MXK Configuration Guide

For software version 1.14.2
May 2008
Document Part Number: 830-01812-01

ZHONE™

Zhone Technologies
@Zhone Way
7001 Oakport Street
Oakland, CA 94621
USA
510.777.7000
www.zhone.com
info@zhone.com

CONTENTS

1 BASIC CONFIGURATION

This chapter describes how to perform the basic configuration of the MXK, including how to modify the default settings for the uplink cards and how to enable slot cards. It includes the following sections:

Configuration overview

For redundant systems, you must configure the physical interfaces on both the active and standby cards. In addition, you must manually keep the configuration of the physical interfaces on the active and standby cards in sync.

MXK default configuration

This default configuration of the MXK is as follows:

- Administrative user name is **admin**, password is **zhone**.
- Slot cards (except the uplink card) must be enabled in a **card-profile** before they will boot up.
- The uplink card provides a serial craft interface and a 10/100 Ethernet interface for management. Any other Ethernet port can be used for management, however, the management port cannot be used for user traffic. The default uplink Ethernet interfaces are:
  - 1-a-1-0/eth

    Management interface (not for user traffic)
  - 1-a-2-0/eth and 1-a-3-0/eth

    XFP based 10-GigE interfaces
  - 1-a-3-0/eth through 1-a-11-0/eth

    SFP based 1-GigE interfaces

- A default **system** profile 0 exists with the following configuration:
  - Authentication traps are not enabled
  - ZMS communication is not configured
  - Alarm notification and output is enabled for all severity levels

Log in to the serial (craft) port

Note: Do not use the serial craft port of a standby card to modify its configuration.

The MXK unit provides an out-of-band RS232 D serial (craft) interface for managing the unit. To access the serial port, configure your terminal interface software with the following settings:

- 9600bps
- 8 data bits
- No parity
- 1 stop bit
- No flow control

Tip: The serial (craft) port settings can be changed by modifying the **rs232-profile**.

You must perform the initial configuration of the system using the serial (craft) interface. After you have completed the initial configuration, you can manage the MXK unit over the network through a telnet session over the Ethernet interface.

Note: The MXK supports 6 concurrent management sessions, 5 telnet sessions and a single local session through the serial (craft) port.

Logging in and out of the system

Log into the system (the default user name is **admin**, the default password is **zhone**):

```
login:admin
password:
zSH>
```

To log out of the system, enter the **logout** command:

```
zSh> logout
```

Tip: The system automatically logs you out after a period of inactivity. The default logout time is 10 minutes, but can be changed with the **timeout** command. Refer to *Zhone CLI Reference Guide* for information on the **timeout** command.

Enabling and disabling logging

By default logging is enabled on the serial craft port and disabled over telnet sessions. To enable or disable logging for the session, using the following command:

```
zSh> log session on | off
```

The **log session** command only applies to the current session. You can also enable or disable logging for all serial craft port sessions using the following command:

```
zSh> log serial on | off
```

This command setting persists across system reboots.

Configure a management interface

This section describes how to configure the following interfaces to remotely manage the MXK:

- *Uplink card 10/100 BaseT Ethernet interface* on page 9
- *Manage the MXK with ZMS* on page 11
- *VLAN management interface* on page 12

Uplink card 10/100 BaseT Ethernet interface

The MXK has a 10/100 BaseT Ethernet interface on the uplink card. The **ip-interface-record** for the uplink card is named **ethernet1**. This interface is shared between the two Ethernet ports on redundant uplink cards (if they exist). The system can be reached using the address configured in the **ethernet1 ip-interface-record**, no matter which card is active.

Configuring the Ethernet IP interface

The following example configures the IP address for the system:

```
zSH> interface add 1-a-1-0/eth 192.168.1.1/24
Created ip-interface-record ethernet1/ip.
zSH> interface show
1 interface
Interface      Status  Rd/Address            Media/Dest Address      IfName
--------------------------------------------------------------------------------
1/a/1/0/ip     UP      1 192.168.1.1/24      00:01:47:13:44:56       ethernet1
-------------------------------------------------------------------------------
```

Verifying the interface

Use the **interface show** command to verify that the Ethernet interface was configured correctly:

```
zSH> interface show
Interface      Status  Rd/Address            Media/Dest Address IfName
--------------------------------------------------------------------------
1/a/1/0/ip     UP      1 192.168.8.21/24    00:01:47:65:02:f2  1-1-1-0
```

Creating a default route

The following example creates a default route using the gateway 192.168.8.1 with a cost of 1 (one):

```
route add default 192.168.8.1 1
```

Verifying the route

Use the **route show** command to verify that the routes were added:

```
zSH> route show
Dest                     Nexthop          Cost      Owner
------------------------------------------------------------
0.0.0.0/0                192.168.8.1      1         STATICLOW
192.168.8.0/24           1/a/1/0/ip       1         LOCAL
```

Use the **ping** command to verify connectivity to the default gateway:

```
zSH> ping 192.168.8.1
PING 192.168.8.1: 64 data bytes
!!!!!
----192.168.8.1 PING Statistics----
5 packets transmitted, 5 packets received
round-trip (ms)  min/avg/max = 0/0/0
```

To stop the ping, press CTRL+C.

Adding a route to the remote LAN

After creating the IP interface, you might need to create a route to the remote device’s LAN interface using the **route add** command. The command uses the following syntax:

```
route add destination mask next-hop cost
```

For example, in the following configuration, add a route to the 192.168.10.0 network using the MXK uplink interface as the gateway.

```
route add 192.168.10.0 255.255.255.0 192.168.8.1 1
```

Manage the MXK with ZMS

Note: For details on using ZMS, refer to the *ZMS Administrator's Guide* and the *NetHorizhon User's Guide*.

The **system** profile contains parameters that configure the system contact information for the MXK and connection information for the ZMS. This profile does not need to be modified in order to manage the MXK with ZMS.

CLI provisioning and ZMS

Making a change to the device configuration

CLI configuration of a device being managed by the ZMS is disabled by default. Attempting to configure the device results in an error:

If you plan to use a script to provision the device from the CLI while it is being managed by the ZMS:

1 Update the **system** profile to disable partial config syncs to ZMS:

```
zSH> update system 0
system  0
Please provide the following: [q]uit.
syscontact: ----------->  {Zhone Global Services and Support 7001 Oakport Street
Oakland Ca. (877) Zhone20 (946-6320) Fax (510)777-7113 support@zhone.com}:
sysname: -------------->  {Zhone MxK}:
syslocation: ---------->  {Oakland}:
enableauthtraps: ------>  {disabled}:
setserialno: ---------->  {0}:
zmsexists: ------------>  {true}:false
zmsconnectionstatus: -->  {inactive}:
zmsipaddress: --------->  {0.0.0.0}:
configsyncexists: ----->  {false}:
configsyncoverflow: --->  {false}:
configsyncpriority: --->  {high}:
configsyncaction: ----->  {noaction}:
configsyncfilename: --->  {}:
configsyncstatus: ----->  {syncinitializing}:
configsyncuser: ------->  {}:
configsyncpasswd: ----->  {** private **}:  ** read-only **
numshelves: ----------->  {1}:
shelvesarray: --------->  {}:
numcards: ------------->  {3}:
ipaddress: ------------>  {0.0.0.0}:
alternateipaddress: --->  {0.0.0.0}:
countryregion: -------->  {us}:
primaryclocksource: --->  {0/0/0/0/0}:
ringsource: ----------->  {internalringsourcelabel}:
revertiveclocksource: ->  {true}:
voicebandwidthcheck: -->  {false}:
alarm-levels-enabled: ->  {critical+major+minor+warning}:
userauthmode: --------->  {local}:
```

```
radiusauthindex: ------>  {0}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

2 After the provisioning is complete, perform a full config sync from ZMS.

VLAN management interface

To create a management interface over the first GigE port, use the interface add command and specify a VLAN:

```
zSH> interface add 1-a-1-0/eth vlan 99 10.10.10.1/24
Created ip-interface-record ethernet1-99/ip
```

Radius support

The MXK supports local and RADIUS (Remote Authentication Dial In User Service) access authentication. The MXK can be configured for local authentication, RADIUS authentication, or RADIUS then local authentication. RADIUS users are configured with the Service-Type attribute as Administrative-User or NAS-Prompt-User. RADIUS is used for only login authentication, not severity levels.

Table 1 shows the mapping of service-type to MXK permissions.

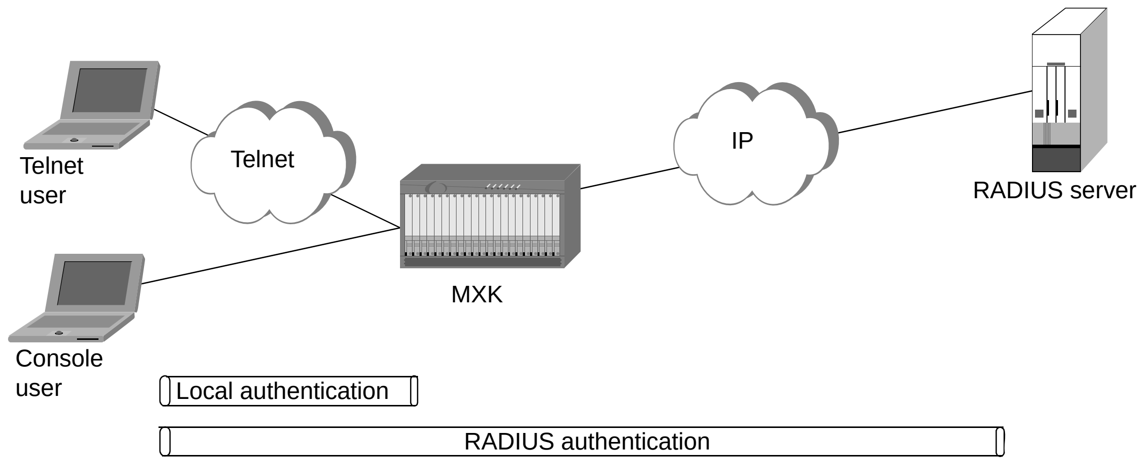
Table 1: Service type mapping to MXK permissions

Service-Type Attribute	MXK permissions
Administrative-User	admin, zhonedebug, voice, data, manuf, database, systems, tools, useradmin
NAS-Prompt-User	admin, voice, data, manuf, database, systems, tools, useradmin

When establishing a connection to the MXK with RADIUS authentication, the MXK passes RADIUS information securely to the RADIUS server. The RADIUS server then authenticates the user and either allows or denies access to the MXK. If access is denied and the local authentication option is also configured, the MXK then authenticates access based on the locally configured users and passwords. For logins and failed logins, a console message is generated with user ID and IP address of the device from which the login originated. Failed logins also are logged as alert level messages in the MXK system log file.

By default, RADIUS access uses the UDP port 1812 for authentication.This parameter can be changed in the radius-client profile.

Figure 1: MXK RADIUS authentication



Note: Follow the RADIUS server guidelines for RADIUS configuration instructions. For example, when using the MXK with the FreeRadius server:

- Create only one entry in the *clients.conf* file for each subnet or individual MXK. For individual MXKs, the IP in this file must match the IP address of the outbound interface used by the MXK to connect to the RADIUS server.
- The MXK uses the value stored in the RADIUS system.sysname file for the NAS-Identifier attribute.
- The shared-secret in the MXK radius-client profile, must exactly match the shared-secret in the RADIUS client entry.

Configuring RADIUS support

The MXK can be configured for local authentication, RADIUS authentication, or RADIUS then local authentication. Multiple radius-client profiles can be defined using the index and subindex numbers. This index scheme can be used to create index numbers for groups of RADIUS servers. When an index number is specified in the system profile, the MXK attempts authenication from each RADIUS server in that group in sequential order of the subindex numbers.

To configure RADIUS support:

Note: Before beginning this procedure, ensure that the MXK has IP connectivity to the RADIUS server.

1. Update the RADIUS server with settings for the Zhone prompts.
2. Create a radius-client profile on the MXK with the desired index number and RADIUS settings for server name, shared secret, number of retries, and other parameters. The first number in the index is used to group radius-client profiles so multiple profiles can be assigned to a MXK. The

second number in the index specifies the order in which radius-client profiles are referenced. This example specifies the radius-client 1/1 with server name *radius1* and a shared-secret of *secret*. A DNS resolver must be configured in the system to resolve the server name and IP address. If a DNS resolver is not available, specify the IP address of the The index 1/1 specifies that this profile is the first profile in group 1.

```
zSH> new radius-client 1/1
Please provide the following: [q]uit.
server-name: ---->  {}: radius1.test.com [DNS resolver must be configured in the system.]
udp-port: ------->  {1812}:
shared-secret: -->  {** password **}: secret
retry-count: ---->  {5}:
retry-interval: ->  {1}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
Record created.
```

Another method to reference the RADIUS server is by specifying the IP address. This example specifies the radius-client 1/1 with server IP address 172.24.36.148 and a shared-secret of *secret*. The index 1/1 specifies that this profile is the first profile in group 1.

```
zSH> new radius-client 1/1
Please provide the following: [q]uit.
server-name: ---->  {}: 172.24.36.248
udp-port: ------->  {1812}:
shared-secret: -->  {** password **}: secret
retry-count: ---->  {5}:
retry-interval: ->  {1}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
Record created.
```

3 Create another radius-client profile on the MXK with the desired RADIUS settings for server name, shared secret, number of retries, and other parameters. This example specifies the radius-client 1/2 with server IP address 172.24.36.148 and a shared-secret of *secret*. The index 1/2 specifies that this profile is the second profile in group 1.

```
zSH> new radius-client 1/2
Please provide the following: [q]uit.
server-name: ---->  {}: 172.24.36.249
udp-port: ------->  {1812}:
shared-secret: -->  {** password **}: secret
retry-count: ---->  {5}:
retry-interval: ->  {1}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
Record created.
```

Create additional radius-client profiles for each additional RADIUS server to be assigned to this MXK.

4 In the system profile on the MXK, set the desired user authentication method and specify the index of the radius profile to use. This examples specifies the **radiusauthindex** of 1. This index is configured with two radius-client profiles (1/1, 1/2). The MXK first attempts authenication using the server specified in radius-client 1/1. If this authenitication fails, the MXK attempts authenication using radius-client 1/2 server. If this authentication also fails, the MXK then attempts authentication based on the authentication mode setting in the system profile. This example uses **radiusthenlocal**.

Caution: If the *userauthmode* field in the system 0 profile is set to "radius", local authentication is disabled. This means if IP connectivity to the RADIUS server is lost or other changes prevent the MXK from receiving RADIUS authentication, the MXK will become inaccessible. Use *radiusthenlocal* to fall back to local authentication when the RADIUS server is not reachable.

```
zSH> update system 0
system  0
Please provide the following: [q]uit.
syscontact: ----------->  {Zhone Global Services and Support 7001 Oakport Street
Oakland Ca. (877) Zhone20 (946-6320) Fax (510)777-7113 support@zhone.com}:
sysname: -------------->  {Zhone MxK}:
syslocation: ---------->  {Oakland}:
enableauthtraps: ------>  {disabled}:
setserialno: ---------->  {0}:
zmsexists: ------------>  {false}:
zmsconnectionstatus: -->  {inactive}:
zmsipaddress: --------->  {0.0.0.0}:
configsyncexists: ----->  {false}:
configsyncoverflow: --->  {false}:
configsyncpriority: --->  {high}:
configsyncaction: ----->  {noaction}:
configsyncfilename: --->  {}:
configsyncstatus: ----->  {syncinitializing}:
configsyncuser: ------->  {}:
configsyncpasswd: ----->  {** private **}:  ** read-only **
numshelves: ----------->  {1}:
shelvesarray: --------->  {}:
numcards: ------------->  {3}:
ipaddress: ------------>  {0.0.0.0}:
alternateipaddress: --->  {0.0.0.0}:
countryregion: -------->  {us}:
primaryclocksource: --->  {0/0/0/0/0}:
ringsource: ----------->  {internalringsourcelabel}:
revertiveclocksource: ->  {true}:
voicebandwidthcheck: -->  {false}:
alarm-levels-enabled: ->  {critical+major+minor+warning}:
userauthmode: --------->  {local}:
radiusauthindex: ------>  {0}: 1
....................
Save changes? [s]ave, [c]hange or [q]uit: s
```

```
Record updated.
```

After completing the RADIUS configuration, the MXK displays console messages for RADIUS login and logout activity.

For users logging in through RADIUS, the system prompt appears as the *username@systemname*. For example, the system prompt for a basic user on a MXK using the default Zhone MXK system name will appear as basicuser@Zhone MXK. The system name is configured using the sysname parameter in the System 0 profile.

2 CONFIGURING BRIDGING

This chapter explains how to configure bridging on the MXK. It includes the following sections:

Overview

This section provided an overview of MXK bridging and the types of MXK bridging:

- *Configuration overview* on page 17
- *Asymmetrical bridging* on page 18
- *Symmetrical bridging* on page 20

Configuration overview

Bridges are configured using the **bridge add** command and the desired bridge type (upl for uplink, dwn for downlink, tls for TLS, hub for hub, and no type for transparent.) This command assigns a **bridge-interface-record** profile to the specified interface.To facilitate bridge setup, the MXK sets the default values for this profile based on the type of bridge specified.

For transparent bridges, the type parameter is omitted to create bridges on the interfaces with default transparent bridge settings. In the **bridge add**, **bridge delete** commands, <slot> and <port> may be replaced with brackets containing numbers in series and/or (dash-separated) ranges; <port> may be replaced with wildcard '*' for all ports on the card.

Refer to the *CLI Reference Guide* for a complete description of the command options and syntax.

Note: When routed and bridged traffic is configured for the same uplink interface, VLAN tags must be used between both downlink ports and the uplink interface for traffic differentiation. For routed traffic, use the **ip-interface-record** profile to specify the VLAN ID.

Bridging involves configuring the MXK to direct traffic based on Ethernet MAC addresses. The MXK supports a variety of asymmetrical and symmetrical bridge types which enable different methods to learn, forward, and manipulate traffic.

Asymmetrical bridging

This section describes the asymmetrical bridge types:

- *Uplink bridges* on page 18
- *Downlink bridges* on page 19
- *Bridge default settings for asymmetric bridges* on page 19

Asymmetrical bridges can utilize VLAN tagging for tagged bridges to provide traffic segregation and no VLANs for untagged traffic without any segregation.

Tagged or Virtual LANs (VLANs) bridging forwards traffic based on MAC addresses and allows the segregation of a single Ethernet network into multiple virtual network segments by VLAN IDs.

Untagged or transparent bridging which forwards traffic based on MAC addresses but does not provide segregation of traffic. Traffic is broadcast over the Ethernet port and is either accepted or rejected based on the destination MAC address. There is no VLAN tagging; all ports are learning and forwarding without restriction without broadcast suppression. Forwarding to a default port is not allowed.

Uplink bridges

An uplink bridge uses one bridge interface in a VLAN as a default, and traffic from all other interfaces exits the system from this interface. As the default interface, packets entering the system on this interface do not have their source MAC addresses learned and associated with this interface. Traffic coming into this uplink interface is sent to the interface where the address has been learned. If the frame is a broadcast, it is filtered, unless it is an ARP or DHCP message that meets some special criteria. Unicasts received on an uplink port are forwarded to the downlink where the MAC address was learned.

Uplink bridge interfaces require an additional bridge-path configuration to set a default path for a specific VLAN for the system onto the uplink bridge. If an uplink is missing this configuration, traffic will not flow across the asymmetric VLAN.

Downlink bridges

A downlink bridge is used in conjunction with an uplink bridge where the uplink bridge is the path upstream to the network, and the downlink bridge is the learning interface facing subscribers. Traffic coming into this interface is forwarded to the uplink regardless of the destination MAC address. Broadcasts and unicasts (known and unknown) will be sent out the default interface, which is the uplink bridge for the VLAN.

Packets entering the system on this interface have their source MAC addresses learned and associated with this interface. Because this interface is not a default, it is required to learn MAC addresses, so that frames from the network that come in on the uplink bridge can be sent to the correct downlink bridge. Broadcasts received on a downlink are sent to the uplink (default) without filtering. Broadcasts will not flow to other downlinks as long as **forwardtodefault** parameter is set to true. Downlink ports learn MAC addresses.

Bridge default settings for asymmetric bridges

Table 2 lists the default bridge-interface-record settings for the supported asymmetric bridge options.

Table 2: Default values for asymmetric bridge-interface-record

Parameter	Uplink	Downlink	Downlink Tagged
vpi	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.
vci	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.
vlanId	0	As specified	As specified
stripAndInsert	False	True	False
customARP	True	False	False

Table 2: Default values for asymmetric bridge-interface-record

Parameter	Uplink	Downlink	Downlink Tagged
filterBroadcast	True	False	False
learnIP	False	True	True
learnUnicast	False	True	True
maxUnicast	0	5	5
learnMulticast	False	True	True
forwardToUnicast	True	False	False
forwardToMulticast	True	False	False
forwardToDefault	False	True	True
floodUnknown	False	False	False
floodMulticast	False	False	False
valndIdCOS	0	0	0
outgoingCOSOption	Disable	Disable	Disable
outgoingCOSValue	0	0	0
s-tagTPID	0x8100	0x8100	0x8100
s-tagId	0	0	0
s-tagStripAndInsert	False	False	False
s-tagOutgoingCOSOption	s-tagdisable	s-tagdisable	s-tagdisable
s-tagIdCOS	0	0	0
s-tagOutgoingCOSValue	0	0	0
mcastControlList			
maxVideoStreams			
isPPPoA			
bridgeIfEgressPacketRule GroupIndex:			
bridgeIfTableBasedFilter			
bridgeIfDhcpLearn			

Symmetrical bridging

This section describes symmetrical bridge types:

- *Transparent LAN service (TLS)* on page 21
- *Hub bridge* on page 21

- *Transparent bridge* on page 21
- *Bridge default settings for symmetric bridges* on page 21

Transparent LAN service (TLS)

A TLS bridge is used with only other TLS bridges. This should not be used with any asymmetrical bridges. TLS bridges learn MAC addresses and forward packets to learned destinations. Broadcasts and unknown unicasts are flooded out all interfaces except the ingress interface.

Packets entering the system on TLS interface have their source MAC addresses learned and associated with the interface so that frames from the network that come in on other TLS bridges in the VLAN can be sent to the correct interface.

Hub bridge

A hub bridge is used with only other hub bridges. Hub bridges do not learn MAC addresses, but flood packets of all types to every other bridge interface in the VLAN, where all ports receive every frame received on the hub interface.

Packets entering the system on this interface do not have their source MAC addresses learned so that frames from the network that come in on other hub bridges in the VLAN can be sent to the correct interface.

Transparent bridge

Transparent or untagged bridges which forward traffic based on MAC addresses but do not provide segregation of traffic. Traffic is broadcast over the Ethernet port and is either accepted or rejected based on the destination MAC address. There is no VLAN tagging; all ports are learning and forwarding without restriction and without broadcast suppression. Forwarding to a default port is not allowed. Only one untagged bridge can be created per port.

Bridge default settings for symmetric bridges

Table 3 lists the default bridge-interface-record settings for the supported symmetric bridge options.

Table 3: Default values for symmetric bridge-interface-record

Parameter	Transparent	TLS	Hub
vpi	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.

Table 3: Default values for symmetric bridge-interface-record

Parameter	Transparent	TLS	Hub
vci	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.	0 for Ethernet interfaces. As specified for other interfaces.
vlanId	0	As specified	As specified
stripAndInsert	True	True	True
customARP	False	False	False
filterBroadcast	False	False	False
learnIP	False	False	True
learnUnicast	True	True	False
maxUnicast	5	100	0
learnMulticast	False	False	False
forwardToUnicast	True	True	False
forwardToMulticast	False	False	False
forwardToDefault	False	False	True
floodUnknown	False	True	True
floodMulticast	False	True	True
bridgeIfCustomDHCP	False	False	False
bridgeIfConfigGroupIndex	0	0	0
valndIdCOS	0	0	0
outgoingCOSOption	Disable	Disable	Disable
outgoingCOSValue	0	0	0
s-tagTPID	0x8100	0x8100	0x8100
s-tagId	0	0	0
s-tagStripAndInsert	False	False	False
s-tagOutgoingCOSOption	s-tagdisable	s-tagdisable	s-tagdisable
s-tagIdCOS	0	0	0
s-tagOutgoingCOSValue	0	0	0
mcastControlList:			
maxVideoStreams			
isPPPoA			

Table 3: Default values for symmetric bridge-interface-record

Parameter	Transparent	TLS	Hub
floodUnknown			
floodMulticast			
bridgeIfEgressPacketRule GroupIndex			
bridgeIfTableBasedFilter			
bridgeIfDhcpLearn			

The **bridge show** command displays the bridge type.

```
zSH> bridge show
Type VLAN       Bridge                        St  Table Data
------------------------------------------------------------------------------------
upl Tagged 998   linkagg-a-1-998/bridge        UP  S VLAN 998 default [U: 3600 sec,
                                              M: 61 sec, I: 30 sec]
dwn Tagged 998   1-9-1-510-gponport-998/bridge  UP  D 00:10:a4:ba:54:ba
                                                  D 01:00:5e:0a:63:0b
tls Tagged 142   linkagg-a-1-142/bridge        UP  D 00:00:0c:07:ac:00
                                                  D 00:30:b6:34:76:30
                                                  D 00:b0:c2:f2:b6:fc
                                                  D 00:b0:c2:f5:54:ee
tls Tagged 142   1-9-1-510-gponport-142/bridge  UP  D 00:0f:8f:e9:23:80
                                                  D 00:1a:6c:f0:29:28
upl Tagged 500   linkagg-a-1-500/bridge        UP  S VLAN 500 default [U: 3600 sec,
                                              M: 150 sec, I: 0 sec]
dwn Tagged 500   1-9-1-510-gponport-500/bridge  UP  D 00:18:39:a0:67:45
dwn Tagged 998   1-9-1-921-gponport-998/bridge  DWN
dwn Tagged 998   1-9-1-922-gponport-998/bridge  UP
tls Tagged 142   1-9-1-531-gponport-142/bridge  UP  D 00:01:47:05:9a:a7
tls Tagged 3843  1-a-1-0-linkAgg-3843/bridge    UP
dwn Tagged 134   1-9-1-510-gponport-134/bridge  UP
```

Broadcasts and bridging

The MXK supports a modified form of broadcast suppression when configured for bridge mode. The MXK configures ports as the entered bridge type.

In general, broadcasts sent from a downlink will traverse the uplink, but will not be sent down other downlinks, even within the same VLAN. This prevents subscribers from maliciously or unintentionally sending or receiving broadcasts between ports on the same system.

Ports configured as uplinks will send broadcasts upstream, but by default will not propagate broadcasts sent from the upstream down to the MXK. The filterBroadcast parameter in the bridge-interface-record profile enables this filtering. This mechanism provides security benefits, as well as reducing unnecessary traffic on low bandwidth interfaces.

One exception to the operational mode described above is ARP broadcast support. When a MXK receives a broadcast frame, it is checked to determine if it is an ARP protocol packet or not. If it is not, it is treated as above. If it is, then the MXK compares and filters the requested IP address with the current forwarding table. If a match is found, the ARP broadcast is forwarded out the interface that has the appropriate host. This host will then reply to the ARP with a standard response. If a match is not found, then the ARP is filtered and it gets dropped as if it were a non-ARP broadcast. This setting is controlled by the customARP parameter.

Another exception to this broadcast filtering is DHCP broadcast support. When a MXK receives a broadcast DHCP OFFER message from a remote DHCP server, if customDHCP is set to true, the broadcast messages are forwarded to the source MAC address. Otherwise, the broadcast DHCP messages are filtered.

✓ **Note:** Ethernet interfaces can be addressed as either *eth* or *ethernetcsmacd*. The *eth* abbreviation is used in command output.

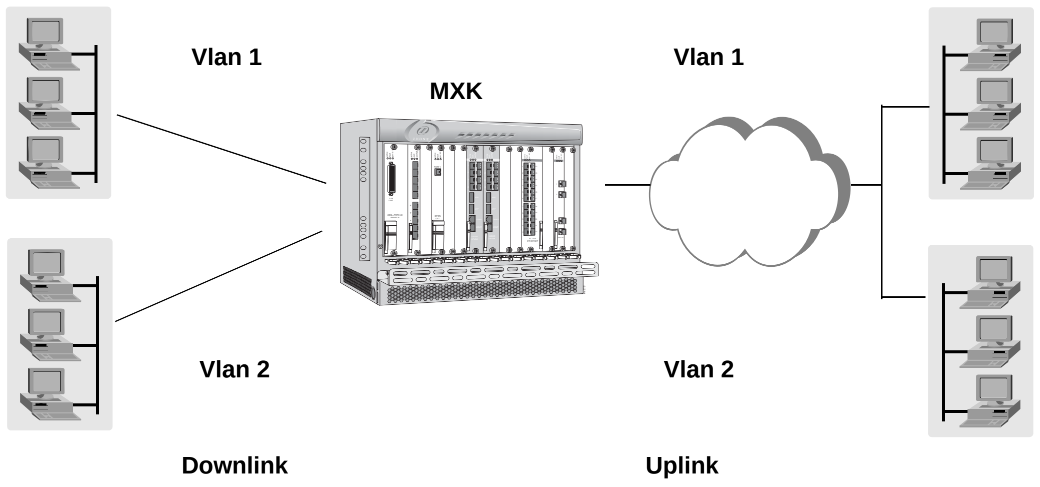
Asymmetrical bridging

This section covers creating basic asymmetrical bridges, uplink bridges and downlink bridges:

- *Create an uplink bridge* on page 25
- *Create a downlink bridge on Ethernet* on page 27
- *Create a downlink bridge on gpon* on page 29

Typical MXK bridging configurations for both uplinks and downlinks designate VLANs. Figure 2 shows a typical VLAN configuration. On the access (subscriber) side, VLANs 1 and 2 are separate Ethernet networks. On the uplink side, VLANs 1 and 2 are on the same physical Ethernet interface, but the traffic is separated based on the VLAN IDs.

Figure 2: Example VLAN network



The side of the connection closest to the subscriber is called the downlink interface. The upstream egress is called the uplink interface. When the MXK is in VLAN mode, it adds (tags) the VLAN ID to the Ethernet frame on the uplink interface and strips (untags) the ID out on the downlink interface. Although VLAN IDs are not typically required on downlink interfaces, you can configure downlink interface as tagged. Tagged downlink interfaces can be used for subtended MXKs or subscribers expecting tagged traffic with Transparent LAN Server (TLS) service.

Note: The MXK supports VLAN IDs from 1 to 4096.

You can configure static VLAN bridge paths, which requires that you enter a MAC address for every bridge on the Ethernet. Or, you can set up the MXK Ethernet interface to learn the VLAN IDs when it receives a packet from a downlink device.

Note that if the MXK receives a packet from an uplink interface before it has learned the VLAN ID or MAC address, it will not deliver the packet.

Create an uplink bridge

Creating an uplink bridge

When creating a uplink bridge all VLANs must be specified. Specifying a VLAN always creates a tagged bridge. even if the parameter *tagged* is not entered.

1 To add a uplink bridge on the MXK uplink, use **bridge add** *interface/type vlan x <tagged>* to add a VLAN interface to the upstream FE/GE interface:

```
zSH> bridge add 1-a-3-0/eth uplink vlan 9
```

Even if the parameter *tagged* is not specified, the uplink bridge is considered a tagged bridge and the bridge will appear as tagged when using **bridge show**.

2 To verify the bridge created, use **bridge show**:

```
zSH> bridge show
Type VLAN        Bridge                        St  Table Data
---------------------------------------------------------------------------------
upl Tagged 9     ethernet3-9/bridge            DWN
```

3 To create the uplink bridge path, enter the **bridge-path add** *interface/bridge vlan x default*:

Caution: For the MXK the *global* parameter is not valid, the parameter *default* must be used. Using the global parameter will create a bridge that will not pass traffic.

```
zSH> bridge-path add ethernet3-9/bridge vlan 9 default
Bridge-path added successfully
```

The *default* setting specifies that the MXK will send all VLAN traffic to this port on the uplink to be passed to the network.

4 To verify the bridge-interface-record settings, enter:

```
zSH> get bridge-interface-record ethernet3-9/bridge
bridge-interface-record  ethernet3-9/bridge
vpi: -------------------------------->  {0}
vci: -------------------------------->  {0}
vlanId: ----------------------------->  {9}
stripAndInsert: --------------------->  {false}
customARP: -------------------------->  {true}
filterBroadcast: -------------------->  {true}
learnIp: ---------------------------->  {false}
learnUnicast: ----------------------->  {false}
maxUnicast: ------------------------->  {0}
learnMulticast: --------------------->  {false}
forwardToUnicast: ------------------->  {true}
forwardToMulticast: ----------------->  {true}
forwardToDefault: ------------------->  {false}
bridgeIfCustomDHCP: ----------------->  {true}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: -------------------------->  {0}
outgoingCOSOption: ------------------>  {disable}
outgoingCOSValue: ------------------->  {0}
s-tagTPID: -------------------------->  {0x8100}
s-tagId: ---------------------------->  {0}
s-tagStripAndInsert: ---------------->  {true}
s-tagOutgoingCOSOption: ------------->  {s-tagdisable}
s-tagIdCOS: ------------------------->  {0}
s-tagOutgoingCOSValue: -------------->  {0}
```

```
mcastControlList: ------------------->  {}
maxVideoStreams: -------------------->  {0}
isPPPoA: ---------------------------->  {false}
floodUnknown: ----------------------->  {false}
floodMulticast: --------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ----------->  {NONE(0)}
bridgeIfDhcpLearn: ------------------>  {NONE(0)}
```

It is recommended not to change the default settings unless advanced bridge configuration is required.

Create a downlink bridge on Ethernet

Creating a downlink bridge on Ethernet

All VLANs must use the <*tagged*> parameter when creating multiple VLANs on a single physical port. Multiple VLANs may be created on the same port to carry different types of traffic, for example, data and video. To configure a VLAN bridge, in this case a downlink:

1 To verify the location of the card before building a bridge, enter **slots**:

```
zSH> slots
Uplinks
 a:*MX TWO TENGIGE EIGHT GIGE (RUNNING)
 b: MX TWO TENGIGE EIGHT GIGE (RUNNING)
Cards
 8: MX LINE CARD 20 ACT ETH (RUNNING)
10: MX LINE CARD 2 PORT GPON (RUNNING)
```

2 To add a tagged bridge on shelf 1 of the Active Ethernet card in slot 8 for downstream subscriber data access, enter **bridge add** *interface/type downlink vlan x <tagged>*:

```
zSH> bridge add 1-8-1-0/eth downlink vlan 777 tagged
Adding bridge on 1-8-1-0/eth
Created bridge-interface-record 1-8-1-0-eth-777/bridge
```

A **bridge-interface-record** is created with the default settings for the bridge depending on the type of bridge, uplink or downlink. Do not change the default settings unless advanced bridging is required.

3 To add another tagged bridge on shelf 1 of the Active Ethernet card in slot 8 for downstream subscriber video access, enter **bridge add** *interface/type downlink vlan x tagged*:

```
zSH> bridge add 1-8-1-0/eth downlink vlan 555 tagged
Adding bridge on 1-8-1-0/eth
Created bridge-interface-record 1-8-1-0-eth-555/bridge
```

A **bridge-interface-record** is created with the default settings for the bridge depending on the type of bridge, uplink or downlink. Do not change the default settings unless advanced bridging is required.

4 To verify the bridges just created, enter **bridge show**:

```
zSH> bridge show
Type VLAN       Bridge                         St  Table Data
---------------------------------------------------------------------------------
tls Tagged 2820  1-b-1-0-linkAgg-2820/bridge    DWN
upl Tagged 2821  1-b-1-0-linkAgg-2821/bridge    DWN S VLAN 2821 default [U: 3600
                                                sec, M: 150 sec, I: 30 sec]
dwn Tagged 777   1-8-1-0-eth-777/bridge         DWN
dwn Tagged 555   1-8-1-0-eth-555/bridge         DWN
```

5 To view the **bridge-interface-record** for a bridge, enter **get bridge-interface-record** *interface/type*:

```
zSH> get bridge-interface-record 1-8-1-0-eth-555/bridge
bridge-interface-record  1-8-1-0-eth-555/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {555}
stripAndInsert: ---------------------->  {false}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {true}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {5}
learnMulticast: ---------------------->  {true}
forwardToUnicast: -------------------->  {false}
forwardToMulticast: ------------------>  {false}
forwardToDefault: -------------------->  {true}
bridgeIfCustomDHCP: ------------------>  {false}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: --------------------------->  {0}
outgoingCOSOption: ------------------->  {disable}
outgoingCOSValue: -------------------->  {0}
s-tagTPID: --------------------------->  {0x8100}
s-tagId: ----------------------------->  {0}
s-tagStripAndInsert: ----------------->  {true}
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}
s-tagIdCOS: -------------------------->  {0}
s-tagOutgoingCOSValue: --------------->  {0}
mcastControlList: -------------------->  {}
maxVideoStreams: --------------------->  {0}
isPPPoA: ----------------------------->  {false}
floodUnknown: ------------------------>  {false}
floodMulticast: ---------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
```

6 To delete a bridge enter **bridge delete** *interface/type*:

```
zSH> bridge delete 1-8-1-0-eth-555/bridge
1-8-1-0-eth-555/bridge Delete complete
```

Create a downlink bridge on gpon

The syntax for entering bridges on GPON ports is *shelf-slot-port-gemport/gponport*. For bridges created on the GPON line card, the *type* entered is *gponport*. For bridges created on the GPON line card, the GEM port is always designated. Valid GPON GEM port numbers are:

- 501-532
- 701-732
- 901-932

Creating a downlink bridge on gpon

1 To add a bridge on GPON port 10, enter:

```
zSH> bridge add 1-10-1-701/gponport downlink vlan 777
Adding bridge on 1-10-1-701/gponport
Created bridge-interface-record 1-10-1-701-gponport/bridge
```

2 To view the bridge just created, enter:

```
zSH> bridge add 1-10-1-701/gponport downlink vlan 777 tagged
Adding bridge on 1-10-1-701/gponport
Created bridge-interface-record 1-10-1-701-gponport-777/bridge
```

Symmetrical TLS bridging

TLS bridges learn MAC addresses and forward packets to learned destinations. Broadcasts and unknown unicasts are flooded out all interfaces except the ingress interface. Packets entering the system on a TLS interface have their source MAC addresses learned and associated with the interface so that frames from the network that come in on other TLS bridges in the VLAN can be sent to the correct interface. A TLS bridge is used only with other TLS bridges.

The most important type of symmetrical bridging is a TLS bridge. TLS bridges can only be used with other TLS bridges.

Creating a TLS bridge on a Active Ethernet card

1 To create a TLS bridge on the MXK Active Ethernet card in slot 4 on port 4, enter:

```
zSH> bridge add 1-6-4-0/eth tls vlan 999
Adding bridge on 1-6-4-0/eth
Created bridge-interface-record 1-6-4-0-eth/bridge
```

2 To delete the TLS bridge, enter:

```
zSH> bridge delete 1-6-4-0-eth/bridge vlan 999
1-6-4-0-eth/bridge Delete complete
```

Creating a TLS bridge on a FE/GE uplink card

1 To create a TLS bridge on the 10 FE/GE port in slot a port 2, enter:

```
zSH> bridge add 1-a-2-0/eth tls vlan 99
Adding bridge on 1-a-2-0/eth
Created bridge-interface-record ethernet2/bridge
```

2 To create a TLS bridge on the 1 FE/GE port in slot 6 port 4, enter:

```
zSH> bridge add 1-6-4-0/eth tls vlan 999
Adding bridge on 1-6-4-0/eth
Created bridge-interface-record 1-6-4-0-eth/bridge
```

Link aggregation bridging

This section describes link aggregation bridging:

- *Create a uplink bridge with link aggregation* on page 30
- *Create a TLS bridge with link aggregation* on page 31

Ethernet ports can be bonded together into groups on the Ethernet uplink card. Link aggregation bridge type is supported on uplink bridges and TLS bridges.

See *Chapter 4, Configuring Link Aggregation* for more information on link aggregation.

Create a uplink bridge with link aggregation

Creating an uplink bridge with link aggregation

1 To verify link aggregation groups, enter:

```
zSH> linkagg show
LinkAggregations:
slot    unit    ifName          admin   numLinks
------------------------------------------------
 a      1       1-a-1-0         up      1
        link: 1-a-2-0   slot a  port 2  subport 0       admin up
 b      1       1-b-1-0         up      1
        link: 1-b-2-0   slot b  port 2  subport 0       admin up
```

2 To create an uplink bridge with link aggregation, enter:

```
zSH> bridge add 1-a-1-0/linkagg uplink vlan 333 tagged
Adding bridge on 1-a-1-0/linkagg
Created bridge-interface-record 1-a-1-0-linkAgg-333/bridge
```

3 To delete a bridge with link aggregation enter:

```
zSH> bridge delete 1-a-1-0-linkAgg-333/bridge vlan 333
1-a-1-0-linkAgg-333/bridge Delete complete
```

Create a TLS bridge with link aggregation

Creating a TLS bridge with link aggregation

You can create bridges with link aggregation using both *linkagg* bridge type or *eth* bridge type. If *eth* bridge type is used on a Ethernet port that has been aggregated, the bridge type automatically changes to *linkagg*.

1 To create a TLS bridge with link aggregation, enter:

```
zSH> bridge add 1-a-1-0/linkagg tls vlan 777
Adding bridge on 1-a-1-0/linkagg
Created bridge-interface-record 1-a-1-0-linkAgg/bridge
```

2 To verify the bridge created, enter:

```
zSH> bridge show
Type VLAN       Bridge                       St  Table Data
--------------------------------------------------------------------------------
tls         777 1-a-1-0-linkAgg/bridge       UP
```

3 To create a TLS bridge and the *eth* bridge type is used, enter:

```
zSH> bridge add 1-a-2-0/eth tls vlan 888
Adding bridge on 1-a-2-0/eth
Created bridge-interface-record linkagg-a-1/bridge
```

The bridge type automatically changes to *linkagg*.

4 To verify the bridge created, enter:

```
zSH> bridge show
Type VLAN       Bridge                       St  Table Data
--------------------------------------------------------------------------------
tls         888 linkagg-a-1/bridge           UP
```

5 To delete the bridge, enter:

```
zSH> bridge delete linkagg-a-1/bridge vlan 888
linkagg-a-1/bridge Delete complete
```

Bridging behavior for tagged, s-tagged, and untagged

This section describes how bridges utilize VLAN and SLAN tagging for untagged, tagged, and s-tagged, traffic segregation:

- *Tagged bridging* on page 32
- *s-tagged bridging* on page 32

- *Untagged bridging* on page 32
- *Bridging examples* on page 35

Tagged bridging

Tagged or Virtual LAN (VLAN) bridging, accepts single-tagged packets based on MAC addresses and allows the segregation of a single Ethernet network into multiple virtual network segments by mapping packets based on the VLAN ID. If a non-zero VLAN ID is configured, the interface accepts only tagged packets matching this VLAN ID. If a VLAN of 0 (zero) is configured, the interface accepts all VLAN tagged packets not matching any configured VLANs on the same interface.

s-tagged bridging

Double-tagged or Service LANs (SLANs) bridging, accepts and sends double-tagged traffic based on MAC addresses and allows the segregation of a single Ethernet network into multiple virtual network segments by mapping packets based on VLAN ID and SLAN ID. If non-zero VLAN ID and SLAN ID are configured, the interface accepts and sends only tagged packets matching both VLAN ID and SLAN ID. If a VLAN of 0 (zero) is configured with a non-zero SLAN ID, the interface accepts and sends only double-tagged packets matching the SLAN and any VLAN tagged packets not destined to another client on the same interface.

When both the VLAN and SLAN tags are zero (0), the bridge accepts all single or double tagged packets not destined to another client on the same interface.

A configured SLAN tag is inserted into outgoing packets when bridge forwarding selects a double-tagged egress interface. Only non-zero SLAN values are recommended for tagged bridges.

Untagged bridging

Untagged or transparent bridging accepts and sends traffic based on MAC addresses but does not provide traffic segregation. Traffic is broadcast over the Ethernet port and is either accepted or rejected based on the destination MAC address. There is no VLAN tagging; all ports are learning and forwarding without restriction, without broadcast suppression. Forwarding to a default port is not allowed. If bridge forwarding selects a single or double-tagged egress interface, the configured VLAN and SLAN tags will be inserted in to packets destined for this interface. Only non-zero values are recommended for VLAN and SLAN settings of untagged bridges.

Configuring untagged or transparent bridging enables you to forward traffic from a downlink interface through the MXK uplink interface based on the destination MAC address without tagging or modification to the frame. Refer to the *CLI Reference Guide* for a complete description of the command options and syntax.

✓ **Note:** Ethernet interfaces can be addressed as either *eth* or *ethernetcsmacd*. The *eth* abbreviation is used in command output.

Configuring an untagged bridge

To add an untagged bridge:

1 Add an untagged bridge to a downstream Ethernet interface:

```
zSH> bridge add 1-6-5-0/eth
Adding bridge on 1-6-5-0/eth
Created bridge-interface-record 1-6-5-0-eth/bridge
```

This example adds a transparent bridge interface to the Active Ethernet card on shelf 1, slot 8, port 5 and sets the parameters to the default transparent bridge interface settings.

The following examples shows the default **bridge-interface-record** settings. It is recommended not to change the default settings unless advanced bridge configuration is required.

```
zSH> get bridge-interface-record 1-6-5-0-eth/bridge
bridge-interface-record  1-6-5-0-eth/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {0}
stripAndInsert: ---------------------->  {true}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {false}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {5}
learnMulticast: ---------------------->  {false}
forwardToUnicast: -------------------->  {true}
forwardToMulticast: ------------------>  {false}
forwardToDefault: -------------------->  {false}
bridgeIfCustomDHCP: ------------------>  {false}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: --------------------------->  {0}
outgoingCOSOption: ------------------->  {disable}
outgoingCOSValue: -------------------->  {0}
s-tagTPID: --------------------------->  {0x8100}
s-tagId: ----------------------------->  {0}
s-tagStripAndInsert: ----------------->  {true}
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}
s-tagIdCOS: -------------------------->  {0}
s-tagOutgoingCOSValue: --------------->  {0}
mcastControlList: -------------------->  {}
maxVideoStreams: --------------------->  {0}
isPPPoA: ----------------------------->  {false}
floodUnknown: ------------------------>  {false}
```

```
floodMulticast: ---------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
```

2 To view the transparent (untagged) bridge just created enter:

```
zSH> bridge show
Type VLAN        Bridge                        St  Table Data
---------------------------------------------------------------------------------
upl Tagged 1201  ethernet2-1201/bridge         UP  S VLAN 1201 default [U: 3600
                                                    sec, M: 150 sec, I: 0 sec]
dwn Tagged 1201  1-10-1-501-gponport-1201/bridg UP
    Untagged     1-6-5-0-eth/bridge            DWN
```

3 Add a transparent bridge to the upstream Ethernet interface:

```
zSH> bridge add 1-a-1-0/linkagg
Adding bridge on 1-a-1-0/linkagg
Created bridge-interface-record linkagg-a-1/bridge
```

This command adds a bridge that accepts transparent/untagged traffic on the MXK units's egress Ethernet port.

4 Verify that both sides of the bridge are present:

```
zSH> bridge show
Type VLAN        Bridge                        St  Table Data
---------------------------------------------------------------------------------
    Untagged     1-6-5-0-eth/bridge            DWN
    Untagged     linkagg-a-1/bridge            UP
```

5 To view the default transparent **bridge-interface-record** settings for the uplink enter:

```
zSH> get bridge-interface-record linkagg-a-1/bridge
bridge-interface-record  linkagg-a-1/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {0}
stripAndInsert: ---------------------->  {true}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {false}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {5}
learnMulticast: ---------------------->  {false}
forwardToUnicast: -------------------->  {true}
forwardToMulticast: ------------------>  {false}
forwardToDefault: -------------------->  {false}
bridgeIfCustomDHCP: ------------------>  {false}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: --------------------------->  {0}
outgoingCOSOption: ------------------->  {disable}
outgoingCOSValue: -------------------->  {0}
```

```
s-tagTPID: ---------------------------> {0x8100}
s-tagId: -----------------------------> {0}
s-tagStripAndInsert: -----------------> {true}
s-tagOutgoingCOSOption: --------------> {s-tagdisable}
s-tagIdCOS: --------------------------> {0}
s-tagOutgoingCOSValue: ---------------> {0}
mcastControlList: --------------------> {}
maxVideoStreams: ---------------------> {0}
isPPPoA: -----------------------------> {false}
floodUnknown: ------------------------> {false}
floodMulticast: ----------------------> {false}
bridgeIfEgressPacketRuleGroupIndex: --> {0}
bridgeIfTableBasedFilter: ------------> {NONE(0)}
bridgeIfDhcpLearn: -------------------> {NONE(0)}
```

It is recommended not to change the default settings unless advanced bridge configuration is required.

Bridging examples

For VLAN tagged (single tagged) bridges, the bridge interface name includes the VLAN ID, even the default VLAN ID of 0. Other examples of a single tagged bridge also show a tagged bridge with VLAN 4000 and a tagged bridge with VLAN 1000 and SLAN 17.

```
zSH> bridge add 1-3-5-0/eth tagged
zSH> bridge add 1-3-5-0/eth vlan 4000 tagged
zSH> bridge add 1-3-5-0/eth vlan 1000 slan 17 tagged

zSH> bridge show
Typ VLAN         Bridge                         State   Table Data
------------------------------------------------------------------------------
    Tagged       1-3-5-0-eth-0/bridge           PENDING
    Tagged 4000  1-3-5-0-eth-4000/bridge        PENDING
    Tg 1000/17   1-3-5-0-eth-1000/bridge        PENDING
```

For VLAN and SLAN tagged (double tagged) bridges, the bridge interface name includes the VLAN ID and SLAN ID, even the default VLAN ID of 0 and the default SLAN of 0. Other examples of doubled tagged bridges also show a bridge with VLAN 4094 and SLAN 4094, a bridge with VLAN 0 and SLAN 17, and a bridge with VLAN 500 and default SLAN.

```
zSH> bridge add 1-3-5-0/eth vlan 4094 slan 4094 s-tagged
zSH> bridge add 1-3-5-0/eth vlan 0 slan 17 s-tagged
zSH> bridge add 1-3-5-0/eth s-tagged
zSH> bridge add 1-3-5-0/eth vlan 500 s-tagged
```

```
zSH> bridge show
Typ VLAN         Bridge                         State   Table Data
------------------------------------------------------------------------------
    ST 4094/4094 1-3-5-0-eth-4094-4094/bridge   PENDING
    ST    0/17   1-3-5-0-eth-0-17/bridge        PENDING
```

```
s-tagged      1-3-5-0-eth-0-0/bridge         PENDING
Tagged 500   1-3-5-0-eth-500-0/bridge       PENDING
```

Bridges can be deleted by specified VLAN ID, SLAN ID, type of tagging, and *all* option. Specifying a VLAN ID all single and double tagged bridges configured for that VLAN.

To delete a bride by a specific SLAN tag:

```
zSH> bridge delete 1-3-5-0/eth slan 17
```

To delete a bridge by a specific VLAN tag or tag type:

```
zSH> bridge delete 1-3-5-0/eth vlan 500
```

```
zSH> bridge delete 1-3-5-0/eth tagged
```

To delete all s-tagged bridges on a port:

```
zSH> bridge delete 1-3-5-0/eth s-tagged all
```

To delete all VLAN 0 bridges on a port:

```
zSH> bridge delete 1-3-5-0/eth vlan 0 all
```

Q-in-Q

This section describes Q-in-Q for the MXK:

- *MXK Q-in-Q overview* on page 36
- *Configure Q-in-Q using the interface command* on page 37
- *Configuring Q-in-Q using the bridge command* on page 39
- *Bridge-path enhancements* on page 41

MXK Q-in-Q overview

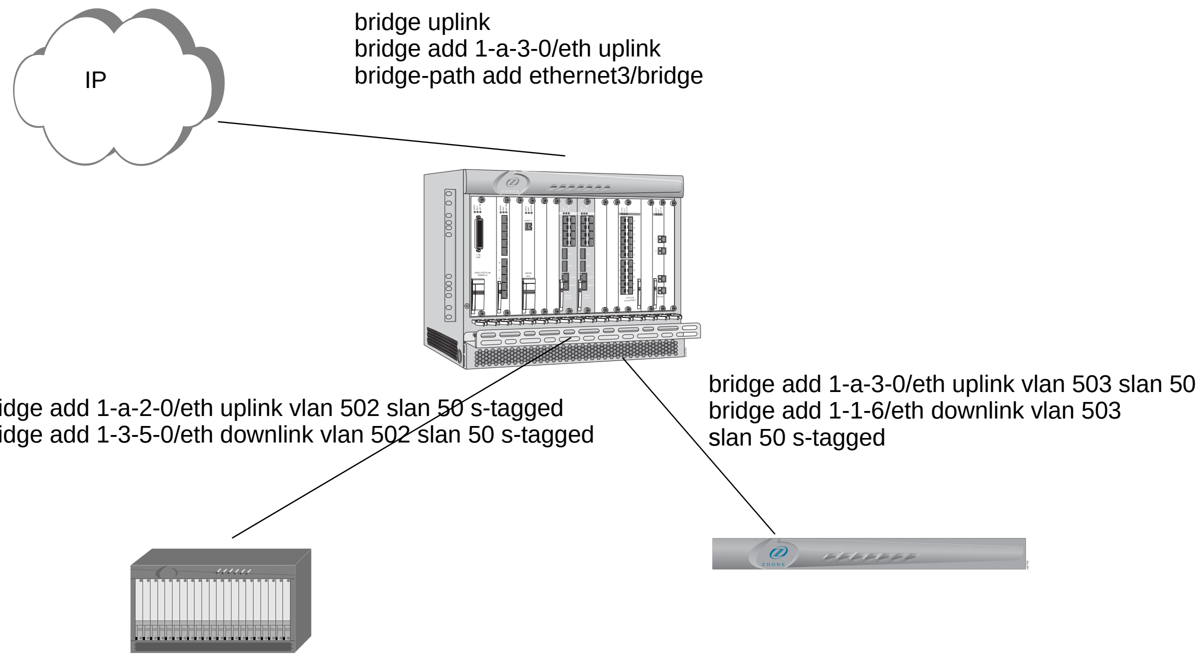
The IEEE 802.1Q-in-Q VLAN tagging expands the VLAN space in the Ethernet frame to support the tagging of previously tagged packets. This second tag (SLAN) creates a "double-tagged" Ethernet frame. The double-tagged Ethernet frame enables service providers to offer additional services, such as Internet access on specific SLANs for specific customers, while still providing single-tagged VLAN services.

The MXK also supports setting COS values in the Ethernet SLAN headers for bridged packets. This service enables you to assign a service level or class of service (COS) to an Ethernet SLAN that is transported across a uplink or downlinked s-tagged bridge. The configured COS level specifies the packet priority and queueing methods used to transport the packet through the Ethernet network. The MXK sets and preserves the COS settings to ensure these settings are passed to other Ethernet devices in the network for QOS processing.

Note: Ethernet interfaces can be addressed as either *eth* or *ethernetcsmacd*. The *eth* abbreviation is used in command output.

Figure 3 illustrates a network of MXK devices configured to support separate SLANs per MXK while also providing individual VLANs per customer port.

Figure 3: Q-in-Q Bridging


IP
bridge uplink
bridge add 1-a-3-0/eth uplink
bridge-path add ethernet3/bridge
bridge add 1-a-2-0/eth uplink vlan 502 slan 50 s-tagged
bridge add 1-3-5-0/eth downlink vlan 502 slan 50 s-tagged
bridge add 1-a-3-0/eth uplink vlan 503 slan 50
bridge add 1-1-6/eth downlink vlan 503
slan 50 s-tagged

Configure Q-in-Q using the interface command

For Q-in-Q VLAN tagging, the interface profile supports the following parameters:

- s-tagTPID

  Identifies the type of VLAN ID used. Typically set to 8100.

- s-tagID

  Specifies the SLAN ID assigned to an Ethernet frame.

- s-tagIDCOS

  Specifies the COS ID associated with the SLAN ID

The **interface** command supports adding s-tagIDs from the command line.

Creating an interface profile

1 To add an interface ethernet1 with VLAN 100, SLAN 200, COS value of 7 and sCOS value of 7 enter:

```
zSH> interface add ethernet1/ip vlan 100 slan 200 cos
7 scos 7 172.16.88.46 255.255.255.0
Created ip-interface-record ethernet1-100/ip.
```

2 To view the interface just created enter:

```
zSH> interface show
2 interfaces
Interface      Status  Rd/Address          Media/Dest Address     IfName
--------------------------------------------------------------------------------
1/a/1/0/ip     UP      1 172.24.64.92/24   00:01:47:13:44:56      ethernet1
1/a/1/0/ip     UP      1 172.16.88.46/24   00:01:47:13:44:56      ethernet1-100
--------------------------------------------------------------------------------
```

3 To view the ip-interface-record of the interface just created enter:

```
zSH> get ip-interface-record ethernet1-100/ip
ip-interface-record  ethernet1-100/ip
vpi: ------------------------->  {0}
vci: ------------------------->  {0}
rdindex: --------------------->  {1}
dhcp: ------------------------>  {none}
addr: ------------------------>  {172.16.88.46}
netmask: --------------------->  {255.255.255.0}
bcastaddr: ------------------->  {172.16.88.255}
destaddr: -------------------->  {0.0.0.0}
farendaddr: ------------------>  {0.0.0.0}
mru: ------------------------->  {1500}
reasmmaxsize: ---------------->  {0}
ingressfiltername: ----------->  {}
egressfiltername: ------------>  {}
pointtopoint: ---------------->  {no}
mcastenabled: ---------------->  {yes}
ipfwdenabled: ---------------->  {yes}
mcastfwdenabled: ------------->  {yes}
natenabled: ------------------>  {no}
bcastenabled: ---------------->  {yes}
ingressPacketRuleGroupIndex: ->  {0}
egressPacketRuleGroupIndex: -->  {0}
ipaddrdynamic: --------------->  {static}
dhcpserverenable: ------------>  {false}
subnetgroup: ----------------->  {0}
unnumberedindex: ------------->  {0}
mcastcontrollist: ------------>  {}
vlanid: ---------------------->  {100}
maxVideoStreams: ------------->  {0}
tosOption: ------------------->  {disable}
tosCOS: ---------------------->  {0}
vlanCOS: --------------------->  {7}
s-tagTPID: ------------------->  {0x8100}
```

```
s-tagId: -------------------->  {200}
s-tagIdCOS: ----------------->  {7}
```

4 To delete an interface enter:

```
zSH> interface delete ethernet1-100/ip vlan 100
Delete complete
```

5 To verify the deletion:

```
zSH> interface show
1 interface
Interface      Status  Rd/Address           Media/Dest Address      IfName
--------------------------------------------------------------------------------
1/a/1/0/ip     UP      1 172.24.64.92/24    00:01:47:13:44:56       ethernet1
--------------------------------------------------------------------------------
```

Configuring Q-in-Q using the bridge command

For Q-in-Q VLAN tagging, the bridge profile supports the following parameters:

- s-tagTPID

  Identifies the type of VLAN ID used. Typically set to 8100.
- s-tagID

  Specifies the SLAN ID assigned to an Ethernet frame.
- s-tagStripAndInsert

  Specifies whether to strip and insert s-tag values in Ethernet frames received and transmitted on the bridge interface.
- s-tagOutgoingCOSOption

  Specifies whether to insert COS value bits on outgoing s-tag packets.
- s-tagIDCOS

  Specifies the COS ID associated with the SLAN ID
- s-tagOutgoingCOSValue

  Specifies the value used to overwrite any existing COS value in outgoing s-tag packets.

Syntax **bridge add**

Adding s-tagIDs with the bridge command

The **bridge** command supports adding s-tagIDs from the command line.

1 To add a interface 1-8-8-0/eth with VLAN 100, SLAN 200, COS value of 7 and SCOS value of 7 enter:

```
zSH> **bridge add 1-8-8-0/eth downlink vlan 100 slan 200 tagged cos 7 scos 7**
Adding bridge on 1-8-8-0/eth
Created bridge-interface-record 1-8-8-0-eth-100/bridge
```

2 To verify the bridge just created enter:

```
zSH> bridge show
Type VLAN       Bridge                       St  Table Data
--------------------------------------------------------------------------------
    Untagged    1-8-5-0-eth/bridge           DWN
dwn Tg  100/200 1-8-8-0-eth-100/bridge       DWN
```

3 To view the bridge-interface-record enter:

```
zSH> get bridge-interface-record 1-8-8-0-eth-100/bridge
bridge-interface-record  1-8-8-0-eth-100/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {100}
stripAndInsert: ---------------------->  {false}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {true}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {5}
learnMulticast: ---------------------->  {true}
forwardToUnicast: -------------------->  {false}
forwardToMulticast: ------------------>  {false}
forwardToDefault: -------------------->  {true}
bridgeIfCustomDHCP: ------------------>  {false}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: --------------------------->  {7}
outgoingCOSOption: ------------------->  {disable}
outgoingCOSValue: -------------------->  {0}
s-tagTPID: --------------------------->  {0x8100}
s-tagId: ----------------------------->  {200}
s-tagStripAndInsert: ----------------->  {true}
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}
s-tagIdCOS: -------------------------->  {7}
s-tagOutgoingCOSValue: --------------->  {0}
mcastControlList: -------------------->  {}
maxVideoStreams: --------------------->  {0}
isPPPoA: ----------------------------->  {false}
floodUnknown: ------------------------>  {false}
floodMulticast: ---------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
```

Bridge-path enhancements

For Q-in-Q VLAN tagging, the bridge-path profile supports the s-tagID parameter to specifies the SLAN ID assigned to an Ethernet frame in static bridge configurations.

Adding a bridge-path s-tagID from the command line

The **bridge-path** command supports adding s-tagIDs from the command line.

1 To first create the bridge enter:

```
zSH> bridge add 1-a-3-0/eth uplink vlan 300 slan 400
Adding bridge on 1-a-3-0/eth
Created bridge-interface-record linkagg-a-1-300/bridge
```

Note: When creating a bridge on an Ethernet port that is a bonded Ethernet port using the *eth* type, the system automatically changes the type to *linkagg*.

2 To verify the bridge just created enter:

```
zSH> bridge show
Type VLAN        Bridge                          St  Table Data
---------------------------------------------------------------------------
    Untagged     1-8-5-0-eth/bridge              DWN
dwn Tg  100/200  1-8-8-0-eth-100/bridge          DWN
upl Tg  300/400  linkagg-a-1-300/bridge          UP
```

3 To create the bridge-path with the vlan and the slan enter:

```
zSH> bridge-path add linkagg-a-1-300/bridge vlan 300 slan 400 default
Bridge-path added successfully
```

4 To view bridge-path interface records with vlan and slan values enter:

```
zSH> bridge-path show
VLAN/SLAN   Bridge                               Address
--------------------------------------------------------------
  300/400   linkagg-a-1-300/bridge               Default
```

COS in bridges

The MXK supports setting COS values in Ethernet VLAN headers for bridged packets:

- *COS bridge interface record profile* on page 42
- *Add COS to an interface* on page 43
- *Add COS to a bridge* on page 43

This service enables you to assign a service level or class of service (COS) to an Ethernet VLAN interface that is transported across a uplink or downlinked tagged bridge. The configured COS level specifies the packet priority and queueing methods used to transport the packet through the Ethernet network. The MXK sets and preserves the COS settings to ensure these settings are passed to other Ethernet devices in the network for QOS processing.

COS bridge interface record profile

The following parameters in the bridge interface record are used for Ethernet COS support.

Parameter	Description
vlanIdCOS	Specifies the value loaded into the COS field of the VLAN header when an untagged packet received on this interface is tagged (VLAN ID inserted) for bridging. Value range is 0 to 7. Default is 0.
outgoingCOSOption	Specifies whether to insert the VLAN COS bits on packets bridged through this interface. Values: **Disable** Leave any existing COS values unchanged. This is the default value. **All** Replace the current COS values in all VLAN headers in tagged and untagged packets originating and transported through this device.
outgoingCOSValue	For outgoing tagged packets, specifies the value used to overwrite any existing COS value in the VLAN header. Value range is 0 to 7. Default is 0.

To display the bridge-record profile, enter the **show bridge-interface-record** command.

```
rpr-uplink-zSH> show bridge-interface-record
vpi:---------------------->   {0}
vci:---------------------->   {0}
vlanId:------------------->   {0 - 2147483647}
stripAndInsert:----------->   false  true
customARP:---------------->   false  true
filterBroadcast:---------->   false  true
learnIp:------------------>   false  true
learnUnicast:------------->   false  true
maxUnicast:--------------->   {0 - 2147483647}
learnMulticast:----------->   false  true
forwardToUnicast:--------->   false  true
forwardToMulticast:------->   false  true
forwardToDefault:--------->   false  true
bridgeIfCustomDHCP:------->   false  true
bridgeIfConfigGroupIndex:->   {0 - 2147483647}
```

```
vlanIdCOS:---------------->   {0 - 7}
outgoingCOSOption:-------->   disable  all
outgoingCOSValue:--------->   {0 - 7}
```

Add COS to an interface

Adding an interface

This example adds interface 1-8-8-0/eth with a COS value of 7.

```
zSH> interface add 1-8-8-0/eth other vlan 1 cos 7 23.23.23.23. 255.255.255.0
Created ip-interface-record 1-8-8-0-eth-1/ip.
```

This example adds interface 1-8-8-0/eth with a COS value of 1 and specifies to add this value to all packets originating from this interface.

```
zSH> interface add 1-8-8-0/eth other vlan 1 cos 1 tosOrig 1 23.23.23.23.
255.255.255.0
Created ip-interface-record 1-8-8-0-eth-1/ip.
```

Add COS to a bridge

Adding a bridge

This example adds interface 1-6-3-0/eth with a vlanIDCOS value of 7. This value is inserted into the priority field of the VLAN header when an untagged packet received on this interface is tagged (VLAN ID inserted) for bridging.

```
zSH> bridge add 1-6-3-0/eth downlink vlan 100 tagged cos 7
Adding bridge on 1-6-3-0/eth
Created bridge-interface-record 1-6-3-0-eth-100/bridge
```

This example adds interface 1-10-1-530/gponport with a vlanIDCOS value of 7 and enables the overwriting of the VLAN ID in all outgoing packets with the value of 7.

```
zSH> bridge add 1-10-1-530/gponport downlink vlan 101 tagged cos 7 outcosall 7
Adding bridge on 1-10-1-530/gponport
Created bridge-interface-record 1-10-1-530-gponport-101/bridge
```

Video bridging

Video bridging on the MXK provides the ability to integrate video streams for multiple sources into one conduit. Video bridging enables video packets to be forwarded over a Layer 2 bridge from a host to a subscriber. As a result, the video travels from its source, or head-end device, and passes through the MXK in a passive manner with only one video stream across the backplane, reducing bandwidth required for video packets to traverse a MXK.

Video bridging requires you to configure both an uplink bridge and a downlink bridge. On the uplink bridge, the forwardToMulticast function is associated with a location that contains video content and allows the MXK to receive video groups from the network. An interface with this value set to true should only transmit multicast traffic for which a JOIN request has been received. Any bridge interface with the forwardToMulticast parameter set to false discards multicast IP traffic. By default, the forwardToMulticast parameter is set to true on uplink bridges.

On the downlink bridge, the learnMulticast function is associated with interfaces that have hosts connected to them and allows the MXK to send video groups from downlink interfaces to the network. By default, the learnMulticast parameter is set to true on downlink bridges.

Note that JOIN operations enter on a learnMulticast interface associated with a downlink bridge and pass through on a forwardToMulticast interface associated with an uplink bridge.

The following table details various video bridge behaviors associated with different combinations of settings for the bridge parameters.

Table 4: learnMulticast-forwardToMulticast Combinations and Behavior

learnMulticast	forwardToMulticast	Behavior
False	False	The interface discards all incoming multicast packets and does not forward any of the packets.
True	False	The interface forwards both default multicast signaling packets an control multicast packets.
True	False	The interface discards incoming multicast content groups and forwards requested content groups.
False	True	The interface forwards control packets received on this interface to all other interfaces that have the **learnMulticast** field set to **true**.
False	True	The interface forwards content groups only to interfaces that have sent JOIN messages for a group.
True	True	Treat the same as an interface with the **learnMulticast** field set to **false** and the **forwardToMulticast** field set to **true**.

The following video bridge example creates a video bridge on a uplink card using the uplink GigE interface as the uplink bridge. When creating the bridge path on that interface enter the multicast aging period and the IGMP query interval.

Create the uplink bridge:

```
zSH> **bridge add 1-a-3-0/eth uplink vlan 77**
Adding bridge on 1-a-3-0/eth
Created bridge-interface-record ethernet3-77/bridge
```

Add the bridge path and a multicast aging period and IGMP query interval.

```
zSH> bridge-path add ethernet3-77/bridge vlan 77 default
mcast 90 igmpqueryinterval 30
Bridge-path added successfully
```

For the downlink bridge, add a downlink bridge and specify a maximum number of video streams and multicast control list. To do so, add the values for the multicast control list and the maximum video streams in the *m/n* format. Set the multicast control list first and the maximum video streams second. Members of the multicast control list must be defined to receive the video signal.

```
zSH> bridge add 1-6-3-0/eth downlink vlan 88 video 1/2
Adding bridge on 1-6-3-0/eth
Created bridge-interface-record 1-6-3-0-eth/bridge
```

To verify bridge settings, use the **get bridge-interface-record** command for each bridge. This command displays the bridge settings, including the learnMulticast and forwardToMulticast.

For the uplink bridge, note that the forwardToMulticast setting is true and the learnMulticast setting is false.

```
zSH> get bridge-interface-record ethernet3-77/bridge
bridge-interface-record  ethernet3-77/bridge
vpi: ------------------------------->  {0}
vci: ------------------------------->  {0}
vlanId: ---------------------------->  {77}
stripAndInsert: -------------------->  {false}
customARP: ------------------------->  {true}
filterBroadcast: ------------------->  {true}
learnIp: --------------------------->  {false}
learnUnicast: ---------------------->  {false}
maxUnicast: ------------------------>  {0}
learnMulticast: -------------------->  {false}
forwardToUnicast: ------------------>  {true}
forwardToMulticast: ---------------->  {true}
forwardToDefault: ------------------>  {false}
bridgeIfCustomDHCP: ---------------->  {true}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: ------------------------->  {0}
outgoingCOSOption: ----------------->  {disable}
outgoingCOSValue: ------------------>  {0}
s-tagTPID: ------------------------->  {0x8100}
s-tagId: --------------------------->  {0}
```

```
s-tagStripAndInsert: ----------------->  {true}
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}
s-tagIdCOS: ------------------------->  {0}
s-tagOutgoingCOSValue: --------------->  {0}
mcastControlList: -------------------->  {}
maxVideoStreams: --------------------->  {0}
isPPPoA: ----------------------------->  {false}
floodUnknown: ------------------------>  {false}
floodMulticast: ---------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
```

For the downlink bridge, note that the forwardToMulticast setting is false and the learnMulticast setting is true.

```
zSH> get bridge-interface-record 1-6-3-0-eth/bridge
bridge-interface-record  1-6-3-0-eth/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {88}
stripAndInsert: ---------------------->  {true}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {true}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {5}
learnMulticast: ---------------------->  {true}
forwardToUnicast: -------------------->  {false}
forwardToMulticast: ------------------>  {false}
forwardToDefault: -------------------->  {true}
bridgeIfCustomDHCP: ------------------>  {false}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: --------------------------->  {0}
outgoingCOSOption: ------------------->  {disable}
outgoingCOSValue: -------------------->  {0}
s-tagTPID: --------------------------->  {0x8100}
s-tagId: ----------------------------->  {0}
s-tagStripAndInsert: ----------------->  {true}
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}
s-tagIdCOS: ------------------------->  {0}
s-tagOutgoingCOSValue: --------------->  {0}
mcastControlList: -------------------->  {1}
maxVideoStreams: --------------------->  {2}
isPPPoA: ----------------------------->  {false}
floodUnknown: ------------------------>  {false}
floodMulticast: ---------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
```

In addition, you can run a **bridge igmp** command to determine whether IGMP is running on the system.

```
zSH> bridge igmp
VlanID MAC Address       MCAST IP         Ifndx Host MAC          Last Join
---------------------------------------------------------------------------
   999 01:00:5e:02:7f:fe 224.2.127.254      921 00:02:02:0b:4a:a0         2
   999 01:00:5e:02:7f:fe 224.2.127.254      922 00:02:02:0a:bb:6d       106
   999 01:00:5e:02:7f:fe 224.2.127.254      923 00:02:02:0a:c0:b7        87
   999 01:00:5e:02:7f:fe 224.2.127.254      924 00:02:02:0b:4e:c5       172
   999 01:00:5e:02:7f:fe 224.2.127.254      925 00:02:02:0b:4c:7e        65
   999 01:00:5e:02:7f:fe 224.2.127.254      926 00:02:02:0b:4f:08        46
   999 01:00:5e:02:7f:fe 224.2.127.254      927 00:02:02:09:c1:7d        90
   999 01:00:5e:02:7f:fe 224.2.127.254      928 00:02:02:0b:44:cd        71
   999 01:00:5e:02:7f:fe 224.2.127.254      929 00:02:02:0b:4c:ca        61
   999 01:00:5e:02:7f:fe 224.2.127.254      930 00:02:02:0b:47:bd         7
   999 01:00:5e:02:7f:fe 224.2.127.254      931 00:02:02:0b:47:c7       177
   999 01:00:5e:02:7f:fe 224.2.127.254      932 00:02:02:0b:4d:35       181
   999 01:00:5e:02:7f:fe 224.2.127.254      933 00:02:02:0b:4d:5b       144
   999 01:00:5e:02:7f:fe 224.2.127.254      934 00:02:02:0b:4a:a5        59
   999 01:00:5e:02:7f:fe 224.2.127.254      935 00:02:02:0b:4c:9e         3
   999 01:00:5e:02:7f:fe 224.2.127.254      936 00:02:02:09:c1:78         6
   999 01:00:5e:02:7f:fe 224.2.127.254      937 00:02:02:0a:c0:ca       131
```

Bridge enhancements to flood unknowns and multicasts

Bridges are now enhanced to enable VPN-like services using the floodUnknowns and floodMulticast parameters. These parameters enable the MXK to forward unknown traffic to all bridge interfaces within the VLAN.

FloodUnknown parameter

The FloodUnknown parameter provides the ability to toggle the flooding of unknown unicast destination frames. When this parameter is set to true, the MXK always forwards frames with an unknown unicast MAC if the bridge is set for forward to unicast. When this parameter is set to false, the MXK always discards frames with an unknown unicast MAC if the bridge is set for forward to unicast. Any frame that does not find a match in the forwarding table will be discarded.

For transparent bridges, the default setting for this parameter is true. For uplink bridges, the default setting for this parameter is false.

FloodMulticast parameter

The FloodMulticast parameter allows the MXK to flood all multicast traffic received on a bridge out to all other ports in the VLAN. This is useful for architectures where the MXK is acting as an aggregation point with no user interfaces. By default, this parameter is set to false for all bridge types.

When set to true, this parameter causes all multicast frames to be forwarded out all of the bridge interfaces within the VLAN, except the interface where the multicast was received.

To view the setting for this parameter, enter get **bridge-interface-record**:

```
zSH> get bridge-interface-record 1-11-1-932-gponport/bridge
bridge-interface-record  1-11-1-932-gponport/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {100}
stripAndInsert: ---------------------->  {true}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {true}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {5}
learnMulticast: ---------------------->  {true}
forwardToUnicast: -------------------->  {false}
forwardToMulticast: ------------------>  {false}
forwardToDefault: -------------------->  {true}
bridgeIfCustomDHCP: ------------------>  {false}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}
vlanIdCOS: --------------------------->  {0}
outgoingCOSOption: ------------------->  {disable}
outgoingCOSValue: -------------------->  {0}
s-tagTPID: --------------------------->  {0x8100}
s-tagId: ----------------------------->  {0}
s-tagStripAndInsert: ----------------->  {true}
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}
s-tagIdCOS: -------------------------->  {0}
s-tagOutgoingCOSValue: --------------->  {0}
mcastControlList: -------------------->  {}
maxVideoStreams: --------------------->  {0}
isPPPoA: ----------------------------->  {false}
floodUnknown: ------------------------>  {false}
floodMulticast: ---------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
```

Advanced bridging configurations

The default settings for bridge interfaces are created based on the usage of the downlink and uplink parameters of the **bridge add** command. It is recommended not to change the default settings unless advanced bridge configuration is required. Examples of advanced bridge configurations include:

- *Tagged bridge—VLANs* on page 49
- *Broadcast suppression* on page 50
- *Secure bridging* on page 51

Refer to the *CLI Reference Guide* for a complete description of the command options and syntax.

Tagged bridge—VLANs

In most configurations, VLAN IDs should be stripped for traffic destined to downlink interfaces and inserted for traffic destined for upstream interfaces. Downlink interfaces typically do not need to know the VLAN ID since they are on a single Ethernet. You can, however, specify that a downlink interface be tagged, or an uplink interface be untagged. You might want to do this if you are subtending MXK devices and aggregating Ethernet traffic.

Configuring stripAndInsert

Configure the **bridge-interface-record** to change the stripping and insert of VLAN tags for a specified interface.

To change the **stripAndInsert** option:

```
zSH> update bridge-interface-record ethernet2-1201/bridge
bridge-interface-record  ethernet2-1201/bridge
Please provide the following: [q]uit.
vpi: ---------------------------------->  {0}:
vci: ---------------------------------->  {0}:
vlanId: ------------------------------->  {1201}:
stripAndInsert: ----------------------->  {false}:true
customARP: ---------------------------->  {true}:
filterBroadcast: ---------------------->  {true}:
learnIp: ------------------------------>  {false}:
learnUnicast: ------------------------->  {false}:
maxUnicast: --------------------------->  {0}:
learnMulticast: ----------------------->  {false}:
forwardToUnicast: --------------------->  {true}:
forwardToMulticast: ------------------->  {true}:
forwardToDefault: --------------------->  {false}:
bridgeIfCustomDHCP: ------------------->  {true}:
bridgeIfIngressPacketRuleGroupIndex: ->  {0}:
vlanIdCOS: ---------------------------->  {0}:
outgoingCOSOption: -------------------->  {disable}:
outgoingCOSValue: --------------------->  {0}:
s-tagTPID: ---------------------------->  {0x8100}:
s-tagId: ------------------------------>  {0}:
s-tagStripAndInsert: ------------------>  {true}:
s-tagOutgoingCOSOption: --------------->  {s-tagdisable}:
s-tagIdCOS: --------------------------->  {0}:
s-tagOutgoingCOSValue: ---------------->  {0}:
mcastControlList: --------------------->  {}:
maxVideoStreams: ---------------------->  {0}:
isPPPoA: ------------------------------>  {false}:
floodUnknown: ------------------------->  {false}:
floodMulticast: ----------------------->  {false}:
bridgeIfEgressPacketRuleGroupIndex: -->  {0}:
bridgeIfTableBasedFilter: ------------->  {NONE(0)}:
bridgeIfDhcpLearn: -------------------->  {NONE(0)}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
```

```
Record updated.
```

Broadcast suppression

Broadcast suppression enables DHCP information to be relayed between DHCP client and host while broadcast filtering is enabled.

CustomDHCP setting

The customDHCP setting enables bridge interfaces to pass DHCP information independent of the filterBroadcast setting. Setting customDHCP to TRUE will cause that bridge interface to pass DHCP OFFER and ACK packets even though the filterBroadcast is set to TRUE.

To enable CustomDHCP:

For an existing bridge, update the bridge-interface-record.

```
zSH> update bridge-interface-record ethernet2-1201/bridge
bridge-interface-record  ethernet2-1201/bridge
Please provide the following: [q]uit.
vpi: --------------------------------->  {0}:
vci: --------------------------------->  {0}:
vlanId: ------------------------------>  {1201}:
stripAndInsert: ---------------------->  {false}:
customARP: --------------------------->  {true}:
filterBroadcast: --------------------->  {true}:
learnIp: ----------------------------->  {false}:
learnUnicast: ------------------------>  {false}:
maxUnicast: -------------------------->  {0}:
learnMulticast: ---------------------->  {false}:
forwardToUnicast: -------------------->  {true}:
forwardToMulticast: ------------------>  {true}:
forwardToDefault: -------------------->  {false}:
bridgeIfCustomDHCP: ------------------>  {false}:true
bridgeIfIngressPacketRuleGroupIndex: ->  {0}:
vlanIdCOS: --------------------------->  {0}:
outgoingCOSOption: ------------------->  {disable}:
outgoingCOSValue: -------------------->  {0}:
s-tagTPID: --------------------------->  {0x8100}:
s-tagId: ----------------------------->  {0}:
s-tagStripAndInsert: ----------------->  {true}:
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}:
s-tagIdCOS: -------------------------->  {0}:
s-tagOutgoingCOSValue: --------------->  {0}:
mcastControlList: -------------------->  {}:
maxVideoStreams: --------------------->  {0}:
isPPPoA: ----------------------------->  {false}:
floodUnknown: ------------------------>  {false}:
floodMulticast: ---------------------->  {false}:
bridgeIfEgressPacketRuleGroupIndex: -->  {0}:
bridgeIfTableBasedFilter: ------------>  {NONE(0)}:
bridgeIfDhcpLearn: ------------------->  {NONE(0)}:
```

```
...................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

Secure bridging

The MXK enables secure DHCP settings on downlink bridges to prevent users with a statically configured IP address from bypassing DHCP security enforcement. This filter blocks users from accessing the network using anything other than valid DHCP offered IP address.

When packets are received or sent out a secure downlink bridge interface, the MXK checks the IP address against the dynamic IP bridge filter. If a match is found (the address was provided by the DHCP server), the packet is allowed to pass through the filter. Otherwise, it is blocked.

The unicast aging setting for allowed packets is determined based on the DHCP lease time.

Configuring a dynamic IP filter on a bridge

A dynamic IP filter can be configured, modified, and deleted using the bridge add, modify, and delete commands.

1 Create a downlink bridge using the **bridge add** command with the **secure** option to create the dynamic IP filter. The **secure** option creates two static bridge paths (MAC and IP) for each host on the bridge that successfully negotiates its IP address from the DHCP server.

```
zSH> bridge add 1-12-1-501/gponport downlink vlan 240 secure
Adding bridge on 1-12-1-501/gponport
Created bridge-interface-record 1-12-1-501-gponport/bridge

zSH> bridge show
Typ        VLAN    Bridge                         St  Table Data
--------------------------------------------------------------------------------
dwn Tagged 240   1-12-1-501-gponport-240/bridge UP  S 00:01:47:04:0f:c0 (Secure,
TimeLeft: 3574 secs)
```

2 Use the **bridge modify** command to add a dynamic IP filter to an existing bridge or remove a dynamic IP filter from a secure bridge.

```
zSH> bridge modify 1-9-1-922-gponport-998/bridge secure

zSH> bridge modify 1-9-1-922-gponport-998/bridge non-secure
```

Advanced bridge interface parameter usage

The default settings for bridge interfaces are created based on the usage of the downlink and uplink parameters of the **bridge add** command. It is recommended not to change the default settings unless advanced bridge configuration is required. Examples of advanced bridge configurations include:

- *Packet-rule records (Option 82)* on page 52
- *VLAN ID stripandInsert parameter* on page 54
- *Broadcast suppression* on page 55

Refer to the *Zhone CLI Reference Guide* for a complete description of the command options and syntax.

Packet-rule records (Option 82)

The MXK supports packet-rule records so an open-ended number of filter settings can be configured for on a uplink or downlink bridge interface. The same filter settings can also be easily applied to multiple bridge interfaces.

Packet-rule-records are typically assigned to bridge configuration groups on downlink bridge interfaces. Each bridge configuration record contains settings for type and value. The **packetRuleValue** parameter specifies the type of filter to be applied to the interface. The following interfaces can be applied to MXK bridge interfaces:

- bridgeinsertoption82

  bridgeinsertoption82 contains an identification text used with Insert option 82 to identify the DHCP host. When this option is specified, option82 information is displayed in standard text format.

- bridgedhcprelay

  bridgedhcprelay contains the DHCP subnet group ID. If only the DHCP relay option is used, option82 information is displayed in hex format as *slot port shelf vlan.*

- bridgeforbidoui

  bridgeforbidoui contains a 3-byte hexadecimal vendor code used with the Forbid OUI to forbid access on the interface.

Enter **packet-rule-record** to view the interface types available. MXK supports *bridgeinsertoption82*.

```
zSH> show packet-rule-record
packetRuleType:--->   bridgeinsertoption82  bridgedhcprelay
bridgeinsertpppoevendortag  bridgeforbidoui  ratelimitdiscard
colorawareratelimitdiscard  dstmacswapstatic  dstmacswapdynamic
packetRuleValue:-->   {260}
packetRuleValue2:->   {260}
packetRuleValue3:->   {260}
packetRuleValue4:->   {260}
packetRuleValue5:->   {260}
```

The bridge-interface-record profile contains the fields to support the packet-rule-record.

```
zSH> show bridge-interface-record
bridgeIfIngressPacketRuleGroupIndex:->   {0 - 2147483647}
bridgeIfEgressPacketRuleGroupIndex:-->   {0 - 2147483647}
```

Configuring bridge interface record

Configure the **bridge-interface-record** to a given bridge configuration group to a specified interface. Bridge configuration groups are assigned to the interface records by setting the bridgeIfIngressPacketRuleGroupIndex parameter.

To configure a bridge configuration group:

```
zSH> update bridge-interface-record 1-a-1-0-linkAgg-2820/bridge
bridge-interface-record  1-a-1-0-linkAgg-2820/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {2820}
stripAndInsert: ---------------------->  {false}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {false}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {100}
learnMulticast: ---------------------->  {false}
forwardToUnicast: -------------------->  {true}
forwardToMulticast: ------------------>  {false}
forwardToDefault: -------------------->  {false}
bridgeIfCustomDHCP: ------------------>  {false}
bridgeIfIngressPacketRuleGroupIndex: ->  {0}1
vlanIdCOS: --------------------------->  {0}
outgoingCOSOption: ------------------->  {disable}
outgoingCOSValue: -------------------->  {0}
s-tagTPID: --------------------------->  {0x8100}
s-tagId: ----------------------------->  {0}
s-tagStripAndInsert: ----------------->  {true}
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}
s-tagIdCOS: -------------------------->  {0}
s-tagOutgoingCOSValue: --------------->  {0}
mcastControlList: -------------------->  {0}
maxVideoStreams: --------------------->  {0}
isPPPoA: ----------------------------->  {false}
floodUnknown: ------------------------>  {true}
floodMulticast: ---------------------->  {true}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

Configuring packet rule records

Create bridge configuration records using the **packet-rule-record** profile. Enter the group/instance index numbers to assign group and instance identification.

Configure a new **packet-rule-record** for *group/instance* and specify either bridgeinsertoption82, bridgedhcprelay, or bridgeforbidoui. Also specify the packet rule values (a string of information you want traced). For example:

```
zSH> new packet-rule-record 1/1
packet-rule-record  1/1
Please provide the following: [q]uit.
packetRuleType: --->  {bridgeinsertoption82}:
packetRuleValue: -->  {}:00:02:02
packetRuleValue2: ->  {}:
packetRuleValue3: ->  {}:
packetRuleValue4: ->  {}:
packetRuleValue5: ->  {}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
Record created.
```

VLAN ID stripandInsert parameter

In most configurations, VLAN IDs should be stripped for traffic destined to downlink interfaces and inserted for traffic destined for upstream interfaces. Downlink interfaces typically do not need to know the VLAN ID since they are on a single Ethernet. You can, however, specify that a downlink interface be tagged, or an uplink interface be untagged. You might want to do this if you are subtending MXK devices and aggregating Ethernet traffic.

Configuring stripAndInsert

Configure the **bridge-interface-record** to change the stripping and insert of VLAN tags for a specified interface.on the downlink:

To change the **stripAndInsert** option:

```
zSH> update bridge-interface-record 1-1-2-0-eth/bridge
bridge-interface-record  1-1-2-0-eth/bridge
Please provide the following: [q]uit.
vpi: --------------------------------->  {0}:
vci: --------------------------------->  {0}:
vlanId: ------------------------------>  {4094}
stripAndInsert: ---------------------->  {true}: false
customARP: --------------------------->  {false}:
filterBroadcast: --------------------->  {false}:
learnIp: ----------------------------->  {false}:
learnUnicast: ------------------------>  {true}:
maxUnicast: -------------------------->  {5}:
learnMulticast: ---------------------->  {false}:
```

```
forwardToUnicast: -------------------->  {true}:
forwardToMulticast: ------------------>  {false}:
forwardToDefault: -------------------->  {false}:
bridgeIfCustomDHCP: ------------------>  {false}:
bridgeIfIngressPacketRuleGroupIndex: ->  {0}:
vlanIdCOS: --------------------------->  {0}:
outgoingCOSOption: ------------------->  {disable}:
outgoingCOSValue: -------------------->  {0}:
s-tagTPID: --------------------------->  {0x8100}:
s-tagId: ----------------------------->  {4094}:
s-tagStripAndInsert: ----------------->  {true}:
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}:
s-tagIdCOS: -------------------------->  {0}:
s-tagOutgoingCOSValue: --------------->  {0}:
mcastControlList: -------------------->  {}:
maxVideoStreams: --------------------->  {0}:
isPPPoA: ----------------------------->  {false}:
floodUnknown: ------------------------>  {false}:
floodMulticast: ---------------------->  {false}:
bridgeIfEgressPacketRuleGroupIndex: -->  {0}:
bridgeIfTableBasedFilter: ------------>  {NONE(0)}:
bridgeIfDhcpLearn: ------------------->  {NONE(0)}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

Broadcast suppression

Broadcast suppression enables DHCP information to be relayed between DHCP client and host while broadcast filtering is enabled.

CustomDHCP setting

The customDHCP setting enables bridge interfaces to pass DHCP information independent of the filterBroadcast setting. Setting customDHCP to TRUE will cause that bridge interface to pass DHCP OFFER and ACK packets even though the filterBroadcast is set to TRUE.

To enable CustomDHCP:

For an existing bridge, update the bridge-interface-record and enter **update bridge-interface-record** *interface/type.*

```
zSH> update bridge-interface-record 1-1-2-0-eth/bridge
bridge-interface-record  1-1-2-0-eth/bridge
Please provide the following: [q]uit.
vpi: --------------------------------->  {0}:
vci: --------------------------------->  {0}:
vlanId: ------------------------------>  {4094}:
stripAndInsert: ---------------------->  {false}:
customARP: --------------------------->  {false}:
filterBroadcast: --------------------->  {false}:
learnIp: ----------------------------->  {false}:
```

```
learnUnicast: ------------------------>  {true}:
maxUnicast: -------------------------->  {5}:
learnMulticast: ---------------------->  {false}:
forwardToUnicast: -------------------->  {true}:
forwardToMulticast: ------------------>  {false}:
forwardToDefault: -------------------->  {false}:
bridgeIfCustomDHCP: ------------------>  {false}: **true**
bridgeIfIngressPacketRuleGroupIndex: ->  {0}:
vlanIdCOS: --------------------------->  {0}:
outgoingCOSOption: ------------------->  {disable}:
outgoingCOSValue: -------------------->  {0}:
s-tagTPID: --------------------------->  {0x8100}:
s-tagId: ----------------------------->  {4094}:
s-tagStripAndInsert: ----------------->  {true}:
s-tagOutgoingCOSOption: -------------->  {s-tagdisable}:
s-tagIdCOS: -------------------------->  {0}:
s-tagOutgoingCOSValue: --------------->  {0}:
mcastControlList: -------------------->  {}:
maxVideoStreams: --------------------->  {0}:
isPPPoA: ----------------------------->  {false}:
floodUnknown: ------------------------>  {false}:
floodMulticast: ---------------------->  {false}:
bridgeIfEgressPacketRuleGroupIndex: -->  {0}:
bridgeIfTableBasedFilter: ------------>  {NONE(0)}:
bridgeIfDhcpLearn: ------------------->  {NONE(0)}:
....................
Save changes? [s]ave, [c]hange or [q]uit: **s**
Record updated.
```

Administrative commands

The MXK provides the following administrative commands:

- **bridge delete**
- **bridge show**
- **bridge showall**
- **bridge-path add**
- **bridge-path show**
- **bridge-path delete**
- **bridge stats**
- **bridge flush**

Refer to the *MALC CLI Reference Guide* for a detailed explanation of the available bridge commands.

Bridge delete command

The **bridge delete** command deletes a specific bridge entry from the system.

Bridge show/showall commands

The **bridge show** and **bridge showall** commands display either a single bridge path entry or the entire bridge table.

```
zSH> bridge showall
Type VLAN       Bridge                         St  Table Data
------------------------------------------------------------------------------------
upl Tagged 998   linkagg-a-1-998/bridge        UP  S VLAN 998 default [U: 3600 sec,
M: 61 sec, I: 30 sec]
tls Tagged 666   1-9-1-710-gponport-666/bridge  UP
tls Tagged 142   linkagg-a-1-142/bridge        UP  D 00:00:0c:07:ac:00
                                                   I=516 A=3189 U=2627 F=0
                                                   D 00:b0:c2:f2:b6:fc
                                                   I=516 A=103330 U=2627 F=0
                                                   D 00:b0:c2:f5:54:ee
                                                   I=516 A=3189 U=2627 F=0
upl Tagged 1101  1-a-1-0-linkAgg-1101/bridge   UP
upl Tagged 500   linkagg-a-1-500/bridge        UP  S VLAN 500 default [U: 3600 sec,
M: 150 sec, I: 0 sec]
dwn Tagged 998   1-9-1-921-gponport-998/bridge  DWN
dwn Tagged 998   1-9-1-922-gponport-998/bridge  UP
tls Tagged 142   1-9-1-531-gponport-142/bridge  DWN
tls Tagged 3843  1-a-1-0-linkAgg-3843/bridge   UP
upl          134 ethernet4/bridge              INI NOT LOCAL
tls Tagged 2820  linkagg-a-1-2820/bridge       UP  D 00:1c:15:00:7b:2a
                                                   I=553 A=103425 U=2669 F=0
tls Tagged 2920  linkagg-a-1-2920/bridge       UP  D 00:1c:15:00:7b:42
                                                   I=554 A=102870 U=2668 F=0
tls Tagged 2820  1-9-1-910-gponport-2820/bridge UP  D 00:1c:15:00:7b:42
                                                   I=555 A=102754 U=2667 F=0
tls Tagged 2920  1-9-1-911-gponport-2920/bridge UP  D 00:1c:15:00:7b:2a
                                                   I=556 A=103425 U=2668 F=0
tls Tagged 2620  linkagg-a-1-2620/bridge       UP  D 00:00:02:00:00:01
                                                   I=562 A=4009 U=2653 F=0
tls Tagged 2720  linkagg-a-1-2720/bridge       UP  D 00:00:01:00:00:01
                                                   I=566 A=3189 U=2648 F=0
tls Tagged 2720  1-9-1-511-gponport-2720/bridge UP
tls Tagged 2620  1-9-1-510-gponport-2620/bridge UP
Aging counter:   48400
Age out counter: 0
Renew failed:    0
Filter renewed:  0
Flap Suppresses: 0
Flap Enabled:    False

zSH> bridge show
Type VLAN       Bridge                         St  Table Data
```

```
------------------------------------------------------------------------------------
upl Tagged 998   linkagg-a-1-998/bridge         UP  S VLAN 998 default [U: 3600 sec,
M: 61 sec, I: 30 sec]
tls Tagged 666   1-9-1-710-gponport-666/bridge  UP
tls Tagged 142   linkagg-a-1-142/bridge         UP  D 00:00:0c:07:ac:00
                                                    D 00:b0:c2:f2:b6:fc
                                                    D 00:b0:c2:f5:54:ee
upl Tagged 1101  1-a-1-0-linkAgg-1101/bridge    UP
upl Tagged 500   linkagg-a-1-500/bridge         UP  S VLAN 500 default [U: 3600 sec,
M: 150 sec, I: 0 sec]
dwn Tagged 998   1-9-1-921-gponport-998/bridge  DWN
dwn Tagged 998   1-9-1-922-gponport-998/bridge  UP
tls Tagged 142   1-9-1-531-gponport-142/bridge  DWN
tls Tagged 3843  1-a-1-0-linkAgg-3843/bridge    UP
upl          134 ethernet4/bridge               INI NOT LOCAL
tls Tagged 2820  linkagg-a-1-2820/bridge        UP  D 00:1c:15:00:7b:2a
tls Tagged 2920  linkagg-a-1-2920/bridge        UP  D 00:1c:15:00:7b:42
tls Tagged 2820  1-9-1-910-gponport-2820/bridge UP  D 00:1c:15:00:7b:42
tls Tagged 2920  1-9-1-911-gponport-2920/bridge UP  D 00:1c:15:00:7b:2a
tls Tagged 2620  linkagg-a-1-2620/bridge        UP  D 00:00:02:00:00:01
tls Tagged 2720  linkagg-a-1-2720/bridge        UP  D 00:00:01:00:00:01
tls Tagged 2720  1-9-1-511-gponport-2720/bridge UP
tls Tagged 2620  1-9-1-510-gponport-2620/bridge UP
```

Bridge stats

The **bridge stats** command displays and clear bridge interface statistics for all bridges, bridges associated with a specified VLAN ID, and a specified bridge interface.

```
zSH> bridge stats
Interface                    Received Packets      Transmitted Packets
Name                         UCast  MCast  BCast   UCast  MCast  Bcast  Error
linkagg-a-1-998              0      8405   0       0      0      0      0
1-9-1-710-gponport-666       0      0      0       0      0      0      0
linkagg-a-1-142              1207   164    28      0      0      0      0
1-a-1-0-linkAgg-1101         0      0      0       0      0      0      0
linkagg-a-1-500              566    0      112k    0      0      0      0
1-9-1-921-gponport-998       0      0      0       0      0      0      0
1-9-1-922-gponport-998       0      0      0       0      3209   0      0
1-9-1-531-gponport-142       0      0      0       0      0      0      0
1-a-1-0-linkAgg-3843         0      0      0       0      0      0      0
ethernet4                    0      0      0       0      0      0      0
linkagg-a-1-2820             0      0      3494    0      0      694    0
linkagg-a-1-2920             0      0      1       0      0      3494   0
1-9-1-910-gponport-2820      0      0      694     0      0      3460   0
1-9-1-911-gponport-2920      0      0      3494    0      0      1      0
linkagg-a-1-2620             0      0      0       0      0      0      0
linkagg-a-1-2720             0      0      0       0      0      0      0
1-9-1-511-gponport-2720      0      0      0       0      0      0      0
1-9-1-510-gponport-2620      0      0      0       0      0      0      0

zSH> bridge stats vlan 998
```

```
Interface                    Received Packets      Transmitted Packets
Name                         UCast  MCast  BCast   UCast  MCast  Bcast  Error
linkagg-a-1-998              0      8429   0       0      0      0      0
1-9-1-921-gponport-998       0      0      0       0      0      0      0
1-9-1-922-gponport-998       0      0      0       0      3218   0      0
```

3 GPON CONFIGURATION

This chapter explains how to configure voice, video, and data connections between the MXK and the Zhone zNID. It includes the following sections:

GPON configuration

The MXK supports configuring GPON voice, data, and video connections between the MXK and 3rd party NID CPEs. The MXK supports three GEM ports per ONU and vendor-specific OMCI file configuration.

Note: Contact Zhone support for more details about 3rd party NID support and OMCI file configuration details.

Multiple GEM Ports

The GPON OLT line card can support one multicast GPON Ecapsulation Method (GEM) port, or up to three unicast GEM ports per ONT.

The three unicast GEM port IDs have a fixed numbering system:

- 501–532
- 701–732
- 901 – 932

Each ONT is automatically assigned a set of three GEM port IDs. For example, for ONT ID 1, the three GEM port IDs are 501, 701, and 901.

The following example shows the GEM port IDs assignment for ONUs 1 and 2 on OLT 1:

```
zSH> gpononu gemports
Slot 18  olt 1
Onu  Name                     GemPorts        Admin   BW(Mbits/sec)
  1  1-18-1-1             1-18-1-501/gponport  UP      5
                          1-18-1-701/gponport  UP      5
                          1-18-1-901/gponport  UP     5
  2  1-18-1-2             1-18-1-502/gponport  UP     5
                          1-18-1-702/gponport  UP     5
                          1-18-1-902/gponport  UP     5
...........
```

Note: Note that the zNID-GPON-4200 ONT only requires one GEM port (5XX) because it can perform traffic shaping on a per VLAN basis inside of that single GEM port.

Activate ONTs

Activating ONTs

There are two sets of commands to activate the ONT. Use gpononu show to view free ONTs, then use **gpononu set** command to activate the ONT.

The **gpononu set** command syntax is:

gpononu set *slot/olt/onu* | *interfaceName* [*sernoID*] [**bw** *[gemport ID/*] *value*][**omci** *omci filename* | **noomci**]

1 To view all free ONTs and any serial numbers under the OLT 3/1 enter:

```
zSH> gpononu show 3/1
Free ONUs for slot 3 olt 1:
    1   17   18   19   20   21   22   23   24   25   26   27 28 29 30 31 32
Discovered serial numbers for slot 3 olt 1:
sernoID   Vendor  Serial Number     sernoID   Vendor  Serial Number
    9      ZNTS    266175
```

In the above example, the 9 is the sernoID (serial number) that corresponds to the serial number 226175. Use the sernoID number in the **gpononu set** command.

2 To attach the ONT 3/1/1 with sernoID 9 (the sernoID 9 is associated with serial number 266175) to activate ONT 3/1/1 enter:

```
zSH> gpononu set 3/1/1 9
Onu 1 successfully enabled with serial number ZNTS 266175
```

3 Verify ONT 3/1/1 is activated with serial number 266175.

```
zSH> gpononu show 3/1/1
Slot 3  olt 1
Onu   Name           Enabled  Serial Number     OMCI filename
 1  1-3-1-1            Yes     ZNTS 266175      (none)
```

4 To disable an ONT, but keep the serial number for this ONT, use the **port down** command:

```
zSH> port down 1-3-1-1/gpononu
1-3-1-1/gpononu set to admin state DOWN
```

5 To disable an ONT, and clear the serial number for this ONT, use the **gpononu clear** command:

```
zSH> gpononu clear 3/1/1
Onu1 (previously with serial number ZNTS 266175) has been cleared
```

GPON alloc-ID profile

The GPON alloc-ID profile is associated with each GEM port. In the GPON alloc-ID profile, you can specify bandwidth, traffic class, and compensation mode for the upstream traffic of the GEM port.

Use **get gpon-alloc-id** to show the GEM port settings in the alloc-ID profile with the syntax **get gpon-alloc-id** *shelf-slot-olt-0-gpon/linegroup/gemportID*:

```
zSH> get gpon-alloc-id 1-4-2-0-gponolt/linegroup/501
gpon-alloc-id  1-4-2-0-gponolt/linegroup/501
onu-id: -------->  {1}
guaranteed-bw: ->  {10240}
traffic-class: ->  {ubr}
compensated: --->  {false}
```

- guaranteed-bw: The bandwidth value is multiple of 512 kbps. By default, the value is 512 kbps.
- traffic-class: The traffic class value is UBR or CBR.
- compensated: For CBR, the compensation mode can be “true” or “false”. Sometimes access can be skipped because of the ranging window. In this case CBR access can be compensated immediately after the ranging window to prevent possible jitter of the CBR channel.

Modifying upstream bandwidths for GEM ports

Use the **gpononu set** command to change the upstream bandwidths for multiple GEM ports. Modify the guaranteed-bw field in the gpon-alloc-id profile when needed. However, you can only modify the traffic-class and compensation mode (compensated) fields directly in the gpon-alloc-id profile.

You may need to change the bandwidths on the different GEM ports for an ONT to provide different services for the end-user such as voice, data, or video.

The following example assigns ONT 3/1/1 with serial number ID (sernoID) 9, changes upstream bandwidths for GEM ports 501 to 12 mbps and 901 to 1 mbps:

```
zSH> gpononu set 3/1/1 9 bw 501/12 bw 901/1
Onu 1 successfully enabled with serial number ZNTS 266175
```

The following example changes upstream bandwidth for ONT 3/1/1 GEM port 501 without assigning the serial number. Specifying **bw** *value* instead of

bw *gemport ID / value* indicates the bandwidth value is assigned to the default GEM port 5xx (i.e. GEM port 501 in this example):

```
zSH> gpononu set 3/1/1 bw 12
Bandwidth for 501 has been changed from 18Mbps to 12 Mbps
```

GPON OMCI configuration

The MXK supports configuring GPON data, voice, and video connections between the MXK-GPONX2-IO card and third party ONTs. For these third party ONTs, the MXK supports three GEM ports per ONT and vendor-specific OMCI file configuration.

OMCI file

The OMCI file (a standards-based ONT Management and Control Interface file) is supplied by Zhone Technologies for use with the OMCI-enabled ONTs.

The OMCI file contains the commands used to configure the ONTs that are related to customer premises equipment (CPE) devices.

The OMCI file must be downloaded from the server, placed in the OMCI directory, and that filename must be entered in the *gpon-olt-onu-config* file.

Downloading an OMCI file

First create the OMCI directory, and then download an OMCI file into this directory.

1 Create a directory at the root level for OMCI. The directory must be named *omci.*

```
zSH> mkdir /omci
```

2 Download the OMCI file to the *omci* directory. In this case the OMCI file is *cigprov_eth3_fxs.txt.*

```
zSH> filedownload 172.16.80.201 pathname/cigprov_eth3_fxs.txt /omci/
cigprov_eth3_fxs.txt
```

Associating the OMCI file with the ONT

The following example shows how to associate the OMCI file *cigprov_eth3_fxs.txt* with ONT 3/1/1:

1 Activate the ONT 3/1/1. Refer to *Activating ONTs* on page 62 for the procedure.

2 Associate the ONT 3/1/1 with the OMCI file *cigprov_eth3_fxs.txt.*

```
zSH> gpononu set 3/1/1 omci cigprov_eth3_fxs.txt
```

3 Verify that the ONT 3/1/1 is associated with the OMCI file *cigprov_eth3_fxs.txt*.

```
zSH> gpononu show 3/1/1
Slot 3  olt 1
Onu  Name            Enabled  Serial Number     OMCI filename
  1  1-3-1-1             Yes  ZNTS 266175      cigprov_eth3_fxs.txt
```

4 Verify that the *gpon-olt-onu-config* file contains correct values for *onu-added* and the *OMCI-file-name* parameter.

```
zSH> get gpon-olt-onu-config 1-3-1-1/gpononu
gpon-olt-onu-config  1-3-1-1/gpononu
serial-no-vendor-id: ------->  {ZNTS}
serial-no-vendor-specific: ->  {266175}
password: ------------------>  {}
auto-learn: ---------------->  {enabled}
power-level: --------------->  {0}
us-ber-interval: ----------->  {5000}
ds-ber-interval: ----------->  {5000}
onu-added: ----------------->  {true}
omci-file-name: ------------>  {cigprov_eth3_fxs.txt}
zSH>
```

Service configuration

Example GPON configurations specifying the GEM ports 501, 701, and 901 for ONT 3/1/1. GEM port 501 is configured for data service, GEM port 701 is configured for voice service, and GEM port 901 is configured for video service.

```
zSH> bridge add 1-3-1-501/gponport downlink vlan 100 tagged  for line side
Adding bridge on 1-3-1-501/gponport
Created bridge-interface-record 1-3-1-501-gponport-200/bridge

zSH> bridge add 1-1-1-0/eth uplink vlan 100        for uplink side
```

Refer to *Bridging behavior for tagged, s-tagged, and untagged* on page 31 for the overall VLAN bridging configuration.

```
zSH> host add 1-3-1-901/gponport vlan 300 dynamic 43 5 video 1/5  Video service
Adding host for 1-3-1-901/gponport
```

Refer to *Video bridging* on page 43 for the overall video configuration.

New Commands

gpononu

For GPON configurations, sets and displays ONU and serial number associations. The default bandwidth is 512 kilobytes/sec.

Syntax The following command can associate a serial number with an ONU and enable the ONU; can sets upstream bandwidth for GPON Ecapsulation Method (GEM) port (s); also can associate a OMCI file with an ONU.

`gpononu set` ***`slot/olt/onu`*** **`|`** ***`interfaceName`*** **`[`*`sernoID`*`] [bw`** ***`[gemport ID/`*`]`*`value`*`][omci`*** ***`omci filename`*** **`| noomci]`**

sernoID
Serial number ID is an ID number displayed in **gpononu show** command output. If sernoID is omitted in the **gpononu set** command, this command modifies bandwidth and / or omci filename only.

`[bw` ***`[gemport ID/`*`]`*** ***value*****`]`**
It sets upstream bandwidth. If no GEM port ID is given, then the default GEM port (5xx) is used. In that case, use **bw** *value* to set bandwidth for the GEM port (5xx) gemport.

The upstream bandwidth must be multiple of 512 Kbps. e.g. 12 for 12 Megabits/sec., 1.5 for 1536 Kilobytes/sec.

`omci` ***omci filename***
It indicates omci provisioning, using named file.

`noomci`
It indicates omci is not used (default).

Syntax The following command displays the names and bandwidth of GEM ports for selected ONUs.

`gpononu gemports [`*`slot`*`[/`*`olt`*`[/onu]] |` ***`interfaceName`*`]`***

Syntax The following command displays the operating status and gpon onu line status for selected ONUs.

`gpononu status [`*`slot`*`[/`*`olt`*`[/onu]] |` ***`interfaceName`*`]`***

Syntax The following command displays the available ONUs and the discovered serial numbers for OLTs.

`gpononu show [`*`slot`*`[/`*`olt`*`]]`

Syntax The following command displays ONUs with serial number data and other information.

`gpononu showall [`*`slot`*`[/`*`olt`*`[/onu]] |` ***`interfaceName`*`]`*** **`[enabled] [free] [all]`**

Syntax The following command disable an ONU and clear the serial number to default.

`gpononu clear` ***`slot/olt/onu`*** **`|`** ***`interfaceName`***

Example

```
zSH> gpononu show 8/1
Free ONUs for slot 8 olt 1:
    1    **2**    4    5    6    7    8    9   10   11   12   13
   14   15   16   17   18   19   20   21   22   23   24   25
   26   27   28   29   30   31   32   33   34   35   36   37
   38   39   40   41   42   43   44   45   46   47   48   49
   50   51   52   53   54   55   56   57   58   59   60   61
   62   63   64
Discovered serial numbers for slot 8 olt 1:
sernoID   Vendor  Serial Number     sernoID   Vendor  Serial Number
    1      ZNTS    220001
    2      ZNTS    220002
```

Assign the available ONU 2 with serial number ID (sernoID) 2. By not specifying a name, the default name is used. The default alloc-id bandwidth is 512Kbps. The ONU number can also be used to specify a ONU. For example, ONU2 instead of 8/1/2. And also by not specifying gemport ID, the gemport 5xx is used.

```
zSH> gpononu set 8/1/2 2 bw 4.5
Onu 2 successfully enabled with serial number ZNTS 220002
Bandwidth has been successfully chaged from 1 to 4.5 Mb/
sec
zSH> MAR 12 11:53:01: alert  : 1/8/1025: alarm_mgr: 01:
8:01:02 Critical ONU Up
Line 1/8/1/2/gpononu CAUSE: active
```

Set bandwidth and OMCI filename to ONU 8/1/2 GEM port 512 without assigning the serial number.

```
zSH> gpononu set 8/1/2 bw 4.5 omci cigprov_eth3_fxs.txt
```

Show the GPON ONU names, admin status, and bandwidth for all GEM ports associated with the ONU.

```
zSH> gpononu gemports
Slot 3 olt 1
Onu  Name         GemPorts        Admin     BW(Mbits/sec)
  1  1-3-1-1   1-3-1-501/gponport  UP       18
               1-3-1-701/gponport  UP        0.5
               1-3-1-901/gponport  UP        0.5
  2  1-3-1-2   1-3-1-502/gponport  UP       18
               1-3-1-702/gponport  UP        0.5
               1-3-1-902/gponport  UP        0.5
...
```

Check the ONU bandwidth setting.

```
zSH> gpononu bw 8/1/2
Current bandwidth allocation is 4.5 Mbits/sec
```

Show all ONU’s on the MXK.

```
zSH> gpononu showall
Slot 8  olt 1
Onu  Name          Enabled  Serial Number    OMCI filename
```

```
 1  1-8-1-1            No  TXPO 754975236    txpces1
 2  1-8-1-2           Yes  TXPO 754975233    txpces2
 3  1-8-1-3            No  ZNTS 0            (none)
...
```

Show only the ONU 2.

```
zSH> gpononu showall 8/1/2
Slot 8  olt 1
Onu  Name          Enabled  Serial Number    OMCI filename
 2  1-8-1-2           Yes  TXPO 754975233    txpces2
```

Show only the enabled ONU’s.

```
zSH> gpononu showall 8/1 enabled
Slot 8 olt 1
Onu  Name          Enabled  Serial Number    OMCI filename
 2  1-8-1-2           Yes  TXPO 754975233    txpces2
Total ONUs = 1
```

Clear or deactivate ONU 2.

```
zSH> gpononu clear 8/1/2
Onu 2 (previously enabled with serial number ZNTS 220002)
has been cleared
zSH> MAR 12 14:17:13: alert  : 1/8/1025: alarm_mgr: 01:
8:03:01 Critical ONU Down
Line 1/8/1/2/gpononu CAUSE: inactive
```

Show the OMCI file name for the ONU.

```
zSH> get gpon-olt-onu-config 1-3-1-1/gpononu
gpon-olt-onu-config  1-3-1-1/gpononu
serial-no-vendor-id: ------->  {ZNTS}
serial-no-vendor-specific: ->  {266175}
password: ------------------>  {}
auto-learn: ---------------->  {enabled}
power-level: --------------->  {0}
us-ber-interval: ----------->  {5000}
ds-ber-interval: ----------->  {5000}
onu-added: ----------------->  {true}
omci-file-name: ------------>  {cigprov_eth3_fxs.txt}
zSH>
```

Change the settings on ONU.

```
zSH> update gpon-olt-onu-config 1-9-1-6/gpononu
gpon-olt-onu-config  1-9-1-6/gpononu
Please provide the following: [q]uit.
serial-no-vendor-id: ------->  {TXPO}:
serial-no-vendor-specific: ->  {754975317}:
password: ------------------>  {}:
auto-learn: ---------------->  {enabled}:
power-level: --------------->  {0}:
```

```
us-ber-interval: ----------->  {5000}:
ds-ber-interval: ----------->  {5000}:
onu-added: ----------------->  {false}:
omci-file-name: ------------>  {cigprov_eth3_fxs.txt}:
....................
Save changes? [s]ave, [c]hange or [q]uit:
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

Show the GPON alloc-ID profile for each GEM port.

```
zSH> get gpon-alloc-id 1-4-2-0-gponolt/linegroup/501
gpon-alloc-id  1-4-2-0-gponolt/linegroup/501
onu-id: -------->  {1}
guaranteed-bw: ->  {10240}
traffic-class: ->  {ubr}
compensated: --->  {false}
```

4 CONFIGURING LINK AGGREGATION

This chapter explains how to configure link aggregation on the MXK-UPLINK-2X10G-8X1GE and the MXK-UPLINK-8X1GE uplink cards.

Link aggregation and LACP

The MXK supports 802.3ad link aggregation on the uplink cards. Link aggregation allows aggregating physical Gigabit Ethernet or Fast (10/100 mbps) Ethernet uplink ports into one single aggregated logical port for additional bandwidth capacity and resiliency. A link aggregation group can consist of up to eight ports.

The MXK-UPLINK-2X10GE-8X1G and MXK-UPLINK-8X1G uplink cards also support the Link Aggregation Control Protocol (LACP), a layer 2 protocol used between network elements to exchange information regarding a link's ability to be aggregated with other similar links.

For redundant GigE uplink card configurations, the link aggregated ports on each card provide redundant uplink port protection.

Note: You will need to configure the Ethernet switch on the remote end for link aggregation.

LACP control is set in the aggregationMode parameter in the ether profile of the Ethernet port.

LACP parameter:

```
aggregationMode:-->   on  off  passive  active
```

The default for the aggregationMode parameter is on.

The aggregationMode parameter field has four values defined as follows:

- On

  This Ethernet link can be aggregated manually using the **linkagg** command. LACP messages are not sent from this port, and any received LACP messages are ignored.

- Off

This Ethernet link cannot be aggregated either manually or dynamically; LACP is not sent from this port and any received LACP messages are ignored.

- Active

  The setting for LACP use, the Ethernet link sends and receives LACP messages and link aggregates automatically when the remote system responds with the appropriate LACP messages.

- Passive

  This mode sets a link to receive LACP messages, and responds with LACP when receiving a far-end LACP initiation.

Table 5 shows the compatibility matrix for the four settings.

Table 5: LACP compatibility matrix settings

Device one	Device two	Comments
Active	Active	Both devices are sending and receiving LACP. Recommended setting for dynamic aggregation.
Active	Passive	One side of the connection between devices attempts to negotiate a aggregated group. Functional, but not recommended.
On	On	Will make links available for manual aggregating; only recommended if the far-end device is not capable of LACP.

Changing the default aggregationMode parameter

To set LACP automatic link aggregation, the default *aggregationMode* parameter must be changed from on to active.

1 Enter the update command for the Ethernet port, in this case 1-a-9-0/eth and enter active in the *aggregationMode* parameter line.

```
zSH> update ether 1-a-9-0/eth
ether  1-a-9-0/eth
Please provide the following: [q]uit.
autonegstatus: ---->  {enabled}:
mauType: ---------->  {mau1000baselxfd}:
restart: ---------->  {norestart}:
ifType: ----------->  {mau1000baselxfd}:
autonegcap: ------->  {b10baseTFD+b100baseTXFD+b1000baseTFD}:
remotefault: ------>  {noerror}:
clksrc: ----------->  {automatic}:
pauseFlowControl: ->  {disabled}:
aggregationMode: -->  {on}: active
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

2 Verify the change.

```
zSH> **get ether 1-a-9-0/eth**
ether  1-a-9-0/eth
autonegstatus: ---->  {enabled}
mauType: ---------->  {mau1000baselxfd}
restart: ---------->  {norestart}
ifType: ----------->  {mau1000baselxfd}
autonegcap: ------->  {b10baseTFD+b100baseTXFD+b1000baseTFD}
remotefault: ------>  {noerror}
clksrc: ----------->  {automatic}
pauseFlowControl: ->  {disabled}
aggregationMode: -->  {active}
```

Figure 4 illustrations the FE/GE physical ports available for link aggregation and Figure 5 show two FE/GE physical ports after they are aggregated. Two one gigabit FE ports are aggregated together to create a single two gigabit FE port.

Figure 4: Ethernet uplink ports not link aggregated

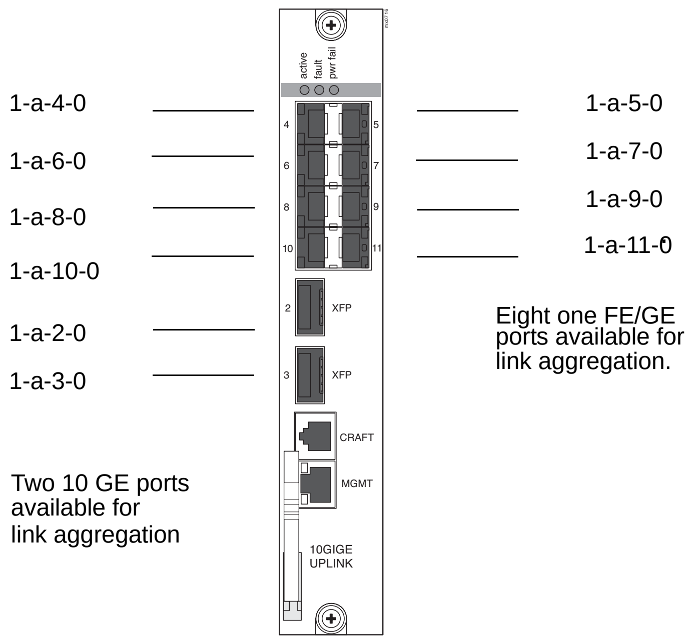
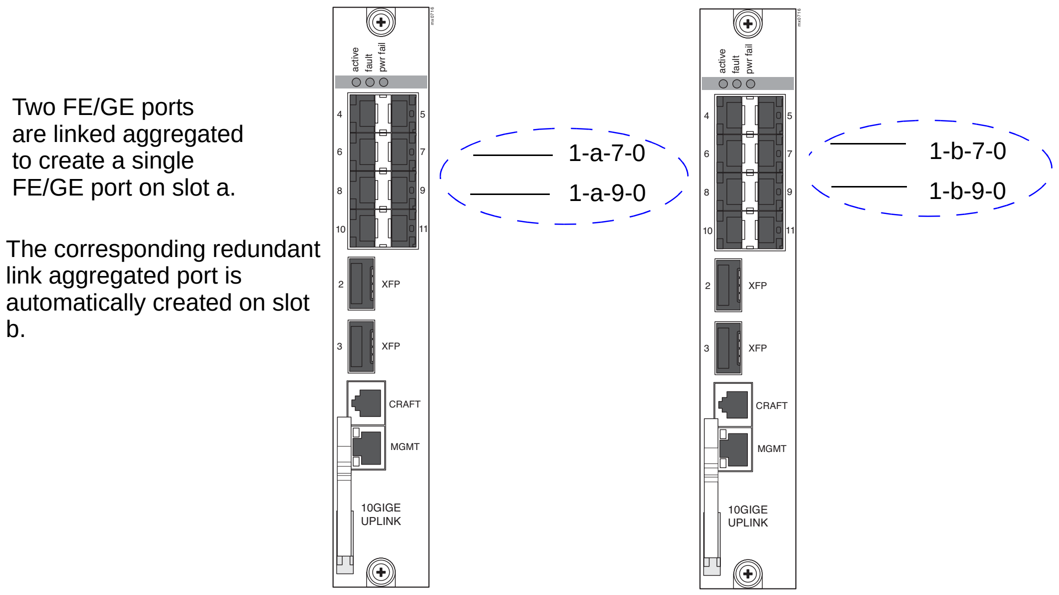


Figure 5: Ethernet redundant uplink ports link aggregated


Two FE/GE ports are linked aggregated to create a single FE/GE port on slot a.
The corresponding redundant link aggregated port is automatically created on slot b.
1-a-7-0
1-a-9-0
1-b-7-0
1-b-9-0
active
fault
pwr fail
XFP
XFP
CRAFT
MGMT
10GIGE UPLINK

Creating link aggregated ports

Link aggregation on the MXK, for either Fast Ethernet or GigE ports, can be performed from the command line interface.The syntax for the link aggregation command is:

```
zSH> linkagg
Usage: linkagg <add|delete|show> group <aggregation name/
type> link <linkname/type>
```

To create a link aggregation between two fast ethernet ports, for example, 1-a-9-0/ and 1-a-10-0/ using the **linkagg add** command enter:

1 Make the physical connections between the two devices, for instance, between the MXK uplink card and a MALC Ethernet card.

2
```
zSH> linkagg add group test/linkagg link 1-a-9-0/eth
Interface test/linkagg does not exist
```

group can be any user-defined string.

Use the same **group** name for both of the ports you are aggregating.

Note: Ignore the message:

```
Interface test/linkagg does not exist
```

```
zSH> linkagg add group test/linkagg link 1-a-10-0/eth
```

The **lingagg add** automatically creates the corresponding redundant link for the uplink card in slot *b*, in this case, *1-b-9-0/* and *1-b-10-0/*.

3 Enter the **linkagg show** command to view the ports just aggregated.

```
zSH> linkagg show
LinkAggregations:
```

```
slot    unit    ifName         partner: Sys           Pri     grp ID  admin   numLinks
------------------------------------------------------------------------------------
a       1       1-a-1-0         00:0c:db:e8:7e:00       0x1     0x1     up      1
        links           slot    port    subport         state   admin
        ---------------------------------------------------------
         1-a-2-0         a       2       0              UP       up
b       1       1-b-1-0                 0x0     0x0     up      1
        links           slot    port    subport         state   admin
        ---------------------------------------------------------
         1-b-2-0         b       2       0              DN       up
```

4 Create an aggregated port on the card the MXK is connected to.

Deleting a link aggregated port

To delete the link aggregated port:

Enter the **linkagg delete** command:

```
zSH> linkagg delete group test/linkagg link 1-a-9-0/
eth
zSH> linkagg delete group test/linkagg link 1-a-10-0/
eth
```

Link resiliency

When an aggregated link fails, the linkagg interface remains up with a single physical port link. If the failed link returns, the link aggregation group adds the link back without any service interruption. If both links in a link aggregation group fail, then the link aggregation group is moved to a down state until at least one of the physical links is restored.

Configuring interfaces for link aggregation

Interfaces can be added to link aggregation ports for bridging and IP routing.

Bridge configurations

Unlearned traffic received on this interface is forwarded to the external network.

Creating bridges with link aggregation

1 To verify link aggregation groups, enter:

```
zSH> linkagg show
LinkAggregations:
slot    unit    ifName         admin   numLinks
------------------------------------------------
 a      1       1-a-1-0        up      1
        link: 1-a-2-0   slot a  port 2  subport 0       admin up
 b      1       1-b-1-0        up      1
```

```
        link: 1-b-2-0   slot b  port 2  subport 0      admin up
```

2 To create an uplink bridge with link aggregation, enter:

```
zSH> bridge add 1-a-1-0/linkagg uplink vlan 333 tagged
Adding bridge on 1-a-1-0/linkagg
Created bridge-interface-record 1-a-1-0-linkAgg-333/bridge
```

3 To delete a bridge with link aggregation enter:

```
zSH> bridge delete 1-a-1-0-linkAgg-333/bridge vlan 333
1-a-1-0-linkAgg-333/bridge Delete complete
```

Interface configurations

To add an interface on the logical link aggregation port:

```
zSH> interface add 1-a-1-0/linkagg 10.10.10.1 255.255.255.0
zSH> interface show
```

This creates an IP interface on the link aggregation port with an IP address of 10.10.10.1, with a subnet mask of 255.255.255.0.

5 DIAGNOSTICS AND ADMINISTRATION

This chapter describes tasks you might need to perform to administer the MXK. It includes the following information:

MXK logging

This section explains how to use logging on the MXK. It includes:

Overview

Logging enables administrators to monitor system events by generating system messages. It sends these message to:

- A management session (either on the serial craft port or over a telnet session)
- A log file on the device
- A syslog server (optional)

The type of information sent in these messages can be configured using the **log** command. By default, the system sends the same type of information to all log message destinations. If you want to send different types of messages to the syslog daemon, use the **syslog** command.

Enable/disable logging

By default, log messages are enabled on the serial craft port. Use the **log session** command and the **log serial** command to enable/disable logging:

The **log session** command enables/disables logging messages for that session only. If the user logs out, the logging setting returns to the default. To enable logging for the current session only:

```
zSH> log session on
```

To disable logging for the session:

```
zSH> log session off
```

The **log serial** command enables/disables logging messages for all sessions on the serial craft port. This setting persists across system reboots. To enable/ disable logging for the serial craft port:

```
zSH> log serial on
```

To disable logging for the serial port:

```
zSH> log serial off
```

Log message format

Log messages contain the following information:

Table 6: Default log message fields

Option	Description
Date	Date stamp of log message. Enabled by default.
Time	Time stamp of log message. Enabled by default.
Ticks	Current tick count. When the **tick** option is used, the date and time fields are not displayed.
Level	Logging level of the message. Enabled by default.
Address	The shelf and slot of the card causing the alarm,
Taskname	Name of task that generated the log message. This is generally useful only for Zhone development engineers. Enabled by default.
Function	Function that generated the log message. This is generally useful only for Zhone development engineers.
Line	Line in code that generated the log message. This is generally useful only for Zhone development engineers.

Table 6: Default log message fields (Continued)

Option	Description
Port	Port related to the log message.
Category	Category of the log message.
System	System related to the log message.
All	Controls all log message options.
Default	Controls the default log message options.
Message text	A description of the error that caused the alarm.

To change the information displayed in the log messages, use the **log option** command. First, display the available options:

```
zSH> log option
Usage: log option  < time     | 1 >  < on | off >
                   < date     | 2 >  < on | off >
                   < level    | 3 >  < on | off >
                   < taskname | 4 >  < on | off >
                   < taskid   | 5 >  < on | off >
                   < file     | 6 >  < on | off >
                   < function | 7 >  < on | off >
                   < line     | 8 >  < on | off >
                   < port     | 9 >  < on | off >
                   < category | 10 > < on | off >
                   < system   | 11 > < on | off >
                   < ticks    | 12 > < on | off >
                   < stack    | 13 > < on | off >
                   < all      | 14 > < on | off >
                   < default  | 15 > < on | off >
option 'ticks' supercedes options 'time' & 'date'
time: date: level: address: log: taskname: function: line: port: category: system:
(0x7cf)
```

Then, turn the option **on** or **off**. For example, the following command will turn the task ID off in log messages:

```
zSH> log option taskid off
time: date: level: address: log: taskname:  (0xf)
```

The following commands will turn on/off the tick counet display in log messages:

```
zSH> log option ticks on
time: date: level: address: log: port: category: system:
ticks:  (0xf07)

zSH> log option ticks off
time: date: level: address: log: port: category: system:
(0x707)
```

The following command will turn all options on in log messages:

```
zSH> log option all on
time: date: level: address: log: taskname: taskid: file:
function: line: port: category: system: ticks:  (0xfff)
```

Modify logging levels

To modify logging, use the **log** command. To modify syslog messages, use the **syslog** command.

To display the current levels for all logging modules, use the **log show** command:

```
zSH> log show
MODULE                                                  LEVEL          STATUS
aal2approv                                              error          enabled
aal2aprec                                               error          enabled
aal2rp                                                  error          enabled
    aal2rpzccapi                                        error          enabled
    aal2rpvcc                                           error          enabled
alarm_mgr                                               error          enabled
assert                                                  error          enabled
atm_cc_mib_hdlr                                         error          enabled
atmmgr                                                  error          enabled
atmmgragnt                                              error          enabled
bds                                                     error          enabled
bds_client                                              error          enabled
callcontrolregistry                                     error          enabled
card                                                    error          enabled
card_resource                                           error          enabled
carddeletehdlr                                          error          enabled
ccrp                                                    error          enabled
cli                                                     error          enabled
...
...
...
```

Logging levels determine the number of messages that are displayed on the console. The higher the log level, the more messages are displayed. The MXK supports the following log levels:

- 1: emergency
- 2: alert
- 3: critical
- 4: error
- 5: warning
- 6: notice
- 7: information

- 8: debug

To change the log level, use the **log** *module* **level** command. For example, the following command changes the card module logging level to emergency:

```
zSH> log level card emergency
Module: card at level: emergency
```

To enable or disable log levels for a module, use the log enable or log disable commands. For example:

```
zSH> log disable card
Module: card is now disabled
```

Using the log cache

The **log cache** command displays the non-persistent log messages. It uses the following syntax:

`log cache`

Displays the log cache.

`log cache max` ***`length`***

Sets the maximum number of log messages to store. The maximum log cache size is 2147483647, depending in the amount of memory available.

`log cache grep` ***`pattern`***

Searches through the log cache for the specified regular expression.

`log cache clear`

Clears the log cache.

`log cache` ***`size`***

Sets the maximum amount of memory for the log cache. Without options, displays the current log size.

`log cache help`

Displays help on the **log cache** command.

Examples

To change the current configured log cache size:

```
zSH> log cache max 200
Maximum number of log messages that can be saved: 200
```

The following example searches through the log cache for the string "Critical":

```
zSH> log cache grep Critical
Searching for: "Critical"
[1]: APR 10 17:07:43: alert  : 1/a/1025: alarm_mgr: 01: a:01 Critical ETHERNET Down -
Ethernet line down
```

```
[2]: APR 10 17:07:43: alert  : 1/a/1025: alarm_mgr: 01: a:01 Critical ETHERNET Up -
Ethernet line up
[3]: APR 10 17:09:57: alert  : 1/b/1025: alarm_mgr: 01: b:01 Critical ETHERNET Down -
Ethernet line down
[4]: APR 10 17:11:10: alert  : 1/a/1025: alarm_mgr: 01: a:00 Critical   MALC NEXT GEN
card running
[5]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:02 Critical ETHERNET Down -
Ethernet uplink down
[6]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:03 Critical ETHERNET Down -
Ethernet line down
[7]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:04 Critical ETHERNET Down -
Ethernet line down
[8]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:05 Critical ETHERNET Down -
Ethernet line down
[9]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:06 Critical ETHERNET Down -
Ethernet line down
[10]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:07 Critical ETHERNET Down -
Ethernet line down
[11]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:08 Critical ETHERNET Down -
Ethernet line down
[12]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:09 Critical ETHERNET Down -
Ethernet line down
[13]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:10 Critical ETHERNET Down -
Ethernet line down
[14]: APR 10 17:11:16: alert  : 1/b/1025: alarm_mgr: 01: b:11 Critical ETHERNET Down -
Ethernet line down
```

View the persistent logs

Use the **log cache** command to view the persistent logs. For example:

```
zSH> log display
MAY 13 14:49:55: emergency: 1/a/12  : shelfctrl: Critical alarm set!
MAY 13 15:21:13: emergency: 1/a/12  : shelfctrl: Critical alarm set!
MAY 13 17:42:46: emergency: 1/a/12  : shelfctrl: _HandleMalcAlarms(): l=2995:
tShelfCtrl: Critical alarm set!
MAY 13 19:05:17: emergency: 1/a/12  : shelfctrl: _HandleMalcAlarms(): l=2995:
tShelfCtrl: Critical alarm set!
MAY 13 20:06:50: emergency: 1/a/12  : shelfctrl: _HandleMalcAlarms(): l=2995:
tShelfCtrl: Critical alarm set!
MAY 13 21:31:38: emergency: 1/a/12  : shelfctrl: _HandleMalcAlarms(): l=2995:
tShelfCtrl: Critical alarm set!
```

Send messages to a syslog server

Modify the following parameters in the **syslog-destination** profile to send messages to a syslog server.

Parameter	Description
address	The IP address of the machine hosting the syslog server. Default: **0.0.0.0**
port	The UDP port to which the syslog messages will be sent. Default: **514**
facility	The syslog facility to which the syslog messages will be sent. Values: **local0** **local1** **local2** **local3** **local4** **local5** **local6** **local7** **no-map** Default: **local0**
severity	The severity level used to filter messages being set to the syslog server. Values: **emergency** **alert** **critical** **error** **warning** **notice** **info** **debug** Default: **debug**

```
zSH> new syslog-destination  1
Please provide the following: [q]uit.
address: -->  {0.0.0.0}: 192.200.42.5 IP address of the syslog server
port: ----->  {514}: leave at default
facility: ->  {local0}:
severity: ->  {debug}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

Specify different log formats for system and syslog messages

The **log-module** profile supports the configuration of persistent log messages, syslog messages, and persistent storage levels by module. You only need to modify this profile if you want to send different messages to admin sessions, the persistent logs, and the syslog server.

Parameter	Description
name	The name of the module whose logging is controlled by this profile. Default: **logtest**
display	Controls the display of messages on the system. Messages logged at this level and above will be displayed. Values: **emergency** **alert** **critical** **error** **warning** **notice** **info** **debug** Default: **error**

Parameter	Description
syslog	Controls the format of messages sent to the syslog server described in the **syslog-destination** profile. This field is similar to the display field, except for the **trackdisplay** value. Values: **emergency** **alert** **critical** **error** **warning** **notice** **info** **debug** **trackdisplay** Messages logged at, and above, the level set in the **display** parameter will also be recorded in the syslog server. Default: **trackdisplay**
store	Controls the persistent storage of messages. This field is similar to the display field, except for the **trackdisplay** value. Values: **emergency** **alert** **critical** **error** **warning** **notice** **info** **debug** **trackdisplay** Messages logged at, and above, the level set in the **display** parameter will also be recorded in the syslog server. Default: **trackdisplay**

```
zSH> new log-module 1
Please provide the following: [q]uit.
name: ---->  {logtest}: test1
display: ->  {error}: warning
syslog: -->  {trackdisplay}:
store: --->  {trackdisplay}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

Example log messages

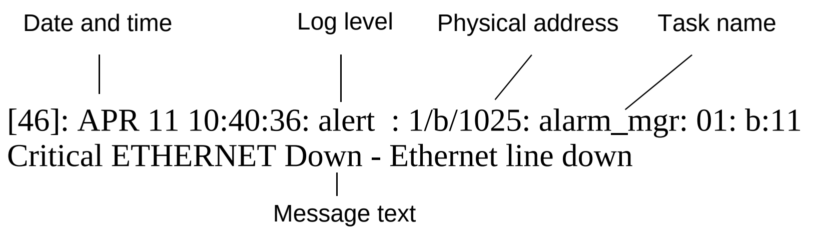
This section provides examples of how to interpret log messages.

Card line down message

The following message appears when a card in the MXK chassis comes up or goes down.

.

The most important parts of the message are the date and time the event occurred, the shelf/slot of the event, and the message text. The remainder of the information is only useful for Zhone development engineers.

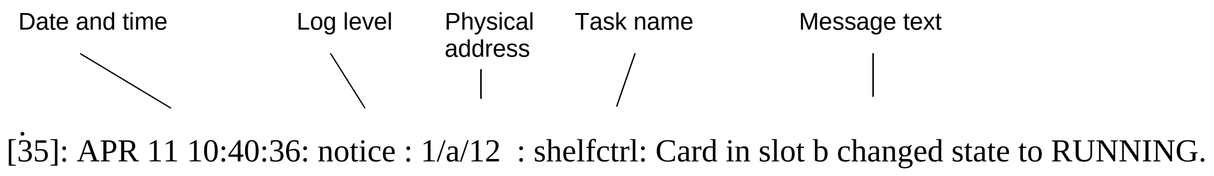


```
[46]: APR 11 10:40:36: alert  : 1/b/1025: alarm_mgr: 01: b:11
Critical ETHERNET Down - Ethernet line down
```

Slot card up message

The next message appears after a slot card has finished loading its software and is ready to be provisioned.

The most important parts of the message are the date and time the event occurred, the physical address (shelf/slot) of the event, and the message text.



```
[35]: APR 11 10:40:36: notice : 1/a/12  : shelfctrl: Card in slot b changed state to RUNNING.
```

Log filter command

The **log filter** command is available as part of the log command functionality. This command enables users to show, set and delete log filters. Log filters limit the scope of log messages to a specific entity for troubleshooting and diagnostics. When a log filter is set, the filter is assigned an index number and only messages relate the specified entity are displayed. Filters can be set for an specific ifindex, slot/port, VCL, or subscriber.

log filter

Restrict the display of log messages to only the log messages for a specified entity.

Syntax **log filter show | set (ifindex|port *slotport*|vcl *ifindex* *vpi* *vci*|subscriber *endpoint*)| delete**

```
zSH> log filter set ifindex 12
New filter saved.

zSH> log filter set port 5 24
New filter saved.

zSH> log filter set vcl 100 0 1
New filter saved.

zSH> log filter set subscriber 22
New filter saved.

zSH> log filter show
Index   Type            Filter Parameters
------  ------------    -----------------------------
 1     Port            slot=1, port=1
 2     Port            slot=1, port=4
 3     IfIndex         IfIndex=12
 4     Port            slot=5, port=24
 5     ATM VCL  IfIndex=100, vpi=0, vci=1
 6     IfIndex         IfIndex=100
 7     IfIndex         IfIndex=104
 8     IfIndex         IfIndex=109
 9     IfIndex         IfIndex=103
10     IfIndex         IfIndex=107

zSH> log filter delete 10
Log filter 10 deleted
```

SNMP

This section describes the following:

- Create SNMP community names and access lists, page 87
- Configure traps, page 89

Create SNMP community names and access lists

Note: By default, the MXK has a single SNMP community defined with the name **ZhonePrivate**. This community has admin access to the system. Zhone recommends that you configure community names and access lists to prevent unauthorized access to the system.

The **community-profile** specifies the community name and an access level for SNMP manager to access the system. It can also optionally specify a **community-access-profile** which is used to verify the source IP address of the SNMP manager. The system supports up to 50 different access lists.

The following community access levels are supported:

- **noaccess**—the community has no access.
- **read**—the community has read-only access to the system, with the exception of information in the **community-profile** and **community-access-profile**.
- **readandwrite**—the community has read/write access to the system, with the exception of information in the **community-profile** and **community-access-profile**.
- **admin**—the community has read and write access to the entire system, including information in the **community-profile** and **community-access-profile**. Note that the ZMS requires admin access to manage the system.

Create a community profile

Note: Configuring a community profile disables the **ZhonePrivate** default community name. If you do change the community name, you must change the name in ZMS or the device will become unmangeable.

The following example defines a community name **public** with read-only privileges:

```
zSH> new community-profile 1
Please provide the following: [q]uit.
community-name: ----->  {}: public
permissions: -------->  {read}:
access-table-index: ->  {0}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

Create community access lists

The following example defines a community name **private** with read/write privileges and also creates an access list to verify that the SNMP managers attempting to access the MXK are coming from known IP addresses 192.168.9.10 and 192.168.11.12:

First, create an access list for the first IP address:

```
zSH> new community-access-profile 2
Please provide the following: [q]uit.
access-table-index: ->  {0}: 1
ip-address: --------->  {0.0.0.0}: 192.168.9.10
```

```
....................
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

Then, create an access list for the second IP address with the same **access-table-index** (1):

```
zSH> new community-access-profile 3
Please provide the following: [q]uit.
access-table-index: ->  {0}: 1
ip-address: --------->  {0.0.0.0}: 192.168.11.12
....................
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

Finally, create a **community-profile** that specifies the community name, and uses the same **access-table-index** (1) as defined in the two **community-access-profiles** you just created:

```
zSH> new community-profile 4
Please provide the following: [q]uit.
community-name: ----->  {}: private ZMS must include this name
permissions: -------->  {read}: readandwrite
access-table-index: ->  {0}: 1
....................
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

Configure traps

The **trap-destination** profile defines a trap recipient the MXK will send traps to. To configure a trap destination you need to know:

- the IP address of the SNMP manager workstation
- the community name the trap recipient expects

Note that the **resendseqno** and **ackedseqno** parameters are set by the ZMS. The other parameters in the **trap-destination** profile can be left at their default values. The following example configures a trap recipient with the IP address 192.168.3.21:

```
zSH> new trap-destination 32
Please provide the following: [q]uit.
trapdestination: ->  {0.0.0.0}: 192.168.3.21
communityname: --->  {}: public
resendseqno: ----->  {0}:
ackedseqno: ------>  {0}:
traplevel: ------->  {low}:
traptype: -------->  {(null)}: 0
trapadminstatus: ->  {enabled}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

MXK alarms

This section describes the following:

- *Alarm manager* on page 90
- *Alarm suppression* on page 91

Alarm manager

The MXK central alarm manager includes the ability to view the active alarms on the system (using the **alarm** command) and the ability to store active alarms on the device. ZMS can use the alarms stored on the device to recreate the state of the alarms if it becomes disconnected.

The alarm command uses the following syntax:

alarm show [**summary**]

For example, the following command displays the number of current active alarms, the total number of alarms, the number of cleared alarms, as well as each active alarm and its severity:

```
zSH> alarm show
************    Central Alarm Manager    ************
        ActiveAlarmCurrentCount         :11
        AlarmTotalCount                 :36
        ClearAlarmTotalCount            :25
        OverflowAlarmTableCount         :0
ResourceId               AlarmType                                 AlarmSeverity
----------               ---------                                 -------------
1-a-2-0/eth              linkDown                                     critical
1-a-3-0/eth              linkDown                                     critical
1-a-6-0/eth              linkDown                                     critical
1-a-7-0/eth              linkDown                                     critical
1-a-8-0/eth              linkDown                                     critical
1-a-9-0/eth              linkDown                                     critical
1-a-10-0/eth             linkDown                                     critical
1-a-11-0/eth             linkDown                                     critical
1-2-2-1/other            linkDown                                     minor
system                   power_supply_b_failure                       warning
system                   not_in_redundant_mode                        major
```

The **summary** option displays the number of current active alarms, the total number of alarms, the number of cleared alarms:

```
zSH> alarm show summary

************    Central Alarm Manager    ************
        ActiveAlarmCurrentCount         :3
        ActiveAlarmTotalCount           :3
        ClearAlarmTotalCount            :0
        OverflowAlarmTableCount         :0
```

Alarm suppression

The alarm suppression feature allows alarm/LED notification and output to be disabled based on alarm severity level for existing and future alarms. When an alarm level is disabled, all existing alarms of that type are cleared from the system. Future alarms of that type do not set LEDs or alarm relays and are not displayed in alarm output.

Alarm suppression is also supported in ZMS.

Table 7 lists the alarm suppression options and the resulting behaviors. By default, alarms for all severity levels are enabled.

Table 7: Alarm suppression options

Alarm Levels Enabled Setting	Alarm Behavior
critical+major+minor+warning	Enables all alarm levels. The default setting.
critical+major+minor	Disables all warning alarms.
critical+major	Disables all minor, and warning alarms.
critical+major+warning	Disables all minor alarms.
critical+minor+warning	Disables all major alarms.
critical+minor	Disables all major and warning alarms.
critical+warning	Disables all major and warning alarms.
critical	Disables all major, minor, and warning alarms.
major	Disables all critical, minor, and warning alarms.
major+minor+warning	Disables all critical alarms.
major+minor	Disables all critical and warning alarms.
major+warning	Disables all critical and minor alarms.
minor	Disables all critical, major, and warning alarms.
minor+warning	Disables all critical and major alarms.
(no levels)	Disables all alarm levels.

This example disables alarm/LED notification and output for all current and future alarms with the severity levels minor and warning.

```
zSH> update system 0
Please provide the following: [q]uit.
syscontact: ----------->  {Zhone Global Services and Support 7001 Oakport
Street Oakland Ca. (877) Zhone20 (946-6320) Fax (510)777-7113
support@zhone.com}:
sysname: -------------->  {Zhone-MXK}:
syslocation: ---------->  {Oakland}:
enableauthtraps: ------>  {disabled}:
```

```
setserialno: ---------->  {0}:
zmsexists: ------------>  {true}:
zmsconnectionstatus: -->  {inactive}:
zmsipaddress: --------->  {172.16.80.160}:
configsyncexists: ----->  {false}:
configsyncoverflow: --->  {true}:
configsyncpriority: --->  {high}:
configsyncaction: ----->  {noaction}:
configsyncfilename: --->  {172.16.80.160_4_1149144921639}:
configsyncstatus: ----->  {synccomplete}:
configsyncuser: ------->  {zmsftp}:
configsyncpasswd: ----->  {** private **}:  ** read-only **
numshelves: ----------->  {1}:
shelvesarray: --------->  {}:
numcards: ------------->  {3}:
ipaddress: ------------>  {172.16.80.160}:
alternateipaddress: --->  {0.0.0.0}:
countryregion: -------->  {us}:
primaryclocksource: --->  {0/0/0/0/0}:
ringsource: ----------->  {internalringsourcelabel}:
revertiveclocksource: ->  {true}:
voicebandwidthcheck: -->  {false}:
alarm-levels-enabled: ->  {critical+major+minor+warning}: critical+major
userauthmode: --------->  {local}:
radiusauthindex: ------>  {0}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
zSH>
```

System maintenance

This section describes the following:

- *MXK file system* on page 93
- *Rename interfaces* on page 94
- *Configure card profiles* on page 101
- *Rename interfaces* on page 94
- *Save and restore configurations* on page 95
- *SNTP* on page 96
- *User accounts* on page 96
- *View chassis and slot information* on page 99
- *Control Telnet access* on page 100
- *Configure card profiles* on page 101

MXK file system

The following commands can be used to access the MXK file system:

- **cd** Changes directory.
- **dir** Lists the contents of the directory.
- **pwd** Displays the current working directory.
- **ata** Used to format or initialize a flash card. This is typically done only for new cards or if you want to completely erase the flash card.
- **image** Verifies software images and downloads software images on the flash to system memory.

The uplink card flash memory contains DOS file system that stores the system boot code, software images, and the configuration. During system startup, the software images on the flash are decompressed and loaded into memory.

Use the **cd**, **dir**, and **pwd** commands to list the contents of the file system, as in the following example:

```
zSH> cd /card1

zSH> pwd
/card1
```

```
zSH> dir
Listing Directory .:
-rwxrwxrwx  1 0       0          690408 May 14 02:03 mxup2tg8graw.bin
-rwxrwxrwx  1 0       0         9047733 May 14 05:28 mxup2tg8g.bin
-rwxrwxrwx  1 0       0         4412696 May 14 05:37 mxlcgp.bin
-rwxrwxrwx  1 0       0         3218033 May 14 05:45 mxlc20ae.bin
-rwxrwxrwx  1 0       0         1097975 Apr  3 15:45 malcmtac.bin
-rwxrwxrwx  1 0       0         1316447 Apr  3 15:46 malcmtacenh.bin
-rwxrwxrwx  1 0       0           33177 Apr  3 15:46 bkp.s
drwxrwxrwx  1 0       0            4096 May 14 17:10 datastor/
drwxrwxrwx  1 0       0            4096 May 13 23:41 onreboot/
drwxrwxrwx  1 0       0            4096 May 14 17:02 log/
drwxrwxrwx  1 0       0            4096 Mar 25 12:49 bulkstats/
drwxrwxrwx  1 0       0            4096 Mar 25 12:49 pub/
drwxrwxrwx  1 0       0            4096 Apr 28 17:09 omci/
-rwxrwxrwx  1 0       0         9041934 May  8 13:25 mxup2tg8g.bin.sv
-rwxrwxrwx  1 0       0         9044190 May 11 17:40 mxup2tg8g.bin.mlw
-rwxrwxrwx  1 0       0         3217274 May 13 22:50 mxlc20ae.bin.juss
-rwxrwxrwx  1 0       0         9048213 May 13 22:37 mxup2tg8g.bin.juss
-rwxrwxrwx  1 0       0         4412809 May 13 22:43 mxlcgp.bin.juss
                              184544980 bytes available
```

Using the ata command

The **ata** command formats and initializes flash cards. Formatting formats the files system, but leaves the boot partition on the card intact. Initialization reinitializes the boot partitions on the cards and formats the file system.

The following example formats flash card:

```
zSH> ata format 1
```

The following example initializes the flash card:

```
zSH> ata init 1
```

Download files

The MXK contains a TFTP server that enables you to download files from a network to the flash card file system using the **image** command.

The **image** command uses the following syntax:

```
image download tftphost image-file name destination
image-file name
```

The following example downloads the image for the uplink card (*mxkup2tg8graw.bin*) from host 192.168.8.21 to the root directory of the first flash card:

```
image download 192.168.8.21 mxup2tg8graw.bin
mxup2tg8graw.bin
```

The **image** command can also verify image files on the flash card. It reads the contents of the file, verifies the file header, and verifies the file checksum. For example:

```
zSH> image verify mxup2tg8graw.bin
File: mxup2tg8graw.bin
Size: 688320 bytes
Header Version: 1
Card Type: MX TWO TENGIGE EIGHT GIGE
Load Type: binary
Load Address: 0x00010000
Checksum: 0x2f66bb70
Image verify successful
```

The command reports any errors it finds in the file. Note that files are also verified as part of the download process.

Rename interfaces

Interfaces on the MXK can be renamed using the **ifName** parameter in the **if-translate** profile for the interface.

For example, to rename an uplink card interface:

```
zSH> update if-translate 1-a-1-0/eth
if-translate  1-a-1-0/eth
Please provide the following: [q]uit.
ifIndex: ----------->  {1}:
shelf: ------------->  {1}:
slot: -------------->  {a}:
port: -------------->  {1}:
subport: ----------->  {0}:
```

```
type: -------------->  {eth}:
adminstatus: ------->  {up}:
physical-flag: ----->  {true}:
iftype-extension: -->  {none}:
ifName: ------------>  {1-a-1-0}:
redundancy-param1: ->  {0}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
```

Save and restore configurations

The **dump** and **restore** commands enable you to save and restore the system configuration. You can save the configuration to the console, a local file, or the network.

The command uses the following syntax:

dump [**console**] [**file** ***filename***] [**network** ***host filename***]

Passwords are encrypted when they are saved to the configuration file. The encrypted passwords are used to restore the correct password, but cannot be used to log in.

Note: The **dump** and **restore** commands use TFTP to transfer files to the network. Set the TFTP server time-out value to at least 5 seconds, and 5 retries to help prevent TFTP timeout or retry errors.

To save the configuration to a console:

1. Configure your terminal emulation software as follows:
   – 9600bps
   – 8 data bits
   – No parity
   – 1 stop bit
   – No hardware flow control
   – VT100
   – Set Line Delay and Character Delay to 40 milliseconds
2. Turn on the file capture utility of your terminal emulation software.
3. Save the configuration by entering:
   ```
   dump console
   ```
   Do not press the Enter key.
4. Start the capture utility on your terminal emulation software and enter a name for the file (use a **.tx**t extension).
5. Press the Enter key.

   The configuration file will be displayed on the screen.

6 When configuration file is finished, stop the capture utility.

Backing up the configuration to a local file

To dump the configuration to a local file:

Specify a file name for the configuration:

```
zSH> dump file filename
```

The file is saved on the MXK file system.

Backing up the configuration to the network

To back up the configuration to the network:

1 Create the file in the destination location of the TFTP server and make it writeable.

2 Save the configuration. The following example saves the configuration to a file named device.cfg on the host 192.168.8.21:

```
zSH> dump network 192.168.8.21 device.cfg
```

Restoring the configuration

For information on restoring your configuration, refer to the release notes for your release.

SNTP

To set up the system to use SNTP:

Update the **ntp-client-config** profile. For example:

```
zSH> update ntp-client-config 0
Please provide the following: [q]uit.
primary-ntp-server-ip-address: --->  {0.0.0.0}: 192.168.8.100
secondary-ntp-server-ip-address: ->  {0.0.0.0}:
local-timezone: ------------------>  {gmt}: pacific
daylight-savings-time: ----------->  {false}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

User accounts

MXK users have access to the CLI and are able to configure and administer the system.

Adding users

Every administrative user on the system must have a user account. The account specifies their username and password, as well as their privilege level, which determines their access to commands.

Users with **admin** privileges have access to all the administrative commands. Users with **user** privileges have access to a very limited set of commands. The highest level of access is **useradmin**, which allows the creation of user accounts.

Note: When entering access level responses, enter **yes** completely or the CLI interprets the response as **no**.

To add a user, enter the following commands:

```
zSH> adduser
Please provide the following: [q]uit.
User Name: jjsmith
User Prompt[zSH>]:

Please select user access levels.
admin: ------->  {no}: yes
zhonedebug: -->  {no}:
voice: ------->  {no}:
data: -------->  {no}:
manuf: ------->  {no}:
database: ---->  {no}:
systems: ----->  {no}:
tool: -------->  {no}:
useradmin: --->  {no}: yes
..................................
User name:(jjsmith)  User prompt:(zSH>)
Access Levels:
(admin)(useradmin)
Save new account? [s]ave, [c]hange or [q]uit: s
User record saved.
TEMPORARY PASSWORD: hmj4mxFU
```

Commands with **zhonedebug** privilege levels are intended for use by Zhone development only.

Immediately after activating the user account, you should change the password something you can remember, as explained in the next section.

Changing default user passwords

When adding users, the system automatically assigns a temporary password to each user. Most users will want to change their password. The **changepass** command changes the password for the current logged in user. The following is an example of changing a password:

```
zSH> changepass
```

```
Current Password:
New Password:
Confirm New Password:
Password change successful.
```

Deleting users

To delete a user, enter the **deleteuser** command and specify the username:

```
zSH> deleteuser jsmith
OK to delete this account? [yes] or [no]: yes
User record deleted.
```

Deleting the admin user account

In addition to deleting regular user accounts, you can also delete the **admin** user account. This account is automatically created by the system and provides full access to the CLI.

Note: You cannot delete the **admin** account (or any other user account with **useradmin** privileges) if you are currently logged into it.

To delete the **admin** account:

```
zSH> deleteuser admin
```

If desired, you can recreate an account named **admin** after deleting it:

```
zSH> adduser admin
Please provide the following: [q]uit.
User Name: admin
User Prompt[zSH>]:

Please select user access levels.
admin: ------->  {no}: yes
zhonedebug: -->  {no}:
voice: ------->  {no}: yes
data: -------->  {no}: yes
manuf: ------->  {no}: yes
database: ---->  {no}: yes
systems: ----->  {no}: yes
tool: -------->  {no}: yes
useradmin: --->  {no}: yes
..................................
User name:(admin)  User prompt:(zSH>)
Access Levels:
(admin)(voice)(data)(manuf)(database)(systems)(tools)(use
radmin)
Save new account? [s]ave, [c]hange or [q]uit: s
User record saved.
TEMPORARY PASSWORD: hmj4mxFU
```

Resetting passwords

If a user forgets their password, an administrative user can reset the password and generate a new one using the **resetpass** command, as in the following example:

```
zSH> resetpass jsmith
Password:
```

View chassis and slot information

The following commands display information about the status of the system:

- **shelfctrl**
- **slots**

To view overall status of the system, use the **shelfctrl monitor** command:

```
zSH> shelfctrl monitor
Shelf uptime: 31 minutes
Shelf Monitor CPLD version: 0.5
Shelf Monitor Firmware version: 0.5
Outlet temperature sensor: 24 degrees C (normal)
Fan Power A: normal
Fan Power B: normal
Power Supply A: normal
Power Supply B: failure
Fan status: OK.
System: Critical alarm set.
Card a: Critical alarm set.
```

To view general system statistics:

```
zSH> shelfctrl stats
Shelf Controller Message Statistics
-----------------------------------
Card updates: 9
Card ECHO: 0
Directory services messages: 2
Clock messages: 6464
Lease messages: 23
Heartbeat messages: 8338
Card update errors: 0
Card ECHO errors: 0
Directory services errors: 0
Clock errors: 0
Lease errors: 0
Heartbeat errors: 0
Receive errors: 0
No Peer heartbeat check errors: 0
```

To verify whether the shelf is active:

```
zSH> shelfctrl show
```

```
Shelf Controller Address: 01:a:12
Shelf Registry Address: 01:a:1038
Lease ID: 0x02070000_00000037
State: active
Slot 1:
   prevState: NONE currentState: NONE
 mode: NONE startTime: 0
Slot 2:
 prevState: CONFIGURING currentState: RUNNING
mode: FUNCTIONAL startTime: 1205872814
```

To view the system slot cards and their status:

```
zSH> slots
Uplinks
 a:*MX TWO TENGIGE EIGHT GIGE (RUNNING)
Cards
 2: MX LINE CARD 2 PORT GPON (RUNNING)
 7: MX LINE CARD 2 PORT GPON (NOT_PROV)
```

To view information about a particular slot card, use the **slots** command and specify a slot number. For example:

```
zSH> slots a
Type            :*MX TWO TENGIGE EIGHT GIGE
Card Version    : 1
EEPROM Version  : 1
Serial #        : 3250057
CLEI Code       : No CLEI
Card-Profile ID : 1/a/10100
Shelf           : 1
Slot            : a
ROM Version     : development
Software Version: MXK CAN 1.14.2.125
State           : RUNNING
Mode            : FUNCTIONAL
Heartbeat check : enabled
Longest hbeat   : 51
Fault reset     : enabled
Uptime          : 44 minutes
```

Control Telnet access

The **port-access** profile specifies from which IP addresses users can telnet to the MXK. If a host’s IP address is not specified in a **port-access** profile, users from that host cannot telnet to the MXK. These restrictions take effect after the first **port-access** profile has been created.

By default, no **port-access** profiles are created, so telnet access is not restricted.

Creating port-access profile entries

Up to 100 **port-access** profile entries can be created on a MXK. To create a **port-access** profile entry:

Create a new port-access profile and specify the telnet port number, host/network IP address to be granted access, and the netmask applied to the IP address to allow access to a range of IP addresses.

This example creates port-access entry 1 on telnet port 23 and allows hosts on the 172.16.41.xx network to telnet to the MXK.

Note: Typically, only port 23 is used for telnet access.

```
zSH> new port-access 1
Please provide the following: [q]uit.
portNumber: ->  {0}: 23
portArg1: --->  {0.0.0.0}: 172.16.41.0
portArg2: --->  {0.0.0.0}: 255.255.255.0
....................S=
Save new record? [s]ave, [c]hange or [q]uit: s
New record saved.
```

Displaying port-access profile entries

To display configured **port-access** profile entries use the **list** command:

```
zSH> list port-access
port-access 1
1 entry found.
```

Modifying port-access profile entries

To modify a configured port-access profile entry use the **update** command. The following example changes the entry’s source IP address to 172.16.40.0:

```
zSH> update port-access 1
portNumber: ->  {23}
portArg1: --->  {172.16.41.0} 172.16.40.0
portArg2: --->  {255.255.255.0}
1 entry found.
....................
Save new record? [s]ave, [c]hange or [q]uit: s
Updated record saved.
```

Configure card profiles

Table 8 provides the MXK card types.

Table 8: MXK card types

Card name	Type	Name of software image
MXK-UPLINK-8X1G 8-port 1-GigE uplink card	10101	mxupg8g.bin
MXK-UPLINK-2X10GE-8X1G 2-port 1-GigE 8-port 1-GigE uplink card	10100	*mxup2tg8g.bin*
MXK-AEX20-FE/GE-2S 20-port FE/GE line card	10200	*mxkc20ae.bin*
MXK-GPONX2-IO 2-port GPON line card	10201	*mxlc4gp.bin*

Add a card profile

Adding a card profile

If necessary, use the **slots** command to verify which slot a card resides in before using the **card add** command to provision the card. To provision a card, the card must be installed in a slot.

1 To verify the location of a card, enter **slots**:

```
zSH> slots
Uplinks
 a:*MX TWO TENGIGE EIGHT GIGE (RUNNING)
 b: MX TWO TENGIGE EIGHT GIGE (RUNNING)
Cards
 8: MX LINE CARD 20 ACT ETH (RUNNING)
10: MX LINE CARD 2 PORT GPON (NOT_PROV)
11: MX LINE CARD 2 PORT GPON (NOT_PROV)
12: MX LINE CARD 2 PORT GPON (NOT_PROV)
```

2 To provision a card, enter **card add** *shelf/slot/type:*

```
zSH> card add 1/12/10201
new card-profile 1/12/10201 added, sw-file-name
"mxlcgp.bin"
```

3 To verify the state of the provisioning, enter **slots** again:

```
zSH> slots
Uplinks
 a:*MX TWO TENGIGE EIGHT GIGE (RUNNING)
 b: MX TWO TENGIGE EIGHT GIGE (RUNNING)
Cards
 8: MX LINE CARD 20 ACT ETH (RUNNING)
10: MX LINE CARD 2 PORT GPON (NOT_PROV)
11: MX LINE CARD 2 PORT GPON (NOT_PROV)
```

```
12: MX LINE CARD 2 PORT GPON (LOADING)
```

Delete a card profile

Deleting a card profile

Deleting a card, deletes the card-profile interface.

Caution: Before deleting card profiles, perform the following:

- Back up the MXK configuration. See the release notes for information.
- For voice cards, ensure all subscribers and voice profiles are deleted before deleting the card.
- Remove the card from the system as explained in the *MXK Hardware Installation Guide*.

Delete the card-profile for a card to delete all the profiles associated with a card. After deleting a card-profile, the specified card reboots.

Caution: A **delete card-profile** command deletes profiles associated with the card and may disrupt service until the system is reprovisioned.

The **card delete** command uses the following syntax:

card delete shelf/*slot/type*

```
zSH> card delete 1/1/10200
card-profile 1/1/10200 deleted
```

The following slots commands show the change of status of the Active Ethernet card in slot 1 immediately after entering card delete. The state of the card changes from running, to booting, to not provisioned.

```
zSH> slots
Uplinks
 a:*MX EIGHT GIGE (RUNNING)
 b: MX EIGHT GIGE (NOT_PROV)
Cards
 1: MX LINE CARD 20 ACT ETH (RUNNING)
18: MX LINE CARD 2 PORT GPON (NOT_PROV)

zSH> slots
Uplinks
 a:*MX EIGHT GIGE (RUNNING)
 b: MX EIGHT GIGE (NOT_PROV)
Cards
 1: MX LINE CARD 20 ACT ETH (BOOTING)
18: MX LINE CARD 2 PORT GPON (NOT_PROV)

zSH> slots
```

```
Uplinks
 a:*MX EIGHT GIGE (RUNNING)
 b: MX EIGHT GIGE (NOT_PROV)
Cards
 1: MX LINE CARD 20 ACT ETH (NOT_PROV)
18: MX LINE CARD 2 PORT GPON (NOT_PROV)
```

Note: You can only delete one card at a time. Wildcards are not supported when deleting cards and their profiles.

INDEX