> ...	Configure a name server on the management interface.	admin
show management access-list	Show access control rules.	monitor
config management access-list allow <address range>...	Extend access control rules to allow connections from the supplied IPv4 address ranges.	admin
config management access-list deny <address range>...	Extend access control rules to deny connections from the supplied IPv4 address ranges.	admin
config no management access-list	Reset all access control rules.	admin
config module <X|Y> function <switch mux custom>	Load the switch or mux bitfile onto the FPGA module, or set it to accept a custom bitfile.	operator
config module <X|Y> power <onoff>	Enable or disable power to the internal module.	operator
config module <X|Y> fpga bitstream <bitstream>	Set a custom bitstream onto FPGA module.	operator
config module <X|Y> fpga reconfigure	Reconfigure the FPGA module according to the custom bitfile.	operator
config module <X|Y> fpga xvc-server port <port>	Enable the XVC server for the FPGA on the specified module.	operator
config module <X|Y> no fpga xvc-server	Disable a currently running XVC server for the FPGA on the specified module.	operator
config module <X|Y> serial-server <tcp-port>	Enable the serial server of the specified module.	operator
config module <X|Y> no serial-server	Disable a currently running serial server on the specified module.	operator
show services	Show which services are running.	monitor
config http enable	Enable http access for JSON RPC API.	admin
config http disable	Disable http access for JSON RPC API.	admin
config update file <tar file>	Update system from an uploaded file.	admin

Name	Description	Min reqd. role
config update usb <tar file>	Update system from a file on an attached USB drive.	admin
config update tftp <server> <tar file>	Download and update system from a file from a tftp server.	admin
reboot	Restart the device.	operator
reboot in [<hh>:]<mm>	Restart the device after the specified amount of time has passed.	operator
reboot cancel	Cancel a previously scheduled reboot.	operator

FPGA Firmware Differences

This page lists some of the major functional differences between the various FPGA functions on the Fusion and Fusion HPT products.

ExaLINK Fusion

This table is based on ExaLINK Fusion firmware version 1.9.0.

Feature	Function Mux	Function Switch	Function Fastmux
mux objects	Unlimited *	Unlimited	4
switch objects	No	Unlimited with switch license **	No
patch/tap objects	Yes (<5ns)	Yes (<5ns)	Yes (<5ns)
mirror objects	Yes	Yes	No
10G port support	Yes	Yes	Yes
1G port support	Yes	No (planned)	No
VLAN support	Yes	Yes	No
Num upstream mux ports	4	Unlimited	4
Num mirror objects with timestamping output		4	0
Minimum cut-through latency (+/- ~3ns)	92ns (down-up, 10G-10G, no VLAN)	86ns (mux-object, 10G-10G, no VLAN)	39.00ns (+/- 0.25ns)
	102ns (up-down layer2 mode, 10G-10G, no VLAN)	95ns (switch object, 10G-10G, no VLAN)	
Maximum cut-through latency (+/- ~3ns)	92ns (down-up, 10G-10G, no VLAN)	120ns (48 port mux object, 10G-10G, no VLAN)	55.50ns (+/-0.25ns)
	102ns (up-down layer2 mode, 10G-10G, no VLAN)	126ns (48 port switch object, 10G-10G, no VLAN)	
FPGA on-chip packet buffering	>=1.2MByte	>=1.5MByte	>=64KByte
Software sending messages through dataplane(e.g. LLDP, BGP, IGMP)	Yes	Yes	No

* There is a limitation of 4 upstream ports with mux firmware. Please refer to Mux objects for further details.

** A switch license is required to create switch objects. A switch license is included when ordering the Fusion in the Switch SKU.

ExaLINK Fusion HPT

This table is based on ExaLINK Fusion HPT firmware version 1.12.0.

Feature	Funtion hpt	Function hpt-40g
mux objects	No	No
switch objects	No	No
mirror objects	1	1
patch/tap objects	Yes (<5ns)	Yes (<5ns)
Max number of 1G input ports	40	0
Max number of 10G input ports	40	24
Max number of 40G input ports	0	4*
10G output ports	1-8 (User selectable)	1-8 (User selectable)
Deep buffering	32GByte DDR4	32GByte DDR4
Timestamp precision	<100ps	<100ps

* 40G inputs only supported on QSFP Line Cards ports.

Introduction

```
# Python examples use the jsonrpc-requests library
# (https://pypi.org/project/jsonrpc-requests/)

from jsonrpc_requests import Server

fusion = Server("http://myfusion/jsonrpc")
```

This page describes the Fusion JSON-RPC interface. This interface allows you to programatically control the ExaLINK Fusion from a remote host via the Ethernet management interface. This interface is enabled from the command line using:

`configure http enable`

Commands are then sent via HTTP POST requests to the Fusion web interface, accessible via the Fusion network address. Authentication is performed by the Fusion server by first sending a login request containing a username and password. After authentication is complete, the Fusion web server will open and maintain an HTTP session, through which JSON RPC requests can be made. The address for requests to the RPC interface is:

`http://<mgmt-addr>/jsonrpc`

The remainder of this document lists the RPC calls that can be made, the arguments you must provide with them, and the return values.

Authentication

Clients must be authenticated prior to being able to issue commands. Once authenticated an HTTP session is opened, with a cookie stored at the client end. It is the user's responsibility to store this cookie and provide it with subsequent requests. The following commands handle this authentication.

Note that an idle timeout of 15 minutes applies to all sessions.

Management Interface

login

```
>>> fusion.login(username="admin", password="admin")
True
```

```
// Request
{
    "method": "login",
    "params": {
        "username":"admin",
        "password":"admin"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Log in to the ExaLINK Fusion. The username and password must be supplied as strings to this command. Returns success if supplied credentials are valid, error otherwise

PARAMETERS

Field	Type	Description
username	string	Username to log in as
password	string	Password for user

RESULT

True if successful

---

logout

```
>>> fusion.logout()
True
```

```
// Request
{
    "method": "logout",
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Log out of the ExaLINK Fusion

PARAMETERS

None

RESULT

True if successful

---

whoami

```
>>> fusion.whoami()
{u'username': u'admin'}
```

```
// Request
{
    "method": "whoami",
    "id": 1
}

// Response
{
    "result": {
        "username": "admin"
    },
    "id": 1
}
```

Returns the username of the current session

PARAMETERS

None

RESULT

The username of the current user

set_password

```
>>> fusion.set_password(username="admin", password="mynewpassword")
True
```

```
// Request
{
    "method": "set_password",
    "params": {
        "password":"mynewpassword"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method can be used to change the user's password

PARAMETERS

Field	Type	Description
username	string	Username of the user whose password will be changed
password	string	The new password

RESULT

True if successful

---

get_hostname

```
>>> fusion.get_hostname()
{u'hostname': u'EXALINK-FUSION'}
```

```
// Request
{
    "method": "get_hostname",
    "id": 1
}

// Response
{
    "result": {
        "hostname": "EXALINK-FUSION"
    },
    "id": 1
}
```

This method returns the hostname of the ExaLINK Fusion

PARAMETERS

None

RESULT

The current hostname

---

set_hostname

```
>>> fusion.set_hostname(hostname="myFusion")
True
```

```
// Request
{
    "method": "set_hostname",
    "params": {
        "hostname":"myFusion"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1

}
```

This method sets the hostname of the ExaLINK Fusion

PARAMETERS

Field	Type	Description
hostname	string	The new hostname

RESULT

True if successful

---

get_management_address_ipv4

```
>>> fusion.get_management_address_ipv4()
{u'mode': u'static', u'netmask': u'255.255.255.0', u'gateway': u'', u'address': u'172.16.0.153'}
```

```
// Request
{
    "method": "get_management_address_ipv4",
    "id": 1
}

// Response
{
    "result": {
        "mode": "static",
        "netmask": "255.255.255.0",
        "gateway": "",
        "address": "172.16.0.153"
    },
    "id": 1
}
```

Get information on the IPv4 management address configuration of the ExaLINK Fusion

PARAMETERS

None

RESULT

Field	Type	Description
mode	string	IPv4 configuration mode: `static` or `dhcp`
netmask	string	Network mask in dot-decimal notation
gateway	string	Gateway address in dot-decimal notation
address	string	IPv4 address in dot-decimal notation

Field	Type	Description

set_management_address_ipv4

```
>>> fusion.set_management_address_ipv4(mode="static", static={"address":"172.16.0.153","netmask":"255.255.255.0", "gateway":""})
True
```

```
// Request
{
    "method": "set_management_address_ipv4",
    "params": {
        "mode": "static",
        "netmask": "255.255.255.0",
        "gateway": "",
        "address": "172.16.0.153"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method can be used to set the IPv4 management address configuration of the ExaLINK Fusion

PARAMETERS

Field	Type	Description
mode	string	IP address assignment mode to use: `static` or `dhcp`

In the case of setting a static IP address, additional required parameters are:

Field	Type	Description
address	string	IPv4 address to use in dot-decimal notation
netmask	string	Network mask in dot-decimal notation
gateway	string	Gateway address in dot-decimal notation

RESULT

True if successful

get_version

```
>>> fusion.get_version()
{u'mainboard': {u'serial': u'EXALINK-FUSION', u'type': u'EBFSN-02'}, u'line_card': {u'A': {u'type': u'LC10G-03'}, u'C': {u'type': u'LC10G-03'}, u'B': {u'type': u'LC10G-03'}}, u'module':
 {u'X': {u'firmware': {u'hash': u'f9120423', u'datecode': 1465301675, u'type': u'mux'}, u'type': u'KU115-03'}}, u'software': {u'date': u'2016-06-08 11:40:15 +1000 (f305efa)', u'version'
: u'1.4.2'}}
```

```
// Request
{
    "method": "get_version",
    "id": 1
}

// Response
{
    "mainboard": {
        "serial": "EXALINK-FUSION",
        "type": "EBFSN-02"
    },
    "line_card": {
        "A": {
          "type": "LC10G-03"
        },
        "C": {
          "type": "LC10G-03"
        },
        "B": {
          "type": "LC10G-03"
        }
    },
    "module": {
        "X": {
          "firmware": {
              "hash": "f9120423",
              "datecode": 1465301675,
              "type": "mux"
          },
          "type": "KU115-03"
        }
    },
    "software": {
        "date": "2016-06-08 11:40:15 +1000 (f305efa)",
        "version": "1.4.2"
    },
    "id": 1
}
```

This method returns version details for the ExaLINK fusion along with a summary of installed hardware.

PARAMETERS

None

RESULT

Field	Type	Description
mainboard	array	A list of strings describing the Mainboard in the Fusion
serial	string	The serial number of this Mainboard
type	string	The type of Mainboard installed into this Fusion
line_card	array	A list of strings describing the Line Cards installed in the Fusion
module	array	A list of the modules describing the Internal Modules installed in the Fusion and their firmware setting
software	array	A list containing the version and date of creation of the current software installed in the Fusion

get_session_timeout

```
>>> fusion.get_session_timeout()
600
```

```
// Request
{
    "method": "get_session_timeout",
    "id": 1
}

// Response

{
    "result": 600,
    "id": 1

}
```

This methods returns the current time (in seconds) of idle while logged in before being logged out automatically.

PARAMETERS

None

RESULT

An integer of the current session timeout value (in seconds).

---

set_session_timeout

```
>>> fusion.set_session_timeout(600)
True
```

```
// Request
{
    "method": "set_session_timeout",
    "params": ["600"],
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Set the maximum idle time (in seconds) before automatic logging out of the Fusion.

PARAMETERS

Field	Type	Description
timeout	integer	The number of seconds of idle activity before logging the user out

RESULT

True if successful.

---

check_user_permission

```
>>> fusion.check_user_permissions("user")
2
```

```
// Request
{
    "method": "check_user_permissions",
    "params": ["admin"],
    "id": 1
}

// Response
{
    "result": 2,
    "id": 1
}
```

PARAMETERS

Field	Type	Description
user	string	Name of the account to check the permission level of

RESULT

Field	Type	Description
access_level	integer	Value of the access level of the account

get_management_access_list

```
>>> fusion.get_management_access_list()
{u'deny': [], u'allow': []}
```

```
// Request
{
    "method": "get_management_access_list",
    "id": 1
}

// Response
{
    "result": {
        "allow": [ ],
        "deny": [ ]
    },
    "id": 1
}
```

PARAMETERS

None

RESULT

Field	Type	Description
allow	array	A list of IPs allowed to access the management interface
deny	array	A list of IPs not allowed to access the management interface

set_time_sync_mode

```
>>> fusion.set_time_sync_mode(ntp_config={"server":"172.160.0.147"}, mode="ntp")
True
```

```
// Request
{
    "method": "set_time_sync_mode",
    "params": {
        "ntp_config": {
            "server": "172.160.0.147"
        },
        "mode": "ntp"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

PARAMETERS

Field	Type	Description
ntp_config	object	An array holding the data required for NTP configuration
server	string	The IP address of the NTP device to base time off
mode	string	The timesync method to be used on the Fusion

RESULT

True if successful

---

get_time_sync_status

```
>>> fusion.get_time_sync_status()
{u'pps': {u'signal_detected': True, u'last_time': 1506489752}, u'gps': {u'time_accuracy': 7, u'position': {u'latitude': -33.8672081, u'altitude': 93.153, u'longitude': 151.2053761}, u'fix': True, u'num_satellites': 12}}
```

```
// Request
{
    "method": "get_time_sync_output",
    "id": 1
}

// Response
{
    "pps": {
        "signal_detected": true,
        "last_time": 1506489752
    },
    "gps": {
        "time_accuracy": 7,
        "position": {
            "latitude": -33.8672081,
            "altitude": 93.153,
            "longitude": 151.2053761
        },
        "fix": true,
        "num_satellites": 12
    }
}
```

This method returns the details of the timesync mode currently in use on the Fusion (in the event of multiple modes used in tandem, both will be returned in the same query).

PARAMETERS

None

RESULT

Based on the timesync modes selected the return value will vary.

PPS return value is as below:

Field	Type	Description
signal_detected	bool	Whether or not the Fusion has acquired a PPS signal currently or not.
last_time	float	The last time indicated via the PPS signal.

NTP return value is as below:

Field	Type	Description
server	string	The IP of the NTP server selected.
poll_interval	unsigned	The interval in seconds between server polls.
offset	double	The time offset on the NTP time.
stratum	unsigned	The stratum the NTP server currently belongs to.

PTP return value is as below:

Field	Type	Description
domain	unsigned	The PTP domain connected to in the network.
clock	array	Consists of entries with corresponding interface name fields, all contain their state variable (below).
state	string	FREERUN or otherwise indicated, dictates the status of the PTP clock.
adev	double	The Allan deviation for the clock on the specific interface.
master	string	The IP of the PTP master defined.
state	string	The PTP role of the Fusion.
offset	string	The PTP offset in the current system.

GPS return value is as below:

Field	Type	Description
time_accuracy	unsigned	The accuracy of the GPS timing.
position	array	The set of coordinates for the GPS signal (containing longitude, latitude and altitude values).
longitude	double	The longitudal coordinate of the device.
latitude	double	The latitudal coordinate of the device.
altitude	double	The altitude coordinate of the device.
fix	bool	The status of positional fix for the Fusion's GPS setting.
num_satellites	unsigned	The number of satellites found through the GPS signal.

get_time_sync_output

```
>>> fusion.get_time_sync_output()
{u'pps': {u'edge': u'rising'}}
```

```
// Request
{
    "method": "get_time_sync_output",
    "id": 1
}

// Response
{
    "result": {
        "pps": {
            "edge": "rising"
        }
    },
    "id": 1
}
```

PARAMETERS

None

RESULT

Field	Type	Description
edge	string	Either ‘rising’ or ‘falling’ based on edge being used for PPS output

set_time_sync_output_pps

```
>>> fusion.set_time_sync_output_pps()
True
```

```
// Request
{
    "method": "set_time_sync_output_pps",
    "params": {
        "enable": true,
        "edge": "rising"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

PARAMETERS

Field	Type	Description
enable	boolean	`True` to enable PPS output, `False` to disable
edge	string	Either `rising` or `failling`

RESULT

update_file

```
>>> fusion.update_file(tarball="exalink_fusion_1.1.1.tar")
True
```

```
// Request
{
    "method": "update_file",
    "params": {
        "tarball":"exalink_fusion_1.1.1.tar"
    },
    "id": 1
}
```

This method can be used to update the ExaLINK Fusion's firmware with a file already placed onto the Fusion via SFTP.

PARAMETERS

Field	Type	Description
tarball	string	Filename of the tarball to be used in the update

RESULT

None

---

update_tftp

```
fusion.update_tftp(server="172.16.0.160", file="exalink_fusion_1.1.1.tar")
True
```

```
// Request
{
    "method": "update_tftp",
    "params": {
        "server":"172.16.0.160",
        "file":"exalink_fusion_1.1.1.tar"
    },
    "id": 1
}
```

This method updates the Fusion from a file taken from a TFTP server.

PARAMETERS

Field	Type	Description
server	string	Address of the device hosting the TFTP service
file	string	Filename of the tarball on the TFTP server

RESULT

None

---

update_usb

```
fusion.update_usb(tarball="exalink_fusion_1.1.1.tar")
True
```

```
// Request
{
    "method": "update_usb",
    "params": {
        "tarball":"exalink_fusion_1.1.1.tar"
    },
    "id": 1
}
```

This method is used to update the fusion via a connected USB storage device.

PARAMETERS

Field	Type	Description
tarball	string	Filename of the tarball on the storage device

RESULT

True if successful

---

reboot

```
>>> fusion.reboot()
True
```

```
// Request
{
    "method": "reboot",
    "id": 1
}
```

This method reboots the ExaLINK Fusion

PARAMETERS

None

RESULT

None

---

schedule_reboot

```
>>> fusion.schedule_reboot(after=200)
True
```

```
// Request
{
    "method": "schedule_reboot",
    "params": {
        "after": 200
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to schedule a reboot after a number of seconds.

PARAMETERS

Field	Type	Description
after	integer	Time (in seconds) to begin reboot sequence (starting from after receiving the call)

RESULT

True if successful.

set_port_lldp

```
>>> fusion.set_port_lldp(port="A1", transmit=1)
True
```

```
// Request
{
    "method": "set_port_lldp",
    "params": {
        "port": "A1",
        "transmit": 1
    },
    "id" : 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to enable or disable transmission of LLDP frames on a specified port.

PARAMETERS

Field	Type	Description
port	string	Name of the port to modify LLDP status on
transmit	boolean	Set to `1` to enable LLDP frame transmission, `0` to disable

RESULT

True if successful

get_port_lldp_neighbors

```
>>> fusion.get_port_lldp_neighbors(port="A1")
[{u'lldp': True, u'neighbors': [{u'system_name': u'fusion2', u'system_description': u'ExaLINK Fusion', u'chassis_id': [100, 63, 95, 128, 25, 128], u'port_id_subtype': 5, u'management_address': [{u'address_subtype': 1, u'interface': 0, u'interface_subtype': 1, u'address': [172, 16, 0, 152]}], u'chassis_id_subtype': 4, u'port_id': [65, 49]}], u'port': u'A2'}]
```

```
// Request
{
    "method": "get_port_lldp_neighbors",
    "params": "A1",
    "id": 1
}

// Response
{
    "result": {
        "lldp": true,
        "neighbors": [
            {
                "system_name": "fusion2",
                "system_description": "ExaLINK Fusion",
                "chassis_id": [
                    100,
                    63,
                    95,
                    128,
                    25,
                    128
                ],
                "port_id_subtype": 5,
                "management_address": [
                    {
                        "address_subtype": 1,
                        "interface": 0,
                        "interface_subtype": 1,
                        "address": [172, 16, 0, 152]
                    }
                ],
                "chassis_id_subtype": 4,
                "port_id": [
                    65,
                    49
                ]
            }
        ]
    },
    "port": "A2",
    "id": 1
}
```

This method returns the LLDP entries for the port from the Fusion's LLDP agent.

PARAMETERS

Field	Type	Description
port	string	The port to request LLDP information from

RESULTS

Field	Type	Description
port	string	Port the following information references
lldp	boolean	True if LLDP is set to transmit
mgmt_addr	array	See below
neighbor	array	See below

The management address field contains some or all of the following values:

Field	Type	Description
address_subtype	number	Value determines the format of the address field (MAC address or otherwise)
address	string	The address based on the substype
interface_subtype	number	The type of description of the management interface
interface	string	The name of the interface, based on the interface_subtype

Field	Type	Description
oid	number	The OID of the system

A neighbor in the Fusion's LLDP agent will be returned with some or all of these fields:

Field	Type	Description
chassis_id_subtype	number	The category of chassis ID used in the LLDP entry
chassis_id	string	Corresponding string to the subtype of ID being provided
port_id	string	Name of the port connected to the neightbor
port_description	string	Description of the port connected to the neighbor
system_name	string	Name of the neighbor system
system_description	string	Description of the neighbor system
system_capabilities	string	All the capabilities of a neighbor system
enabled_capabilities	string	Capabilities of the neighbor currently enabled
management_address	string	Address of the neighbor

get_license

```
>>> fusion.get_license()
{u'serial': u'EXAFSN-A-00198', u'support_end': 1577836800, u'features': [u'switch'], u'expiry': 1577836800}
```

```
// Request
{
    "method": "get_license",
    "id": 1
}

// Response
{
    "result": {
        "serial": "EXAFSN-A-00198",
        "support_end": 1577836800,
        "features": ["switch"],
        "expiry": 1577836800
    },
    "id": 1
}
```

This method gets the licensed features and serial number of the Fusion.

PARAMETERS

None

RESULT

Field	Type	Description
serial	string	The serial number of the Fusion being queried
support_end	integer	The date support for this Fusion will end, as a Unix timestamp in seconds
features	array	A set of strings detailing the licensed features on the Fusion.
expiry	integer	The date any features being evaluated will end, as a Unix timestamp in seconds

set_license

```
>>> fusion.set_license("license_string")
True
```

```
// Request
{
    "method": "set_license",
    "params": ["my_license_key_here"],
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to set the license of the Fusion

PARAMETERS

Field	Type	Description
key	string	The license key to be given to the Fusion

RESULT

True if successful.

---

get_running_config

```
>>> fusion.get_running_config("patch")
{u'version': 1, u'patch': [[u'B1', u'C15'], [u'B2', u'Y9'], [u'B3', u'C5'], [u'B5', u'C13']]}
```

```
// Request
{
    "method": "get_running_config",
    "params": "patch",
    "id": 1
}

// Response
{
    "patch": [
        [
            "B1",
            "C15"
        ],
        [
            "B2",
            "Y9"
        ],
        [
            "B3",
            "C5"
        ],
        [
            "B5",
            "C13"
        ]
    ],
    "id": 1
}
```

This method returns information from the specified part of the running configuration.

PARAMETERS

Field	Type	Description

Field	Type	Description
category	string	The specific aspect of the running configuration to obtain information for.

Available categories: *mgmt_net, session, users, roles, ports, patch, tap, module, switch, mux, mirror, time, service, tacacs*

RESULT

The result format is based on the argument provided to the method, consult other sections for sections like TACACS or Modules.

---

get_startup_config

```
>>> fusion.get_startup_config("patch")
{u'version': 1, u'patch': [[u'B1', u'C15'], [u'B2', u'Y9'], [u'B3', u'C5'], [u'B5', u'C13']]}
```

```
// Request
{
    "method": "get_startup_config",
    "params": "patch",
    "id": 1
}

// Response
{
    "patch": [
        [
            "B1",
            "C15"
        ],
        [
            "B2",
            "Y9"
        ],
        [
            "B3",
            "C5"
        ],
        [
            "B5",
            "C13"
        ]
    ],
    "id": 1
}
```

This method returns information from the specified part of the startup configuration.

PARAMETERS

Field	Type	Description
category	string	The specific aspect of the running configuration to obtain information for.

Available categories: *mgmt_net, session, users, roles, ports, patch, tap, module, switch, mux, mirror, time, service, tacacs*

RESULT

The result format is based on the argument provided to the method, consult other sections for sections like TACACS or Modules.

---

erase_running_config

```
>>> fusion.erase_running_config(all=False, management=False, module=False, data_plane=True)
True
```

```
// Request
{
    "method": "erase_running_config",
    "params": {
        "all": false,
        "management": false,
        "data_plane": true,
        "module": false
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method erases parts of the running configuration based on the arguments specified

PARAMETERS

Field	Type	Description
all	boolean	`True` to erase all configuration on the Fusion, `False` to not erase.
management	boolean	`True` to erase all settings related to accessing the management interface, `False` to not erase.
data_plane	boolean	`True` to erase all settings related to object configuration, `False` to not erase.
module	boolean	`True` to erase all settings related to an internal module, `False` to not erase.

RESULT

True if successful.

---

erase_startup_config

```
>>> fusion.erase_startup_config()
True
```

```
// Request
{
    "method": "erase_startup_config",
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method will erase the startup configuration, restoring it to default settings. Unlike erase_running_config(), this will erase all of the config without needing to specify.

PARAMETERS

None

RESULT

True if successful

---

load_startup_config

```
>>> fusion.load_startup_config()
True
```

```
// Request
{
    "method": "load_startup_config",
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method will copy the current startup configuration to the running configuration. Note: This will reload the network settings, please wait some time before calling anymore methods.

PARAMETERS

None

RESULT

True if successful

---

save_startup_config

```
>>> fusion.save_startup_config()
True
```

```
// Request
{
    "method": "save_startup_config",
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method will copy the current running configuration to the starting configuration.

PARAMETERS

None

RESULT

True if sucessful

Diagnostics and Logging

read_current_sensors

```
>>> fusion.read_current_sensors()
{u'line_card_a': {u'current': 0.43359375, u'voltage': 12.1875}, u'line_card_c': {u'current': 0, u'voltage': 12.265625}, u'line_card_b': {u'current': 0.6171875, u'voltage': 12.1875}, u'module_y': {u'current': 0, u'voltage': 12.265625}, u'module_x': {u'current': 2.5078125, u'voltage': 12.1875}}
```

```
// Request
{
    "method": "read_current_sensors",
    "id": 1
}

// Response
{
    "result": {
        "line_card_a": {
            "current": 0.4609375,
            "voltage": 12.1875
        },
        "line_card_c": {
            "current": 0.44140625,
            "voltage": 12.2265625
        },
        "line_card_b": {
            "current": 0.97265625,
            "voltage": 12.1484375
        },
        "module_y": {
            "current": 0,
            "voltage": 12.2265625
        },
        "module_x": {
            "current": 2.73046875,
            "voltage": 12.1875
        }
    },
    "id": 1
}
```

This method can be used to return the reading from various voltage and current sensors within the ExaLINK Fusion.

PARAMETERS

None

RESULT

This will return an array of devices that are connected to the Fusion, their fields are as below:

Field	Type	Description
current	number	The current being drawn by the device
voltage	number	The voltage read at the device

---

read_power_supplies

```
>>> fusion.read_power_supplies()
{u'psu_0': {u'pin': 105, u'vout': 12.171875, u'pout': 97, u'iout': 8.125, u'vin': 239, u'temp_2': 31.5, u'model': u'DS460', u'iin': 0.4375, u'temp_1': 25.5, u'present': True, u'manufacturer': u'EMERSON'}, u'psu_1': {u'pin': 0, u'vout': 0, u'pout': 0, u'iout': 0, u'vin': 0, u'temp_2': 30.5, u'model': u'DS460', u'iin': 0, u'temp_1': 29, u'present': True, u'manufacturer': u'EMERSON'}}
```

```
// Request
{
    "method": "read_power_supplies",
    "id": 1
}

// Response
{
    "result": {
        "psu_0": {
            "pin": 85,
            "vout": 12.234375,
            "pout": 67,
            "iout": 5.875,
            "vin": 217.5,
            "temp_2": 31.5,
            "model": "DS460",
            "iin": 0.390625,
            "temp_1": 25,
            "present": true,
            "manufacturer": "EMERSON"
        },
        "psu_1": {
            "pin": 0,
            "vout": 0,
            "pout": 0,
            "iout": 0,
            "vin": 0,
            "temp_2": 24.5,
            "model": "DS460",
            "iin": 0,
            "temp_1": 23.5,
            "present": true,
            "manufacturer": "EMERSON"
        }
    },
    "id": 1
}
```

This method can be used to query status information from the Power Supplies

PARAMETERS

None

RESULT

This method returns two sets of results for both power supply slots in the Fusion.

Field	Type	Description
pin	number	Input power drawn by the power supply
vout	number	Output voltage from the power supply
pout	number	Output power from the power supply
iout	number	Output current from the power supply
vin	number	Input voltage to the power supply
temp_2	number	Internal temperature of the power supply
model	string	Model number of the power supply
iin	number	Input current drawn by the power supply
temp_1	number	External temperature of the power supply
present	boolean	True if power supply has been plugged into the appropriate slot
manufacturer	string	Manufacturer of the power supply

read_temperature_sensors

```
>>> fusion.read_temperature_sensors()
{u'crosspoint': [39.50, 40.41, 38.67, 37.8], u'module_x': [36, 41], u'mainboard': [29.625, 33], u'line_card_a': [37.4], u'line_card_c': [35.1], u'line_card_b': [31.6]}
```

```
// Request
{
    "method": "read_temperature_sensors",
    "id": 1
}

// Response
{
    "result": {
        "crosspoint": [39.50, 40.41, 38.67, 37.8],
        "module_x": [37, 41],
        "mainboard": [29.625, 33],
        "line_card_a": [37.4],
        "line_card_c": [35.1],
        "line_card_b": [31.6]
    },
    "id": 1
}
```

This method can be used to query the various temperature sensors within the ExaLINK Fusion

PARAMETERS

None

RESULT

This returns an array of devices in the Fusion, each with their temperature reading. NB some devices have multiple sensors.

---

read_fan_speeds

```
>>> fusion.read_fan_speeds()
{u'fan_3': 7108, u'fan_2': 7113, u'fan_1': 7155, u'fan_0': 7199}
```

```
// Request
{
    "method": "read_fan_speeds",
    "id": 1
}

// Response
{
    "result": {
        "fan_3": 7108,
        "fan_2": 7113,
        "fan_1": 7155,
        "fan_0": 7199
    },
    "id": 1
}
```

This method can be used to query the speed of the fan modules installed in the ExaLINK Fusion

PARAMETERS

None

RESULT

Returns a list of the four fans and their RPM.

get_port_latency_histogram

```
>>> fusion.get_port_latency_histogram(port="B1", interval=1)
```

```
// Request
{
    "method": "get_port_latency_histogram",
    "params": {
        "interval": 1,
        "port": "B1"
    },
    "id": 1
}

// Response
{
    "result": {
        "port": "B1",
        "data": [
            [
                [ 81, 86.714285714285708 ],
                3722813
            ]
        ]
    },
    "id": 1
}
```

Obtain a tabulated set of histogram data of latency for a specific port.

PARAMETERS

Field	Type	Description
port	string	The port to obtain latency statistics for
interval	integer	Class width of the histogram to obtain

RESULT

An array of sets of data, representing the percentile, latency number and packet count.

---

set_tacacs_servers

```
>>> fusion.set_tacacs_servers("172.16.0.13")
0
```

```
// Request
{
    "method": "set_tacacs_servers",
    "params": ["172.16.0.13"],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to set the server the TACACS is being hosted on.

PARAMETERS

Field	Type	Description
server	string	The IP address of the TACACS server

RESULT

`0` if successful.

---

set_tacacs_secret

```
>>> fusion.set_tacacs_secret("asdasdasdasd")
0
```

```
// Request
{
    "method": "set_tacacs_secret",
    "params": ["asdasdasdasd"],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to set the TACACS secret key to be used on the Fusion.

PARAMETERS

Field	Type	Description
secret	string	Key to be used to try and access the TACACS server

RESULT

`0` if succesfully set the key on the Fusion

---

set_tacacs_accounting

```
>>> fusion.set_tacacs_accounting()
0
```

```
// Request
{
    "method": "set_tacacs_accounting",
    "params": ["verbose"],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to set the SNMP field

PARAMETERS

Field	Type	Description
mode	string	The level of logging to be used on the TACACS server (`disabled`, `standard`, `verbose`)

RESULT

`0` if successful.

set_tacacs_timeout

```
>>> fusion.set_tacacs_timeout(100.0)
0
```

```
// Request
{
    "method": "set_tacacs_timeout",
    "params": [100.0],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to set the TACACS request timeout.

PARAMETERS

Field	Type	Description
timeout	float	Number (in seconds) for TACACS request timeouts (limited to 1 decimal place)

RESULT

`0` if successful.

set_tacacs_periodic

```
>>> fusion.set_tacacs_periodic(120)
0
```

```
// Request
{
    "method": "set_tacacs_periodic",
    "params": [120],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to set the time between periodic permission checking with the TACACS server and Fusion.

PARAMETERS

Field	Type	Description
time	integer	Time (in seconds) between regular checking of user permissions with the TACACS server and Fusion

RESULT

`0` if successful.

set_snmp_location

```
>>> fusion.set_snmp_location("Server room")
0
```

```
// Request
{
    "method": "set_snmp_location",
    "params": ["Server room"],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to set the SNMP field

PARAMETERS

Field	Type	Description
location	string	Name to be used in the SNMP Location field.

RESULT

`0` if successful.

---

set_snmp_contact

```
>>> fusion.set_snmp_contact("Admin")
0
```

```
// Request
{
    "method": "set_snmp_contact",
    "params": ["Admin"],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to set the SNMP Contact field

PARAMETERS

Field	Type	Description
contact	string	Name to be used in the SNMP Contact field.

RESULT

`0` if successful

---

set_snmp_read_community

```
>>> fusion.set_snmp_read_community("test-community")
0
```

```
// Request
{
    "method": "set_snmp_read_community",
    "params": "test-community",
    "id": 1
}

// Response
{
    "result": "0",
    "id": 1
}
```

Use this to set the community name to be used in the Fusion's SNMP setting.

PARAMETERS

Field	Type	Description
community	string	Name of the community name to be used in the SNMP setting.

RESULT

0 if successful.

---

set_snmp_port

```
>>> fusion.set_snmp_port()
0
```

```
// Request
{
    "method": "set_snmp_port",
    "params": ["161"],
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

PARAMETERS

Field	Type	Description
port	integer	Port number to communicate with the SNMP destination on

RESULT

`0` if successful

---

enable_service_tacacs

```
>>> fusion.enable_service_tacacs()
0
```

```
// Request
{
    "method": "enable_service_tacacs",
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to enable TACACS on the Fusion.

PARAMETERS

None

RESULT

`0` if successful.

---

enable_service_snmp

```
>>> fusion.enable_service_snmp()
0
```

```
// Request
{
    "method": "enable_service_snmp",
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to enable SNMP on the Fusion.

PARAMETERS

None

RESULT

`0` if successful.

---

enable_service_snmptrap

```
>>> fusion.enable_service_snmptrap()
0
```

```
// Request
{
    "method": "enable_service_snmptrap",
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method is used to enable SNMP traps targetting configured hosts.

PARAMETERS

None

RESULT

`0` if successful

---

disable_service_tacacs

```
>>> fusion.disable_service_tacacs()
0
```

```
// Request
{
    "method": "disable_service_tacacs",
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method disables TACACS on the Fusion (reverting to internal authentication).

PARAMETERS

None

RESULT

`0` if successful.

---

disable_service_snmp

```
>>> fusion.disable_service_snmp()
0
```

```
// Request
{
    "method": "disable_service_snmp",
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method disables SNMP on the Fusion.

PARAMETERS

None

RESULT

`0` if successful.

disable_service_snmptrap

```
>>> fusion.disable_service_snmptrap()
0
```

```
// Request
{
    "method": "disable_service_snmptrap",
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1
}
```

This method disables SNMP Traps on the Fusion.

PARAMETERS

None

RESULT

`0` if successful.

---

get_service_snmp

```
>>> fusion.get_service_snmp()
0
```

```
// Request
{
    "method": "get_service_snmp",
    "id": 1
}

// Response
{
    "result": {
        "service": "snmp",
        "enabled": true
    },
    "id": 1
}
```

This method is used to determine the status of SNMP on the Fusion.

PARAMETERS

None

RESULT

Field	Type	Description
service	string	Name of the service (i.e. SNMP)
enabled	boolean	`True` if enabled, `False` if not

---

set_snmptrap_target

```
>>> fusion.set_snmptrap_target()
0
```

```
// Request
{
    "method": "set_snmptrap_target",
    "params": {
        "community": "public",
        "address": "172.16.0.229"
    },
    "id": 1
}

// Response
{
    "result": 0,
    "id": 1

}
```

PARAMETERS

Field	Type	Description
community	string	Name of the community to be used for SNMP traps
address	string	The IP address to target for SNMP traps

RESULT

`0` if successful.

---

get_snmptrap_config

```
>>> fusion.get_snmptrap_config()
{u'to': [{u'community': u'public', u'address': u'172.16.0.220'}, {u'community': u'private-ro', u'address': u'172.16.0.121'}, {u'community': u'public', u'address': u'172.16.0.229'}, {u'community': u'public', u'address': u'172.16.0.229'}, {u'community': u'public', u'address': u'172.16.0.229'}], u'enabled': True}
```

```
// Request
{
    "method": "get_snmptrap_config",
    "id": 1
}

// Response
{
    "result": {
        "to":
            [
                {
                    "address": "172.16.0.220",
                    "community": "public"
                },
                {
                    "address": "172.16.0.121",
                    "community": "private-ro"
                },
                {
                    "address": "172.16.0.229",
                    "community": "public"
                }
            ],
        "enabled": true },
    "id": 1
}
```

Get current SNMP Trap configuration.

PARAMETERS

None

RESULT

Field	Type	Description
to	object	An array containing entries that consist of `address` and `community` (if present)
address	string	The IP address targetted for SNMP traps
community	string	The community being used on the host

get_messages

```
>>> fusion.get_messages()
{u'warning': [], u'error': []}
```

```
// Request
{
    "method": "get_messages",
    "id": 1
}

// Response
{
    "result": {
        "error": [ ],
        "warning": [ ]
    },
    "id": 1
}
```

Get a list of error and warning messages from the Fusion.

PARAMETERS

RESULT

Field	Type	Description
error	string	Any messages pertaining to any errors having occurred on the Fusion.
warning	string	Any messages pertaining to any warnings generated by the Fusion.

get_switch_stats

```
>>> fusion.get_switch_stats()
{u'used_memory': 0}
```

```
// Request
{
    "method": "get_switch_stats",
    "id": 1
}
// Response
{
    "result": {
        "used_memory": 0
    },
    "id": 1
}
```

Get switch statistics from the Fusion.

PARAMETERS

None

RESULT

Field	Type	Description
used_memory	integer	Packet buffer memory usage (Fusion HPT only)

debug_dump

```
>>> fusion.debug_dump()
debug/debug_info_20161021T032122.tar.gz
```

```
// Request
{
    "method": "debug_dump",
    "id": 1
}
// Response
{
    "result": "debug\/debug_info_20161021T032122.tar.gz",
    "id": 1
}
```

This method generates a debug file on the Fusion that can be retrieved via SFTP.

PARAMETERS

None

RESULT

The name and path to the debug file on the Fusion.

Port Status and Configuration

get_ports

```
>>> fusion.get_ports("A1")
[{u'data_rate': u'ethernet_10g', u'name': u'A1', u'sfp_type': u'auto', u'enabled': False, u'alias': u'myalias', u'address': u'643F5F000000', u'has_signal': True, u'autoneg': True, u'present': True, u'description': u''}]
>>>
>>>
>>> fusion.get_ports("A10","A11","B16")
[{u'lldp_transmit': False, u'data_rate': u'ethernet_10g', u'description': u'', u'in_use': False, u'sfp_type': u'auto', u'enabled': True, u'alias': u'', u'address': u'643F5F801999', u'has_signal': False, u'autoneg': True, u'present': False, u'name': u'A10'}, {u'lldp_transmit': False, u'data_rate': u'ethernet_10g', u'description': u'', u'in_use': False, u'sfp_type': u'auto', u'enabled': True, u'alias': u'', u'address': u'643F5F80199A', u'has_signal': False, u'autoneg': True, u'present': False, u'name': u'A11'}, {u'lldp_transmit': False, u'data_rate': u'ethernet_10g', u'description': u'', u'in_use': False, u'sfp_type': u'auto', u'enabled': True, u'alias': u'', u'address': u'643F5F8019AF', u'has_signal': True, u'autoneg': True, u'present': True, u'name': u'B16'}]
```

```
// Request
{
    "method": "get_ports",
    "params": [
        "A1"
    ],
    "id": 1
}

// Response
{
    "result": [
        {
            "data_rate": "ethernet_10g",
            "name": "A1",
            "sfp_type": "auto",
            "enabled": false,
            "alias": "myalias",
            "address": "643F5F000000",
            "has_signal": false,
            "autoneg": true,
            "present": false,
            "description": ""
        }
    ],
    "id": 1
}
```

This method can be used to get status information for one or more ports

PARAMETERS

This method can be provided with a list of port names as an array of strings. If no parameters are provided, information is returned for all ports.

RESULT

This method returns an array of objects with the following fields:

Field	Type	Description
name	string	The name of the port the data is being returned for
alias	string	The alias of the port
address	string	The MAC address of the port(s) being requested
description	string	The description of the port
data_rate	string	Port data rate, either `ethernet_10g` or `ethernet_1g`
sfp_type	string	Configured SFP type, either `auto` or `active` or `passive`
enabled	boolean	True if the port is enabled
autoneg	boolean	True if autonegotiation is enabled on the port
present	boolean	True if the SFP is present
has_signal	boolean	True if the SFP is not reporting loss-of-signal
in_use	boolean	True if the port is in use in the current configuration

set_port_alias

```
>>> fusion.set_port_alias(port="A1",alias="myalias")
True
```

```
// Request
{
    "method": "set_port_alias",
    "params": {
        "port": "A1",
        "alias": "myalias"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method can be used to set the alias for a port. The alias can be used on the ExaLINK Fusion command line to identify the port in place of the port name.

Note that the alias cannot be used to identify the port in API calls.

PARAMETERS

Field	Type	Description
port	string	The port you wish to set the alias for, in Exablaze standard format, eg `A1`
alias	string	The alias you wish to use for this port. Note only the alias can only consist of A-Z, a-z, 0-9, - and _ characters. An alias must not be a normal port name (eg `A1`). It must also not be a number, or a number with a single letter prefix and/or suffix

RESULT

True if successful

---

set_port_description

```
>>> fusion.set_port_description(port="A1",description="A verbose description of this port")
True
```

```
// Request
{
    "method": "set_port_description",
    "params": {
        "port": "A1",
        "description": "a verbose description of this port"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method can be used to set a description associated with a particular port.

PARAMETERS

Field	Type	Description
port	string	The port you wish to set the description for, in Exablaze standard format, eg `A1`
description	string	The description you wish to set for this port

RESULT

True if successful

---

get_port_sfp_info

```
>>> fusion.get_port_sfp_info(port="B1")
{u'vendor_pn': u'BN-CKM-SP-SR', u'date_code': u'120717', u'vendor_oui': [0, 23, 106], u'serial_encoding': 6, u'vendor_sn': u'AD1229A0112', u'nominal_bit_rate': 10300, u'connector_type':
 7, u'vendor_rev': u'G2.3', u'wavelength': 850, u'vendor_name': u'Blade Network', u'port': u'B1', u'transceiver_code': [16, 0, 0, 0, 0, 0, 0, 0]}
```

```
// Request
{
    "method": "get_port_sfp_info",
    "params": {
        "port": "B1"
    },
    "id": 1
}

// Response
{
    "result": {
        "vendor_pn": "BN-CKM-SP-SR",
        "date_code": "120717",
        "vendor_oui": [
            0,
            23,
            106
        ],
        "serial_encoding": 6,
        "vendor_sn": "AD1229A0112",
        "nominal_bit_rate": 10300,
        "connector_type": 7,
        "vendor_rev": "G2.3",
        "wavelength": 850,
        "vendor_name": "Blade Network",
        "port": "B1",
        "transceiver_code": [
            16,
            0,
            0,
            0,
            0,
            0,
            0,
            0,
            0
        ]
    },
    "id": 1
}
```

This method can be used to query the diagnostic interface on a SFP. The fields in the returned data closely follow the fields as defined in the SFP MSA. Please refer to the SFP MSA Specification for further details.

PARAMETERS

Field	Type	Description
port	string	The name of the port to be queried

RESULT

Field	Type	Description
port	string	Port name
vendor_oui	array	SFP vendor IEEE company identifier, as an array of 3 bytes
vendor_name	string	SFP vendor name
vendor_pn	string	Part number provided by SFP vendor
vendor_rev	string	Revision level provided by SFP vendor
connector_type	integer	Connector type code provided by SFP
transceiver_code	array	Transceiver code provided by SFP

Field	Type	Description
serial_encoding	integer	Serial encoding used by SFP
nominal_bit_rate	integer	Nominal bit rate reported by SFP
wavelength	integer	Laser wavelength reported by SFP

get_port_sfp_diagnostics

```
>>> fusion.get_port_sfp_diagnostics(port="A1")
{u'vcc': 3.2734, u'temperature': 34.81640625, u'tx_bias': 3.791, u'tx_power': 0.6558, u'rx_power': 0.641, u'port': u'A1'}
```

```
// Request
{
    "method": "get_port_sfp_diagnostics",
    "params": {
        "port": "A1"
    },
    "id": 1
}

// Response
{
    "result": {
        "vcc": 3.2734,
        "temperature": 34.81640625,
        "tx_bias": 3.791,
        "tx_power": 0.6558,
        "rx_power": 0.641,
        "port": "A1"
    },
    "id": 1
}
```

This method returns diagnostic information retreived from a SFP inserted into a port.

PARAMETERS

Field	Type	Description
Port	string	The name of the port to be queried

RESULT

Field	Type	Description
port	string	Port name
temperature	number	Temperature (degC)
vcc	number	Supply voltage (V)
tx_bias	number	Measured TX bias current (mA)
tx_power	number	Measured TX output power (mW)
rx_power	number	Measured RX input power (mW)

set_port_data_rate

```
>>> fusion.set_port_data_rate(port="A1", data_rate="ethernet_1g")
True
>>> fusion.set_port_data_rate(port="V1A", data_rate="ethernet_10g")
True
```

```
// Request
{
    "method": "set_port_data_rate",
    "params": {
        "port":"A1",
        "data_rate":"ethernet_1g"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method can be used to configure the data rate that a port should operate at.

PARAMETERS

Field	Type	Description
port	string	The name of the port you wish to retrieve SFP status info for
data_rate	string	The data rate you wish the port to operate at:`ethernet_10g` or `ethernet_1g`

RESULT

True if successful

---

get_port_stats

```
>>> fusion.get_port_stats("A1")
{u'rx_128_255b': 3653009, u'tx_64b': 0, u'tx_256_511b': 0, u'rx_65_127b': 1713621, u'rx_256_511b': 7307595, u'rx_link_changes': 0, u'tx_multicast': 0, u'port': u'A1', u'tx_512_1023b': 0
, u'rx_multicast': 0, u'rx_1024_1518b': 14123282, u'rx_link': True, u'tx_1519_1522b': 0, u'rx_runt': 0, u'tx_packets': 0, u'lldp_rx': True, u'rx_packets': 41411108, u'rx_unicast': 0, u'r
x_1519_1522b': 0, u'tx_128_255b': 0, u'rx_broadcast': 41411108, u'tx_bytes': 0, u'rx_dropped': 13669377, u'tx_unicast': 0, u'rx_bytes': 32835757254, u'tx_errors': 0, u'rx_64b': 0, u'rx_
pause': 0, u'rx_512_1023b': 14613601, u'tx_broadcast': 0, u'rx_errors': 0, u'tx_1024_1518b': 0, u'tx_65_127b': 0}
>>>
>>>
>>> fusion.get_port_stats("B2", "B1")
[{u'rx_128_255b': 3653009, u'tx_64b': 0, u'tx_256_511b': 0, u'rx_65_127b': 1713621, u'rx_256_511b': 7307595, u'rx_link_changes': 0, u'tx_multicast': 0, u'port': u'B1', u'tx_512_1023b':
0, u'rx_multicast': 0, u'rx_1024_1518b': 14123282, u'rx_link': True, u'tx_1519_1522b': 0, u'rx_runt': 0, u'tx_packets': 0, u'lldp_rx': True, u'rx_packets': 41411108, u'rx_unicast': 0, u
'rx_1519_1522b': 0, u'tx_128_255b': 0, u'rx_broadcast': 41411108, u'tx_bytes': 0, u'rx_dropped': 13669377, u'tx_unicast': 0, u'rx_bytes': 32835757254, u'tx_errors': 0, u'rx_64b': 0, u'r
x_pause': 0, u'rx_512_1023b': 14613601, u'tx_broadcast': 0, u'rx_errors': 0, u'tx_1024_1518b': 0, u'tx_65_127b': 0}, {u'rx_128_255b': 0, u'tx_64b': 0, u'tx_256_511b': 22949296, u'rx_65_
127b': 1110, u'rx_256_511b': 530, u'rx_link_changes': 0, u'tx_multicast': 0, u'port': u'B2', u'tx_512_1023b': 45979983, u'rx_multicast': 1110, u'rx_1024_1518b': 0, u'rx_link': True, u'tx
_1519_1522b': 378861, u'rx_runt': 0, u'tx_packets': 131227642, u'rx_packets': 1640, u'rx_unicast': 0, u'rx_1519_1522b': 0, u'tx_128_255b': 11461092, u'rx_broadcast': 530, u'tx_bytes': 10
6789190969, u'rx_dropped': 0, u'tx_unicast': 0, u'rx_bytes': 277952, u'tx_errors': 0, u'rx_64b': 0, u'rx_pause': 0, u'rx_512_1023b': 0, u'tx_broadcast': 131227642, u'rx_errors': 0, u'tx
_1024_1518b': 45373309, u'tx_65_127b': 3946960}]
```

```
// Request
{
    "method":"get_port_stats",
    "params": [
        "B2", "B1"
    ],
    "id": 1
}

// Response
{
    "result": [
        {
            "port":"B1",
            "rx_link":true,
            "rx_link_changes":0,
            "lldp_rx":true,
            "rx_packets":41411108,
            "rx_unicast":0,
            "rx_multicast":0,
            "rx_broadcast":41411108,
            "rx_64b":0,
            "rx_65_127b":1713621,
```

```
            "rx_128_255b":3653009,
            "rx_256_511b":7307595,
            "rx_512_1023b":14613601,
            "rx_1024_1518b":14123282,
            "rx_1519_1522b":0,
            "rx_bytes":32835757254,
            "rx_errors":0,
            "rx_dropped":13669377,
            "rx_runt":0,
            "rx_pause":0,
            "tx_packets":0,
            "tx_unicast":0,
            "tx_multicast":0,
            "tx_broadcast":0,
            "tx_64b":0,
            "tx_65_127b":0,
            "tx_128_255b":0,
            "tx_256_511b":0,
            "tx_512_1023b":0,
            "tx_1024_1518b":0,
            "tx_1519_1522b":0,
            "tx_bytes":0,
            "tx_errors":0
        },
        {
            "port":"B2",
            "rx_link":true,
            "rx_link_changes":0,
            "lldp_rx":true,
            "rx_packets":1640,
            "rx_unicast":0,
            "rx_multicast":1110,
            "rx_broadcast":530,
            "rx_64b":0,
            "rx_65_127b":1110,
            "rx_128_255b":0,
            "rx_256_511b":530,
            "rx_512_1023b":0,
            "rx_1024_1518b":0,
            "rx_1519_1522b":0,
            "rx_bytes":277952,
            "rx_errors":0,
            "rx_dropped":0,
            "rx_runt":0,
            "rx_pause":0,
            "tx_packets":131227642,
            "tx_unicast":0,
            "tx_multicast":0,
            "tx_broadcast":131227642,
            "tx_64b":0,
            "tx_65_127b":3946960,
            "tx_128_255b":11461092,
            "tx_256_511b":22949296,
            "tx_512_1023b":45979983,
            "tx_1024_1518b":45373309,
            "tx_1519_1522b":378861,
            "tx_bytes":106789190969,
            "tx_errors":0
        }
    ]
}
```


This method obtains the port statistics from a specific port.

PARAMETERS

This method can be provided with a list of port names as an array of strings.

RESULTS

field	type	description
port	string	Port name
rx_link	boolean	True if port has link with another interface
rx_link_changes	number	Counter of times rx_link has changed state
lldp_rx	boolean	True if LLDP traffic is received on this port
rx_packets	number	Number of packets received on the port

field	type	description
rx_unicast	number	Number of unicast packets received on the port
rx_multicast	number	Number of multicast packets recevied on the port
rx_broadcast	number	Number of broadcast packets recevied on the port
rx_64b	number	Number of 64b frames received on the port
rx_65_127b	number	Number of 65-127b frames received on the port
rx_128_255b	number	Number of 128-255b frames received on the port
rx_256_511b	number	Number of 256-511b frames received on the port
rx_512_1023b	number	Number of 512-1023b frames received on the port
rx_1024_1518b	number	Number of 1024-1518b frames received on the port
rx_1519-1522b	number	Number of 1519-1522b frames received on the port
rx_bytes	number	Raw number of bytes received on the port
rx_errors	number	Number of corrupt packets received on the port
rx_dropped	number	Number of packets dropped by the port
rx_runt	number	Number of runt frames received on the port
rx_pause	number	Number of pause frames received on the port
tx_packets	number	Number of packets sent by the port
tx_unicast	number	Number of unicast packets sent by the port
tx_multicast	number	Number of multicast packets sent by the port
tx_broadcast	number	Number of broadcast packets sent by the port
tx_64b	number	Number of 64b frames sent by the port
tx_65_127b	number	Number of 65-127b frames sent by the port
tx_128_255b	number	Number of 128-255b frames sent by the port
tx_256_511b	number	Number of 256-511b frames sent by the port
tx_512_1023b	number	Number of 512-1023b frames sent by the port
tx_1024_1518b	number	Number of 1024-1518b frames sent by the port
tx_1519_1522b	number	Number of 1519-1522b frames sent by the port
tx_bytes	number	Raw number of bytes sent by the port
tx_errors	number	Number of corrupt packets sent by the port

All fields are optional. A field will be omitted if the statistic is not available for the port.

Note when querying the stats of multiple ports will return an array consisting of entries in this format. Order of the entries is dependant on the order given when calling get_port_stats().

create_virtual_port

```
>>> fusion.create_virtual_port(name="V1")
True
```

```
// Request
{
    "method":"create_virtual_port",
    "params": {
        "name":"V1"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method creates a virtual port with a specified name. Note that if the virtual port is going to be used in a switch, mux or mirror object, the data rate must be set using the `set_port_data_rate` function.

PARAMETERS

Field	Type	Description
name	string	The name of the virtual port to be created

RESULT

True if successful

---

delete_virtual_port

```
>>> fusion.create_virtual_port(name="V1")
True
```

```
// Request
{
    "method":"delete_virtual_port",
    "params": {
        "name":"V1"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to delete a specified virtual port. **Note:** The virtual port must be removed from all network objects before this can be used.

PARAMETERS

Field	Type	Description
name	string	The name of the virtual port to be deleted

RESULT

True if successful

---

get_virtual_ports

```
>>> fusion.get_virtual_ports()
[{u'alias': u'', u'data_rate': u'ethernet_10g', u'name': u'V1A', u'description': u''}, {u'alias': u'', u'data_rate': u'ethernet_10g', u'name': u'V1B', u'description': u''}, {u'alias': u'', u'data_rate': u'ethernet_10g', u'name': u'V2A', u'description': u''}, {u'alias': u'', u'data_rate': u'ethernet_10g', u'name': u'V2B', u'description': u''}]
```

```
// Request
{
    "method":"get_virtual_port",
    "id": 1
}

// Response
{
    "result": [
        {
            "alias": "",
            "data_rate": "ethernet_10g",
            "name": "V1A",
            "description": ""
        },
        {
            "alias": "",
            "data_rate": "ethernet_10g",
            "name": "V1B",
            "description": ""
        },
        {
            "alias": "",
            "data_rate": "ethernet_10g",
            "name": "V2A",
            "description": ""
        },
        {
        "alias": "",
        "data_rate": "ethernet_10g",
        "name": "V2B",
        "description": ""
        }
    ],
    "id": 1
}
```

This method is used to see all current virtual ports and the descriptions for them

PARAMETERS

None

RESULT

Each virtual port entry consists of the following fields:

Field	Type	Description
alias	string	The alias assigned to the virtual port
data_rate	string	The data rate of the virtual port, either `ethernet_10g` or `ethernet_1g`
name	string	The name of the virtual port given when it was created
description	string	The description assigned to the virtual port

set_port_enable

```
>>> fusion.set_port_enable(port="A1", enable=0)
True
```

```
// Request
{
    "method":"set_port_enable",
    "params": {
        "port":"A1",
        "enable":0
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method can be used to enable or disable a port. Disabled ports remain members of any objects they were part of (for example a downstream port in a mux object), however the receiver and transmitter within that port are disabled.

PARAMETERS

Field	Type	Description
port	string	The port name to set the state of
enable	boolean	True if the port should be enabled

RESULT

True if successful

---

reset_port_stats

```
>>> fusion.reset_port_stats("A1")
True
```

```
// Request
{
    "method":"reset_port_stats",
    "params":"A1",
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is to reset all packet statistics on a port to zero.

PARAMETERS

Field	Type	Description
port	string	The port name to reset

RESULT

True if successful

Patches and Taps

create_patch

```
>>> fusion.create_patch(ports=["A10","C1"])
true
```

```
// Request
{
    "method":"create_patch",
    "params": {
        "ports": ["A10", "C1"]
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Create a new patch between two ports.

PARAMETERS

field	type	description
ports	array	The pair of ports to be patched. This must be an array of 2

elements.

RESULT

True if successful

---

delete_patch

```
>>> fusion.delete_patch(ports=["A10","C1"])
True
```

```
// Request
{
    "method":"delete_patch",
    "params": {
        "ports": ["A10", "C1"]
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Remove an existing patch between two ports.

PARAMETERS

Field	Type	Description
ports	array	The ports of an existing patch

RESULT

True if successful

---

get_patch

```
>>> fusion.get_patch()
[[u'A10', u'C1']]
```

```
// Request
{
    "method": "get_patch",
    "id": 1
}

// Response
{
    "result": [
        [
            "A10",
            "C1"
        ]
    ],
    "id": 1
}
```

Get the list of currently configured patches.

PARAMETERS

None

RESULT

A list of the currently active patches.

---

create_tap

```
>>> fusion.create_tap(port="A16", src_port="A10", direction="output")
True
```

```
// Request
{
    "method": "create_tap",
    "params": {
        "port":"A16",
        "src_port": "A10",
        "direction": "output"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Create a tap on a port.

A tap replicates data flow at layer 1 and sends it out another port. The data flow on the tapped port is not disturbed.

PARAMETERS

Field	Type	Description
port	string	Output port for tapped data
src_port	string	Port to be tapped
direction	string	Data direction to be tapped, either `input` or `output`

RESULT

True if successful

delete_tap

```
>>> fusion.delete_tap(port="A16", src_port="A10", direction="output")
True
```

```
// Request
{
    "method": "delete_tap",
    "params": {
        "port":"A16",
        "src_port": "A10",
        "direction": "output"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Remove an existing tap on a port.

PARAMETERS

Field	Type	Description
port	string	Output port of the tap
src_port	string	Tapped port
direction	string	Data direction of the tap, either `input` or `output`

RESULT

True if successful

---

get_tap

```
>>> fusion.get_tap()
[{u'direction': u'output', u'src_port': u'A10', u'port': u'A16'}]
```

```
// Request
{
    "method": "get_tap",
    "id": 1
}

// Response
{
    "result": [
        { "direction":"output", "src_port":"A10", "port":"A16" }
    ],
    "id": 1
}
```

Get the list of currently configured taps.

RESULT

This method returns an array of objects with the following fields:

Field	Type	Description
port	string	Output port of the tap

Field	Type	Description
src_port	string	Tapped port
direction	string	Data direction of the tap, either `input` or `output`

Switches and Muxes

get_object

```
>>> fusion.get_object()
[{u'object': {u'type': u'switch', u'name': u'my_switch'}, u'ports': [{u'port': u'A1'}, {u'port': u'A2'}, {u'port': u'A3'}, {u'port': u'A4'}]}, {u'object': {u'type': u'mux', u'name': u'my_mux'}, u'mode': u'layer2', u'ports': [{u'port': u'B1', u'side': u'up'}, {u'port': u'B2', u'side': u'down'}, {u'port': u'B3', u'side': u'down'}, {u'port': u'B4', u'side': u'down'}]}]
```

```
// Request
{
    "method": "get_object",
    "id": 1
}

// Response
{
    "result": [
        {
            "object": {
                "type": "switch",
                "name": "my_switch"
            },
            "ports": [
                {
                    "port": "A1"
                },
                {
                    "port": "A2"
                },
                {
                    "port": "A3"
                },
                {
                    "port": "A4"
                }
            ]
        },
        {
            "object": {
                "type": "mux",
                "name": "my_mux"
            },
            "mode": "layer2",
            "ports": [
                {
                    "port": "B1",
                    "side": "up"
                },
                {
                    "port": "B2",
                    "side": "down"
                },
                {
                    "port": "B3",
                    "side": "down"
                },
                {
                    "port": "B4",
                    "side": "down"
                }
            ]
        }
    ],
    "id": 1
}
```

This method returns a list of currently configured objects.

PARAMETERS

None

RESULT

The returned data is an array of objects with the following fields:

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of object
mode	string	Object mode if applicable: `raw` or `layer2` for mux objects
enable_vlan	boolean	True if VLAN support is enabled for this object, omitted otherwise
ports	array	Information about ports in the object

Each entry in the ports array has the following fields:

Field	Type	Description
port	string	Port name
side	string	`up` for upstream port, or `down` for downstream port (mux only)
vlan_id	integer	VLAN ID used by this port (VLAN tagging ports only)
block	array	A list of blocked output ports (omitted if no ports are blocked)

---

create_object

```
>>> fusion.create_object(type="mux", name="my_mux", mode="layer2")
True
```

```
// Request
{
    "method":"create_object",
    "params": {
        "type":"mux",
        "name":"my_mux",
        "mode":"layer2"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Create a new switch or mux object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of new object
mode	string	Object mode: `raw` or `layer2` for mux objects, omitted for switch objects
enable_vlan	boolean	Set to true if VLAN support is desired (optional)

RESULT

True if successful

delete_object

```
>>> fusion.delete_object(type="switch", name="my_switch")
True
```

```
// Request
{
    "method":"delete_object",
    "params": {
        "type":"switch",
        "name":"my_switch",
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Delete a switch or mux object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of object to be deleted

RESULT

True if successful

add_object_port

```
>>> fusion.add_object_port(object={"type":"mux","name":"my_mux"}, port="B12", side="up", vlan_id=str(11))
True
```

```
// Request
{
    "method":"add_object_port",
    "params": {
        "object": {
            "type": "mux",
            "name": "my_mux"
        },
        "port": "B12",
        "side": "up",
        "vlan_id": "11"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Add a port to a switch or mux object.

For mux objects, the `side` field must be provided to specify whether the port is an upstream or downstream port. If VLAN is enabled and a VLAN is to be assigned to the port, the vlan_id field must be filled as well.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of object
port	string	Port to be added to the object
side	string	`up` for upstream ports or `down` for downstream ports (mux objects only)
vlan_id	integer	VLAN ID used by this port (optional)

RESULT

True if successful

---

remove_object_port

```
>>> fusion.remove_object_port(object={"type":"mux":"name":"my_mux"}, port="B12")
True
```

```
// Request
{
    "method":"remove_object_port",
    "params": {
        "object": {
            "type": "mux",
            "name": "my_mux"
        },
        "port": "B12"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Remove a port from a switch or mux object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of object
port	string	Port to be removed from the object

RESULT

True if successful

---

set_object_mode

```
>>> fusion.set_object_mode(object={"type":"mux","name":"my_mux"}, mode="raw")
True
```

```
// Request
{
    "method": "set_object_mode",
    "params": {
        "object": {
            "type": "mux",
            "name": "my_mux"
        },
        "mode": "raw"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Change the mode attribute of a mux object.

PARAMETERS

Field	Type	Description
type	string	Object type: `mux`
name	string	Name of mux object
mode	string	New mode of the mux object: `raw` or `layer2`

RESULT

True if successful

---

set_object_vlan

```
>>> fusion.set_object_vlan(object={"type":"mux","name":"my_mux"}, enable_vlan=True)
True
```

```
// Request
{
    "method": "set_object_vlan",
    "params": {
        "object": {
            "type": "mux",
            "name": "my_mux"
        },
        "enable_vlan": true
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Enable or disable VLAN support on a switch or mux object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of object
enable_vlan	boolean	Set to true to enable VLAN support, or false to disable

Field	Type	Description

RESULT

True if successful

set_object_unknown_unicast

```
fusion.set_object_unknown_unicast(object={"type":"switch", "name":"my_switch"}, port="B2", unknown_unicast=0)
>>> True
```

```
// Request
{
    "method":"set_object_unknown_unicast",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "port": "B1",
        "unknown_unicast": "0"
    },
    "id": 1
}
// Response
{
    "result": true,
    "id": 1
}
```

Allow transmission of unknown unicast packets on a specified port

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object
port	string	Port to associate the behaviour to
unknown_unicast	bool	'0' to disallow unknown unicast, '1' to accept

RESULT

True if successful

add_object_static_mac

```
>>> fusion.add_object_static_mac(object={"type":"switch", "name":"my_switch"}, port="B1", static_mac="643F5F011630")
True
```

```
// Request
{
    "method":"add_object_static_mac",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "port": "B1",
        "static_mac": "643F5F011630"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Add a static MAC address to a single port on an object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object
port	string	Port to associate the static MAC address with
static_mac	string	12 hex chars representing the MAC address to be assigned

RESULT

True if successful

---

remove_object_static_mac

```
>>> fusion.remove_object_static_mac(object={"type":"switch", "name":"my_switch"}, port="B1", static_mac="643F5F011630")
True
```

```
// Request
{
    "method":"remove_object_static_mac",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "port": "B1",
        "static_mac": "643F5F011630"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Remove a static MAC address from a single port on an object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`

Field	Type	Description
name	string	Name of the object
port	string	Port to remove static MAC address from
static_mac	string	MAC address to be removed

RESULT

True if successful

---

get_object_static_mac

```
>>> fusion.get_object_static_mac(object={"type":"switch","name":"my_switch"})
{u'static_mac': {u'643F5F011630': [u'B1']}, u'object': {u'type': u'switch', u'name': u'my_switch'}
```

```
// Request
{
    "method": "get_object_static_mac",
    "params": {
        "type": "switch",
        "name": "my_switch"
    },
    "id": 1
}

// Response
{
    "result": {
        "static_mac": {
            "643F5F011630": [
                "B1"
            ]
        },
        "object": {
            "type": "switch",
            "name": "my_switch"
        }
    },
    "id": 1
}
```

Retrieve all static MAC addresses assigned to a specified object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object

RESULT

This returns a list of all static MAC addresses currently assigned to the specified object.

---

add_object_igmp_static_group

```
>>> fusion.add_object_igmp_static_group(object={"type":"switch","name":"my_switch"}, port="A1", group="224.1.1.1")
True
```

```
// Request
{
    "method": "add_object_igmp_static_group",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "port": "A1",
        "group": "224.1.1.1"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Adds a static route of a multicast group to a port for a given mux/switch object

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object
port	string	The port to add a static route to
group	string	The IGMP group to add

RESULT

True if successful

---

remove_object_igmp_static_group

```
>>> fusion.remove_object_igmp_static_group(object={"type":"switch","name":"my_switch"}, port="A1", group="224.1.1.1")
True
```

```
// Request
{
    "method": "remove_object_igmp_static_group",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "port": "A1",
        "group": "224.1.1.1"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Removes a static route of a multicast group to a port for a given mux/switch object

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`

Field	Type	Description
name	string	Name of the object
port	string	The port to remove a static route from
group	string	The IGMP group to remove

RESULT

True if successful

get_object_mac_table

```
>>> fusion.get_object_mac_table(object={"type":"switch", "name":"my_switch"})
{u'mac_table': {u'643F5F011630': [{u'learned_routes': [u'B1'], u'port': u'B2', u'static_routes': []}, {u'learned_routes': [u'B1'], u'port': u'B3', u'static_routes': []}], u'102233445566
': [{u'learned_routes': [], u'lock': True, u'port': u'B1', u'static_routes': [u'B1']}, {u'learned_routes': [], u'lock': True, u'port': u'B2', u'static_routes': [u'B1']}, {u'learned_rout
es': [], u'lock': True, u'port': u'B3', u'static_routes': [u'B1']}]}, u'object': {u'type': u'switch', u'name': u'my_switch'}}
```

```
// Request
{
    "method": "get_object_mac_table",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        }
    },
    "id": 1
}

// Response
{
    "mac_table": {
        "643F5F011630": [
            {
                "learned_routes": [
                    "B1"
                ],
                "port": "B2",
                "static_routes": []
            },
            {
                "learned_routes": [
                    "B1"
                ],
                "port": "B3",
                "static_routes": []
            }
        ],
        "102233445566": [
            {
                "learned_routes": [],
                "lock": true,
                "port": "B1",
                "static_routes": [
                    "B1"
                ]
            },
            {
                "learned_routes": [],
                "lock": true,
                "port": "B2",
                "static_routes": [
                    "B1"
                ]
            },
            {
                "learned_routes": [],
                "lock": true,
                "port": "B3",
                "static_routes": [
                    "B1"
                ]
            }
        ]
    },
    "object": {
        "type": "switch",
        "name": "my_switch"
    },
    "id": 1
}
```

This method allows for retrieval of all MAC address table entries for a specific object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object

RESULT

Returns an array of the MAC address table of the requested object, noting static and dynamically learned routes

set_object_mac_learning

```
>>> fusion.set_object_mac_learning(object={"type":"switch","name":"my_switch"}, mac_address_learning="False")
True
```

```
// Request
{
    "method": "set_object_mac_learning",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "mac_address_learning": false
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method will allow for the enabling or disabling of MAC address learning on objects that support it. Note: MAC address learning is enabled by default for objects that support it

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object
mac_address learning	boolean	Set to enable or disable MAC address learning

RESULT

True if successful

set_object_igmp

```
>>> fusion.set_object_igmp(object={"type":"switch","name":"my_switch"},snooping="enabled",flood_unknown="true",fast_leave="true",querier="192.168.10.37",version="2");
True
```

```
// Request
{
    "method": "set_object_igmp",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "snooping": "enabled",
        "flood_unknown": "true",
        "fast_leave": "true",
        "querier": "192.168.10.37",
        "version": "2"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method allows for setting of an object's IGMP snooping setting and associated features

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object
snooping	string	Enables IGMP Snooping for the object
flood_uknown	string	Determines if unknown multicast traffic is broadcast or dropped
fast_leave	string	If true, IGMP leave messages will result in immediate removal of the group, else wait for a timeout
querier	string	If specified, sets the Fusion as an IGMP querier with the specified source IP address
version	string	IGMP protocol version to use: `1`, `2` or `3`

RESULT

True if successful

---

get_object_igmp

```
>>> fusion.get_object_igmp(object={"type":"switch","name":"my_switch"})
{u'snooping': True, u'object': {u'type': u'switch', u'name': u'my_switch'}, u'version': 2, u'querier': u'192.168.10.37', u'fast_leave': True, u'flood_unknown': True}
```

```
// Request
{
    "method": "get_object_igmp",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        }
    },
    "id": 1
}

// Response
{
    "result": {
        "snooping": true,
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "version": 2,
        "querier": "192.168.10.37",
        "fast_leave": true,
        "flood_unknown": true
    },
    "id": 1
}
```

Read the current IGMP settings for an object

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	The name of the object

RESULT

Field	Type	Description
snooping	boolean	Based on whether IGMP Snooping has been enabled for an object or not

Field	Type	Description
type	string	Object type: `switch` or `mux`
name	string	Name of the object
version	number	The version number of the IGMP protocol in use
querier	string	The source the IP address the querier's messages will be given
fast_leave	boolean	If true, IGMP leave messages will result in immediate removal of the group, else wait for a timeout

add_object_block

```
>>> fusion.add_object_block(object={"type":"switch","name":"my_switch"}, port="A1", dst_port="A2")
True
```

```
// Request
{
    "method": "add_object_block",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "port": "A1",
        "dst_port": "A2"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Adds an output port to the blocked routes of a port in a switch object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch` only
name	string	Object name
port	string	Input port to block traffic from
dst_port	string	Output port to block traffic to

RESULT

True if successful

remove_object_block

```
>>> fusion.remove_object_block(object={"type":"switch","name":"my_switch"}, port="A1", dst_port="A2")
True
```

```
// Request
{
    "method": "remove_object_block",
    "params": {
        "object": {
            "type": "switch",
            "name": "my_switch"
        },
        "port": "A1",
        "dst_port": "A2"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Removes an output port from the blocked routes of a port in a switch object.

PARAMETERS

Field	Type	Description
type	string	Object type: `switch`
name	string	Object name
port	string	Input port to block traffic from
dst_port	string	Output port to block traffic to

RESULT

True if successful

Mirrors

get_mirror

```
>>> fusion.get_mirror("my_mirror")
[{u'timestamp_mode': u'fcs', u'output': [u'A1'], u'objects': {u'switch': [u'my_switch']}, u'name': u'my_mirror', u'ports': [u'B10']}]
```

```
// Request
{
    "method": "get_mirror",
    "params": "my_mirror",
    "id": 1
}

// Response
{
    "timestamp_mode": "fcs",
    "output": [
        "A1"
    ],
    "objects": {
        "switch": [
            "my_switch"
        ]
    },
    "name": "my_mirror",
    "ports": [
        "B10"
    ],
    "id": 1
}
```

This method gets the current status and structure of a mirror.

PARAMETERS

Field	Type	Description
name	string	Name of the mirror

RESULT

Field	Type	Description
name	string	Name of the mirror
output	string	The port the mirror will output its traffic to
ports	string	The ports that have their traffic mirrored to the output
objects	string	The objects that are having their traffic mirrored to the output
timestamp_mode	string	Reports the timestamping mode (i.e. format) in use with a particular mirror

create_mirror

```
>>> fusion.create_mirror(name="my_mirror")
True
```

```
// Request
{
    "method": "create_mirror",
    "params": {
        "name": "my_mirror"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method creates a mirror object on the Fusion with the specified name.

PARAMETERS

Field	Type	Description
name	string	Name of the mirror

RESULT

True if successful

delete_mirror

```
>>> fusion.delete_mirror(name="my_mirror")
True
```

```
// Request
{
    "method": "delete_mirror",
    "params": {
        "name": "my_mirror"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method removes a mirror object on the Fusion with the specified name.

PARAMETERS

Field	Type	Description
name	string	Name of the mirror

RESULT

True if successful

---

set_mirror_timestamp_mode

```
>>> fusion.set_mirror_timestamp_mode(mirror="my_mirror", mode="fcs")
True
```

```
// Request
{
    "method": "set_mirror_timestamp_mode",
    "params": {
        "mirror": "my_mirror",
        "mode": "fcs"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method enables or disables the timestamping mode of a mirror object.

PARAMETERS

Field	Type	Description
mirror	string	Name of the mirror to modify
mode	string	Timestamping mode to use: `fcs` or `fcs-compat` or `none`

RESULT

True if successful

---

add_mirror_output

```
>>> fusion.add_mirror_output(mirror="my_mirror", port="B8")
True
```

```
// Request
{
    "method": "add_mirror_output",
    "params": {
        "mirror": "my_mirror",
        "port": "B8"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Use this method to specify the output port for all of the mirror's traffic

PARAMETERS

Field	Type	Description
mirror	string	Name of the mirror
port	string	Name of the port to add as output

RESULT

True if successful

---

remove_mirror_output

```
>>> fusion.remove_mirror_output(mirror="my_mirror", port="B8")
True
```

```
// Request
{
    "method": "remove_mirror_output",
    "params": {
        "mirror": "my_mirror",
        "port": "B8"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

Removes the output port from a specified mirror object.

PARAMETERS

Field	Type	Description
mirror	string	Name of the mirror
port	string	Name of the port to remove as output

RESULT

True if successful

add_mirror_port

```
>>> fusion.add_mirror_port(mirror="my_mirror", port="C1")
True
```

```
// Request
{
    "method": "add_mirror_port",
    "params": {
        "mirror":"my_mirror",
        "port":"C1"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to add a singular port to a mirror object.

PARAMETERS

Field	Type	Description
mirror	string	Name of the mirror
port	string	Name of the port whose traffic is to be mirrored

RESULT

True if successful

remove_mirror_port

```
>>> fusion.remove_mirror_port(mirror="my_mirror", port="C1")
True
```

```
// Request
{
    "method": "remove_mirror_port",
    "params": {
        "mirror":"my_mirror",
        "port":"C1"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to remove specific ports from a mirror object.

PARAMETERS

Field	Type	Description
mirror	string	Name of the mirror
port	string	Name of the port to remove from mirroring

RESULT

True if successful

---

add_mirror_object

```
>>> fusion.add_mirror_object(mirror="my_mirror", type="switch", object="my_switch")
True
```

```
// Request
{
    "method": "add_mirror_object",
    "params": {
        "mirror": "my_mirror",
        "type": "switch",
        "object": "my_switch"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to add a whole object to a mirror object.

PARAMETERS

Field	Type	Description
mirror	string	Name of the mirror to add the object to
type	string	The type of the object to add: `switch` or `mux`
object	string	Name of the object to add

RESULT

True if successful

---

remove_mirror_object

```
>>> fusion.remove_mirror_object(mirror="foo", type="my_mirror", object="my_switch")
True
```

```
// Request
{
    "method": "remove_mirror_object",
    "params": {
        "mirror": "my_mirror",
        "type": "switch",
        "object": "my_switch"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to remove objects currently assigned to a mirror.

PARAMETERS

Field	Type	Description
mirror	string	Name of the mirror to remove the object from
type	string	The Type of the object to add: `switch` or `mux`
object	string	Name of the object to remove

RESULT

True if successful

Internal Module Configuration

power_on_module

```
>>> fusion.power_on_module(module="Y")
True
```

```
// Request
{
    "method": "power_on_module",
    "params": {"module": "X"},
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to turn on a specific internal module. Note: this does not apply to FPGA internal modules.

PARAMETERS

Field	Type	Description
module	string	Name of the module to power on (“X” or “Y”)

RESULTS

True if successful

power_off_module

```
>>> fusion.power_off_module(module="Y")
True
```

```
// Request
{
    "method": "power_off_module",
    "params": {
        "module": "X"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to turn off a specific internal module. Note: this does not apply to FPGA internal modules.

PARAMETERS

Field	Type	Description
module	string	Name of the module to power off (“X” or “Y”)

RESULTS

True if successful

---

set_module_xvc_server

```
>>> fusion.set_module_xvc_server(module="X", enable=True, port=80)
True
```

```
// Request
{
    "method": "set_module_xvc_server",
    "params": {
        "enable": true,
        "port": 6767,
        "module": "X"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to enable or disable the XVC server of an FPGA module.

PARAMETERS

Field	Type	Description
module	string	Name of the module to enable the XVC server on
enable	boolean	Set to `1` to enable XVC, `0` to disable
port	int	Port number to be assigned to the XVC connection

RESULTS

True if successful

---

set_module_serial_server

```
>>> fusion.set_module_serial_server(enable=True, port=6768, module="X")
True
```

```
// Request
{
    "method": "set_module_serial_server",
    "params": {
        "enable": true,
        "port": 6768,
        "module": "X"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to enable or disable the serial server of an internal module.

PARAMETERS

Field	Type	Description
module	string	Name of the module to enable the serial server connection on
enable	boolean	Set to `1` to enable the serial server on the module

RESULTS

True if successful

---

set_module_serial_console

```
>>> fusion.set_module_serial_console(module="X", enable=True)
True
```

```
// Request
{
    "method": "set_module_serial_console",
    "params": {
        "enable": true,
        "port": 6769,
        "module": "X"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method is used to enable or disable the serial console interface for an internal module.

PARAMETERS

Field	Type	Description
module	string	Name of the module to enable the serial console on
enable	boolean	Set to `1` to enable to serial console connection for the module, `0` to disable

RESULTS

True if successful

get_modules

```
>>> fusion.get_modules()
[{u'function': u'switch', u'state': u'running', u'name': u'X'}]
```

```
// Request
{
    "method": "get_modules",
    "id": 1
}

// Response
{
    "result": [
        {
         "function": "switch",
         "state": "running",
         "name": "X"
        }
    ],
    "id": 1
}
```

This method shows information about the Internal Modules installed in the ExaLINK Fusion

PARAMETERS

None

RESULT

Field	Type	Description
function	string	The function of the internal module: `switch`, `mux`, or `custom`
state	string	The current state of the internal module: `running`, `stopped`, or `initializing`
name	string	The position of the internal module: `X` or `Y`

set_module_function

```
>>> fusion.set_module_function(module="X", function="switch")
True
>>> fusion.set_module_function(module="Y", function="custom")
True
```

```
// Request
{
    "method": "set_module_function",
    "params": {
        "module": "X",
         "function": "switch"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method sets the firmware that should be used on a FPGA Internal Module

PARAMETERS

Field	Type	Description
module	string	Select which module the function is being set: `X` or `Y`
function	string	The firmware type the internal module should run: `switch`, `mux` or `custom`

RESULT

True if successful

---

set_module_fpga_bitstream

```
>>> fusion.set_module_fpga_bitstream(module="Y", bitstream="mybitfile.bit")
True
```

```
// Request
{
    "method": "set_module_fpga_bitstream",
    "params": {
        "module": "Y",
        "bitstream": "mytbitfile.bit"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method sets the bitfile that should be loaded on an FPGA module configured to run custom firmware

PARAMETERS

Field	Type	Description
module	string	Select which internal module the bitstream should be loaded to
bitstream	string	The filename of the bitfile

RESULT

True if successful

It is possible place a custom bitfile onto the Fusion via SFTP. Invoke sftp, connecting to the Fusion's admin account (i.e. admin@myfusion) then using the command `put mybitfile.bit`. Note that a module will also needs its function set to *custom* via *set_module_function* or otherwise.

---

reconfigure_module_fpga

```
>>> fusion.reconfigure_module_fpga(module="Y")
True
```

```
// Request
{

    "method": "reconfigure_module_fpga",
    "params": {
      "module": "Y"
    },
    "id": 1
}

// Response
{
    "result": true,
    "id": 1
}
```

This method can be used to reload the configuration bitstream for an internal module

PARAMETERS

Field	Type	Description
module	string	The module in which the FPGA should be configured: `X` or `Y`

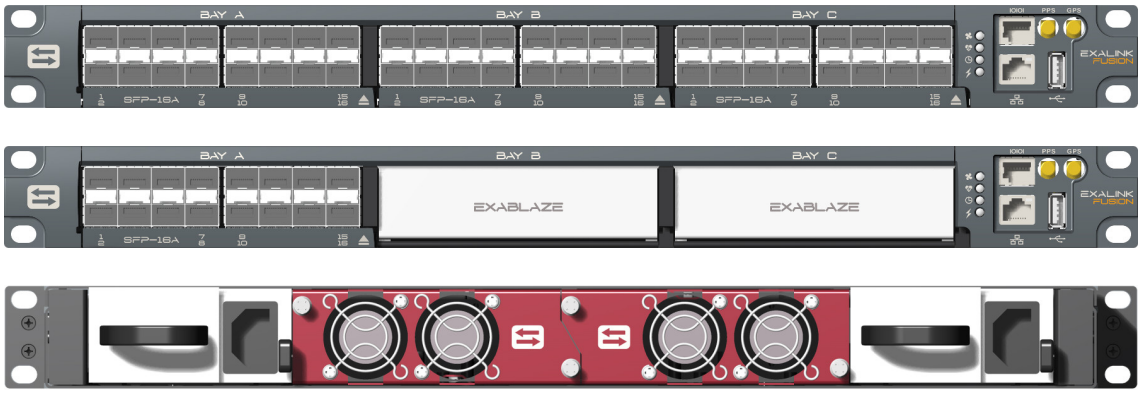
RESULT

True if successful

Miscellaneous Downloads

Visio Stencils

A set of Visio stencils for the ExaLINK Fusion is available for download here.



Legal Notices

Some components of the ExaLINK Fusion firmware are free software and licensed under the GNU General Public License or the GNU Lesser General Public License. You may obtain the source code for these components by contacting us at support@exablaze.com.

OpenSSL

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org). This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

Berkeley DB

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Redistributions in any form must be accompanied by information on how to obtain complete source code for the DB software and any accompanying software that uses the DB software. The source code must either be included in the distribution or be available for no more than the cost of distribution plus a nominal fee, and must be freely redistributable under reasonable conditions. For an executable file, complete source code means the source code for all modules it contains. It does not include source code for modules or files that typically accompany the major components of the operating system on which the executable file runs.

THIS SOFTWARE IS PROVIDED BY ORACLE ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED. IN NO EVENT SHALL ORACLE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

To obtain the complete source code for the DB software and accompanying software, please contact support@exablaze.com.