EtherXtend 3300 Series User’s Guide

For software version 1.14.2
June 2008
Document Part Number: 830-01613-06

Z H O N E ™

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

CONTENTS

About This Guide

This guide is intended for use by EtherXtend users and administrators. EtherXtend users should have a fundamental knowledge of DSL protocols, Ethernet, and IP concepts.

Refer to the *EtherXtend Quick Start Guide* for hardware installation and connection information.

Style and notation conventions

The following conventions are used in this document to alert users to information that is instructional, warns of potential damage to system equipment or data, and warns of potential injury or death. Carefully read and follow the instructions included in this document.

Caution: A caution alerts users to conditions or actions that could damage equipment or data.

Note: A note provides important supplemental or amplified information.

Tip: A tip provides additional information that enables users to more readily complete their tasks.

WARNING! A warning alerts users to conditions or actions that could lead to injury or death.

WARNING! A warning with this icon alerts users to conditions or actions that could lead to injury caused by a laser.

Typographical conventions

The following typographical styles are used in this guide to represent specific types of information.

Bold	Used for names of buttons, dialog boxes, icons, menus, profiles when placed in body text, and property pages (or sheets). Also used for commands, options, parameters in body text, and user input in body text.
`Fixed`	Used in code examples for computer output, file names, path names, and the contents of online files or directories.
`Fixed Bold`	Used in code examples for text typed by users.
`Fixed Bold Italic`	Used in code examples for variable text typed by users.
Italic	Used for book titles, chapter titles, file path names, notes in body text requiring special attention, section titles, emphasized terms, and variables.
PLAIN UPPER CASE	Used for environment variables.
Command Syntax	Brackets [ ] indicate optional syntax. Vertical bar | indicates the OR symbol.

Acronyms

The following acronyms are related to the EtherXtend and will appear throughout this manual:

Acronym	Description
AAL2	ATM Adaption Layer 2
CAS	Channel Associated Signaling
CID	AAL2 Channel Identifier
CLI	Command Line Interface
CO	Central Office
CPE	Customer Premises Equipment
dB	Decibel
DSL	Digital Subscriber Line
DSLAM	Digital Subscriber Line Access Multiplexer
DSS1	Digital Subscriber Signaling System number 1
EFM	Ethernet First Mile
ELCP	Emulated Loop Control Protocol
G.SHDSL	Global Symmetrical High-bit-rate Digital Subscriber Line

Acronym	Description
IANA	Internet Assigned Numbers Authority
Kbps	Kilobytes per second
LAN	Local Area Network
MALC	Zhone Multi-Access Loop Concentrator
MIB	Management Information Base
NAT	Network Address Translation
PMA	Physical Medium Attachment
PMD	Physical layer, media dependent
PME	Physical Medium Entities
SDSL	Symmetric Digital Subscriber Line
SHDSL	Single pair high speed Digital Subscriber Line
SNMP	Simple Network Management Protocol
TDM	Time Division Multiplexing
TFTP	Trivial File Transfer Protocol
VCI	Virtual Channel Identifier
VCL	Virtual Channel Link
VPI	Virtual Path Identifier
WAN	Wide Area Network

Related documents

Refer to the following publications for additional information:

- *EtherXtend Quick Start Guide*
- *EtherXtend Release Notes*

Contacting Global Service and Support

Contact Global Service and Support (GSS) if you have any questions about this or other Zhone products. Before contacting GSS, make sure you have the following information:

- Zhone product you are using
- System configuration
- Software version running on the system

- Description of the issue

Technical support

If you require assistance with the installation or operation of your product, or if you want to return a product for repair under warranty, contact GSS. The contact information is as follows:

E-mail	support@zhone.com
Telephone (North America)	877-ZHONE20
Telephone (International)	510-777-7133
Internet	www.zhone.com/support

If you purchased the product from an authorized dealer, distributor, Value Added Reseller (VAR), or third party, contact that supplier for technical assistance and warranty support.

Service requirements

If the product malfunctions, all repairs must be performed by the manufacturer or a Zhone-authorized agent. It is the responsibility of users requiring service to report the need for service to GSS.

1 OVERVIEW

The EtherXtend devices are next-generation Ethernet-enabled customer premises equipment (CPE). EtherXtend provides 4-port and 8-port models that perform the functions of a network extender by lengthening the reach of Ethernet packets. EtherXtend allows Ethernet packets to pass over existing copper wires that link the service provider (a central office, street cabinet, pole, or tower) to a subscriber's CPE. This distance is known as Ethernet in the First Mile (EFM). EFM is a set of specifications that allow users to run Ethernet protocols over previously unsupported media such as single pairs of copper wires in subscriber access networks. EtherXtend EFM is covered by IEEE standard 802.3ah.

This chapter includes the following topics:

Product models

Table 1 provides the model numbers for the EtherXtend products.

Table 1: EtherXtend models

Model Name	Description
ETHX-3344-US	4-port SHDSL EFM device running on alternating current for US power requirements.
ETHX-3344-UK	4-port SHDSL EFM device running on alternating current for United Kingdom power requirements.
ETHX-3344-EU	4-port SHDSL EFM device running on alternating current for European Union power requirements.
ETHX-3344-DC	4-port SHDSL EFM device running on direct current power requirements.
ETHX-3384-US	8-port SHDSL EFM device running on alternating current for US power requirements.
ETHX-3384-UK	8-port SHDSL EFM device running on alternating current for United Kingdom power requirements.
ETHX-3384-EU	8-port SHDSL EFM device running on alternating current for European power requirements.
ETHX-3384-DC	8-port SHDSL EFM device running on direct current power requirements.

Product description

The 4- and 8-port EttherXtend SHDSL Ethernet access devices (ETHX-SHDSL-4 and ETHX-SHDSL-8) deliver bonded high-speed EFM services over SHDSL. The ETHX-SHDSL-x device family provides 802.3ah compliant SHDSL bonding with advanced features, including Zhone's Multimedia Traffic Management (MTM). In addition, full compliance to 802.3ah industry standards provide high reliability, low latency, and integrated operation, administration, and maintenance (OAM) features for advanced management of Ethernet lines.

Figure 1: Rear view of the EtherXtend 8-port device

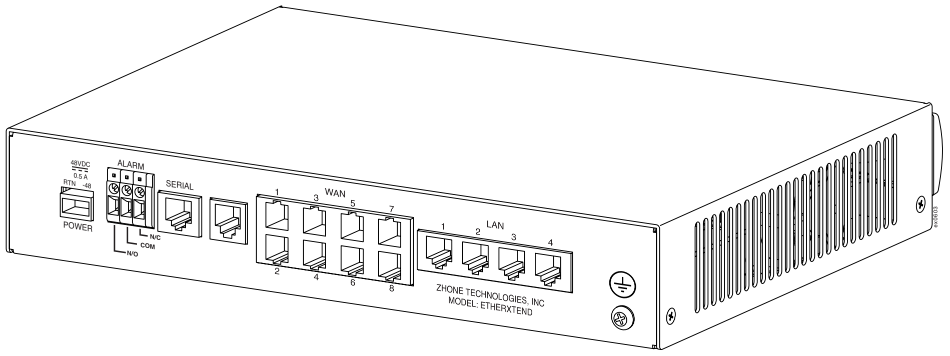
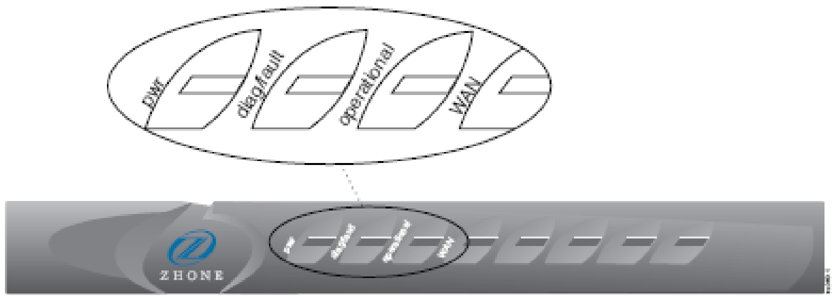


Figure 2: Front view of the LEDs on the EtherXtend device



With Zhone's proprietary lightweight Ethernet bonding technologies, EtherXtend provides dual-mode operations to allow EtherXtend to aggregate into standards-based products such as the MALC broadband loop carrier, and other EtherXtend-SHDSL-x units. EtherXtend can also connect into pre-standard Ethernet loop bonding products such as Zhone's Ethernet access products.

Technology description

EFM provides high performance for data traveling across a connection between the subscriber and the service provider. This connection has been a stopgap for Internet traffic, limiting performance to whatever the constraints of this connection are. Regardless of how fast data could travel over xDSL network links, the data usually would be slowed down on the last link connecting the subscriber to the service provider.

The EtherXtend provides an ideal solution for the following scenarios:

- LAN extensions in metro areas
- a cellular site backhaul

- any point-to-point application requiring Ethernet connectivity over the WAN

The EtherXtend can be used to create a point-to-point connection over a dry copper pair. Within a campus environment or multi-tenant unit (for example, a hospital) where the subscriber owns the cabling infrastructure, copper pairs are often available throughout the facility as unused telephone cabling.

Any cell site running EvDO CDMA or 3G provides an Ethernet interface for mobile applications such as mini-web browsing. Typically, these Ethernet ports are connected to a T1 router so they can be carried over long distances to the CO where they are terminated into Ethernet.

The EtherXtend platforms not only provide a more cost-effective alternative to deploying dedicated T1 router equipment at each cellular tower, but they also provide loop bonding capabilities to reach the higher bandwidth demands of EvDO and 3G.

The Ethernet port of the cell tower simply connects to the appropriate EtherXtend platform, and the Ethernet frames are transported directly to four SHDSL lines. At the CO, the extended LAN connection presents a single Ethernet RJ-45 interface for simple connectivity to the data network.

Loop bonding allows multiple physical lines to be grouped together to achieve higher data rates. For example, four 5.7 Mbps SHDSL lines can be combined to provide up to 20 Mbps of bandwidth. Loop bonding is an easy way to increase line speeds where extra copper is available.

Zhone’s EtherXtend technology uses existing copper facilities to deliver high bandwidth Ethernet services over existing copper loops to business customers. Products from this technology are both EFM standards compliant and compatible with existing bonded copper solutions to deliver point-to-point Ethernet connectivity over the WAN.

EtherXtend transports Ethernet directly over SHDSL, T1, or E1. This makes it easy to interconnect LANs over virtually any distance. These transparent LAN services allow businesses with broadly distributed remote offices to operate as if located on the same local network.

EFM media

Copper wire access lines are the dominant access media today. While optical fiber facilities appear to be gaining market share, the existing media (copper) still is more dominant in terms of volume of footage, number of entry points, and number of regions governed by a technology type.

EFM over copper services offers more than fiber class speed. They support a wide range of applications and opportunities. EFM technology provides benefits for installations provisioning high-speed, high-value services and for users of those services who realize cost and performance improvements delivered rapidly and flexibly.

EFM over copper applications include:

- Frame Relay migration to E-LAN services
- T1/E1 replacement with E-Line services, including internet access, wireless backhaul, voice access, and wholesale services.
- Transparent LAN service (TLS)

EFM over copper technology supports a wide range of voice, data, and video services and applications. Many applications are extensions of those now supported by T1/E1 and Frame Relay services. However, they are more responsive to new demands by customers by providing much higher data rates.

Ethernet services

The Metro Ethernet Forum (MEF) has helped to define standards for Ethernet services. These standards helped to establish the user network interface and Ethernet virtual connection for two principal services: E-Line and E-LAN.

E-Line services include point-to-point Ethernet connections using Layer 2 VLAN infrastructure with two types of user network interface (UNI) access: 802.1Q-in-802.1Q (QinQ) and dot1Q encapsulation. These services are also known as Ethernet Relay Service (ERS) and Ethernet Wire Service (EWS).

E-LAN services provide multipoint-to-multipoint services using virtual private LAN services (VPLS) or simply layer 2 VLAN core.

Carrier Ethernet has developed a series of extensions to standard Ethernet that provide improved performance on the Internet.

2 Safety, Regulations, and Certifications

This chapter describes how to prepare your site for the installation of the EtherXtend platform. It includes the following topics:

- Grounding and isolation, page 17
- Installation safety precautions on page 17
- EMI notices, page 20
- ACTA customer information, page 20
- Supplier's declaration of conformity, page 22
- CE marking, page 23
- Contacting Global Service and Support, page 23
- Technical support, page 23
- Service requirements, page 24

Grounding and isolation

The EtherXtend uses an integrated frame and logic ground system as follows:

- The EtherXtend device and logic ground are bonded.
- Cable shielding is terminated on the EtherXtend system device ground.

When the AC plug is not grounded, it is recommended to ground the device using minimum 16-gauge wire to a building or earth ground.

Installation safety precautions

Avoid creating a hazardous condition by maintaining even weight distribution within the device.

Maximum operating temperature should not exceed 65^{0}C (149^{0}F). Observe the maximum recommended operating temperature as indicated here.

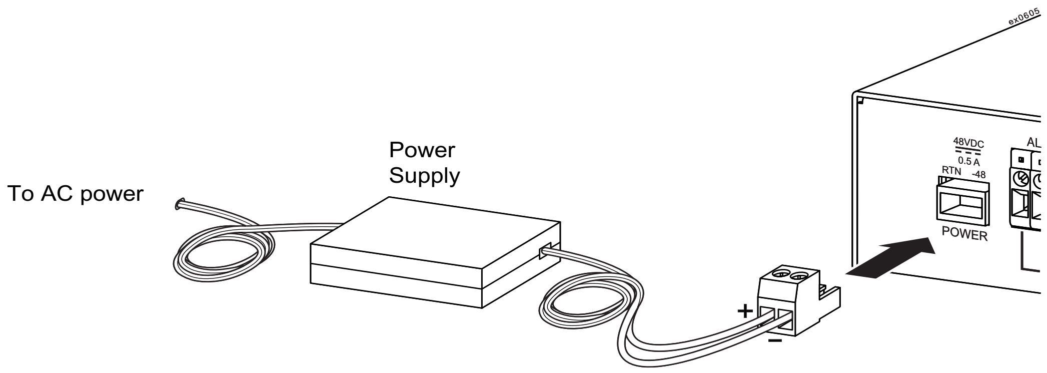
Do not block system air vents; this will deprive the system of the airflow required for proper cooling. Sufficient clearance must exist on all sides of the rack to permit equipment access. Connect the system to the power supply circuit as described in this document. Do not overload the system or power supply circuit. Ensure that proper system grounding is performed and maintained.

Important safety instructions

Read and follow all warning notices and instructions marked on the product or included in the manual.

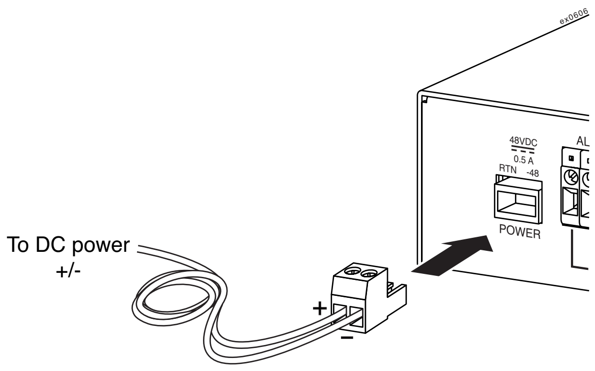
1. Slots and openings in the product are provided for ventilation. To ensure reliable operation of the product and to protect it from overheating, these slots and openings must not be blocked or covered.
2. Do not allow anything to rest on the power cord and do not locate the product where persons will walk on the power cord.
3. Do not attempt to service this product yourself, as opening or removing covers may expose you to hazardous voltage or to other risks. Refer all servicing to qualified service personnel.
4. General purpose cables are used with this product for connection to the network. Special cables, which may be required by the regulatory inspection authority for the installation site, are the responsibility of the customer. Use a UL Listed, CSA certified (or a cable that is certified in the country in which it is being installed), minimum No. 26 AWG (.163mm^2) line cord for connection to the Digital Subscriber Line (DSL) network.
5. When installed, the product must comply with the applicable Safety Standards and regulatory requirements of the country in which it is installed. If necessary, consult with the appropriate regulatory agencies and inspection authorities to ensure compliance.
6. A rare phenomenon can create a voltage potential between the earth grounds of two or more buildings. If products installed in separate buildings are interconnected, the voltage potential may cause a hazardous condition. Consult a qualified electrical consultant to determine whether or not this phenomenon exists and, if necessary, implement corrective action prior to interconnecting the products.
7. When using a certified class II transformer/Input power to this product must be provided with one of the following: (1) a NRTL certified power source with a Class 2 output for use in North America, or (2) a certified power source, with a Safety Extra Low Voltage (SELV) output having a maximum of 240 VA available, for use in the country of installation.

Figure 3: AC feed power connection



For DC direct feed operation: Connect the 48 VDC SELV supply source that is electrically isolated from the AC source. Use Stranded 18 AWG. (1.045mm^2) gauge wire, type SPT or HO3 wire. The length may vary up to a maximum of 6 feet. All conductors on both ends of the wire should be stripped back .25 inches, but not tinned. Use Figure 4 as a guide to wire the two-wire terminal provided. The 48 VDC source is to be reliably connected to earth. Ground by attaching an earthing ground wire to the Sem Screw Zhone P/N 150-00071-01 (8-32 x 1/2 split lock steel screw) provided on the rear panel of the chassis next to the earth ground symbol. The wire should be a minimum of 14 AWG (2.7 mm^2) grounding conductor with insulation colored green with a yellow strip and should connect to a copper grounding lug Heyco P/N 1851 (Zhone P/N 170-02880-01) or equivalent. The copper ground lug will connect to grounding screw terminal located on the back of the EtherXtend chassis next to the earth ground symbol.

Figure 4: DC feed power connection



In addition, since the equipment is to be used with telecommunications circuits, take the following precautions:

- Never install telephone wiring during a lightning storm.
- Never install telephone jacks in wet locations unless the jack is specifically designed for wet locations.
- Never touch uninsulated telephone wires or terminals unless the telephone line has been disconnected at the network interface.
- Use caution when installing or modifying telephone lines.
- Avoid using a telephone (other than a cordless type) during an electrical storm. There may be a remote risk of electric shock from lightning.
- Do not use the telephone to report a gas leak in the vicinity of the leak.

EMI notices

The following are EMI notices for the United States.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

The authority to operate this equipment is conditioned by the requirements that no modifications will be made to the equipment unless the changes or modifications are expressly approved by Zhone Technologies, Inc.

If the equipment includes a ferrite choke or chokes, they must be installed as described in the installation instructions.

Canada - EMI notice

This Class A digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

ACTA customer information

This equipment complies with Part 68 of the FCC rules and the requirements adopted by the ACTA. On the bottom of the network extender is a label that contains, among other information, a product identifier in the format US:AAAEQ##TXXXX. If requested, this number must be provided to the telephone company.

A plug and jack used to connect this equipment to the premises wiring and telephone network must comply with the applicable FCC Part 68 rules and requirements adopted by the ACTA. See installation instructions for details.

If the network extender causes harm to the telephone network, the telephone company will notify you in advance that temporary discontinuance of service may be required. But if advance notice isn't practical, the telephone company will notify the customer as soon as possible. Also, you will be advised of your right to file a complaint with the FCC if you believe it is necessary.

The telephone company may make changes in its facilities, equipment, operations or procedures that could affect the operation of the equipment. If this happens the telephone company will provide advance notice in order for you to make necessary modifications to maintain uninterrupted service.

If trouble is experienced with this equipment, please contact your local sales representative, service representative, or distributor directly for any help needed. For additional information concerning warranty, sales, service, repair, installation, documentation, training, distributor locations, or Zhone Technologies worldwide office locations, contact Global Service and Support.

If the equipment is causing harm to the telephone network, the telephone company may request that you disconnect the equipment until the problem is resolved.

The customer may make no repairs to the equipment.

Connection to party line service is subject to state tariffs. Contact the state public utility commission, public service commission or corporation commission for information.

Supplier’s declaration of conformity

Place of Issue: Zhone Technologies

8545 126th Avenue North

Largo, FL 33773-1502

USA

Date of Issue: July 10, 2006

Zhone Technologies, located at the above address, hereby certifies that the Model ETHX-SHDSL-X (where X = 4 or 8 denoting the total number of WAN ports) bearing labeling identification number US:6RTDLNANETHX complies with: the Federal Communications Commission’s (“FCC”) Rules and Regulations 47 CFR Part 68, and the Administrative Council on Terminal Attachments (“ACTA”)-adopted technical criteria TIA-968-A, “Telecommunications - Telephone Terminal Equipment -Technical Requirements for Connection of Terminal Equipment To the Telephone Network, October 2002,” as amended by TIA-968-A-1, TIA-968-A-2 and TIA-968-A-3.

Keith Nauman

Keith Nauman
Vice President

Notice to Users of the Canadian Telephone Network

NOTICE: This equipment meets the applicable Industry Canada Terminal Equipment Technical Specifications. This is confirmed by the registration number. The abbreviation IC before the registration number signifies that registration was performed based on a Declaration of Conformity indicating that Industry Canada technical specifications were met. It does not imply that Industry Canada approved the equipment.

NOTICE: The Ringer Equivalence Number (REN) for this terminal equipment is labeled on the equipment. The REN assigned to each terminal equipment provides an indication of the maximum number of terminals allowed to be connected to a telephone interface. The termination on an interface may consist of any combination of devices subject only to the requirement that the sum of the Ringer Equivalence Numbers of all the devices does not exceed five.

If your equipment is in need of repair, contact your local sales representative, service representative, or distributor directly.

CE marking

When the product is marked with the CE mark on the equipment label, a supporting Declaration of Conformity may be downloaded from the Zhone World Wide Web site at www.zhone.com.

Contacting Global Service and Support

Contact Global Service and Support (GSS) if you have any questions about this or other Zhone products. Before contacting GSS, make sure you have the following information:

- Software version running on the system
- System configuration
- Zhone product you are using
- Description of the issue

Technical support

If you require assistance with the installation or operation of your product, or if you want to return a product for repair under warranty, contact GSS.

Table 2: Contact information

E-mail	**support@zhone.com**
Telephone (North America)	877-ZHONE20
Telephone (International)	510-777-7133

Table 2: Contact information (Continued)

E-mail	**support@zhone.com**
Internet	www.zhone.com/support

If you purchased the product from an authorized dealer, distributor, Value Added Reseller (VAR), or third party, contact that supplier for technical assistance and warranty support.

Service requirements

If the product malfunctions, all repairs must be performed by the manufacturer or a Zhone-authorized agent. It is the responsibility of users requiring service to report the need for service to GSS.

3 Features and Capabilities

Zhone EtherXtend provides EFM functionality for use in subscriber access networks. EtherXtend products have a series of features to deliver performance enhancements such as bonding and aggregation to the networks.

This chapter provides:

Ethernet in the first mile (EFM) overview

Ethernet in the First Mile (EFM), is a collection of protocols (IEEE 802.3ah) for Ethernet access and management technology across both the copper and fiber network infrastructure.

Principally, EFM was established to enhance subscriber access networks. EFM technology provides the following benefits:

- Ethernet interfaces and transport over previously unsupported media, such as single pairs of telephone copper wires
- A universal Ethernet service extending the reach of Ethernet to locations served by both copper and fiber
- A consistent set of Ethernet services with adjustable bandwidth and rates from 1 Mbps to 10 Gbps.

2BASE-TL

EFM over Copper (EFMC) is a point-to-point Ethernet over copper standard targeted at small to medium enterprise sites served by copper. The two protocols associated with EFM are: 2BASE-TL and 10PASS-TS. Zhone currently supports only 2BASE-TL. The following table provides details about 2BASE-TL.

Table 3: 2BASE-TL protocol

Protocol	Point Method	Standard	Media	Mode	Data Rate	Distance	Bonding
2BASE-TL	Point-to-Point	SHDSL	Copper	Symmetric	1 - 5.7 Mbps	2,700 meters	32 pairs; 8 pairs (actual)

2BASE-TL is based on the SHDSL standard for symmetric DSL services. The data rates over a single copper pair range between 1 Mbps and 5.7 Mbps. 2BASE-TL can serve distances up to 2,700 meters. Theoretically, up to 32 copper pairs can be bonded together to increase data rates and resiliency. In practice, no more than 8 pairs are bounded.

The standard closely resembles the SHDSL specification for the PMA and PMD sublayers. The PMA sublayer consists of a framer/deframer and a scrambler/descrambler. The PMA sublayer then sends a framed and scrambled bit stream to the PMD layer for modulation over the single copper pair.

The PMD sublayer first passes the bit stream through a convolutional encoder/decoder and then through the PCM modulator/demodulator. The PMD sublayer then passes electrical signals across a single pair of voice grade copper lines.

Zhone EFM features

Zhone’s EtherXtend equipment adds enhancements to the EFM standard. Some of these enhancements are:

- PME aggregation on page 26
- Loop bonding on page 27

PME aggregation

Physical Medium Entities (PME) aggregation uses *frame fragmentation* to divide, load balance, and transmit MAC frames across up to 32 parallel links.

Frame fragmentation is used to so that MAC frames are broken up into smaller pieces and sent in parallel across multiple links. This increases effective throughput while minimizing latency and jitter across the bonded links.

Fragment sizes must be between 64 and 512 bytes and must be multiples of 64 bytes. All fragments must be at least 64 bytes long. The algorithm for fragmenting MAC frames is left up to the implementer so there can be differences in performance between different vendors in transmitting data across bonded links.

Loop bonding

Ethernet loop bonding refers to a technology where multiple physical SHDSL lines are grouped together to provide greater bandwidth potential over a single logical connection. The WAN connection terminates into a single Ethernet connection, providing a single connection to the end user.

However multiple lines aggregate on the WAN to provide higher bandwidth. This enables carriers to provide greater distances for higher bandwidth services, for example, IP television, VoIP, and other converged services.

In DSL loop bonding, two or four copper pairs are utilized to yield up to 8x the bandwidth at any given distance. This allows extremely long local loops to deliver high-speed DSL. For example, while SHDSL only supports 144 Kbps at 25,000 feet, four SHDSL lines together would provide 576 Kbps or just over .5 Mbps. The loop bonding aspect of the connection is invisible to the consumer who realizes an increase in access speed.

The following table shows the data rates that bonded loops can produce.

Table 4: Selected data rates (in Kbps) realized through bonding loops

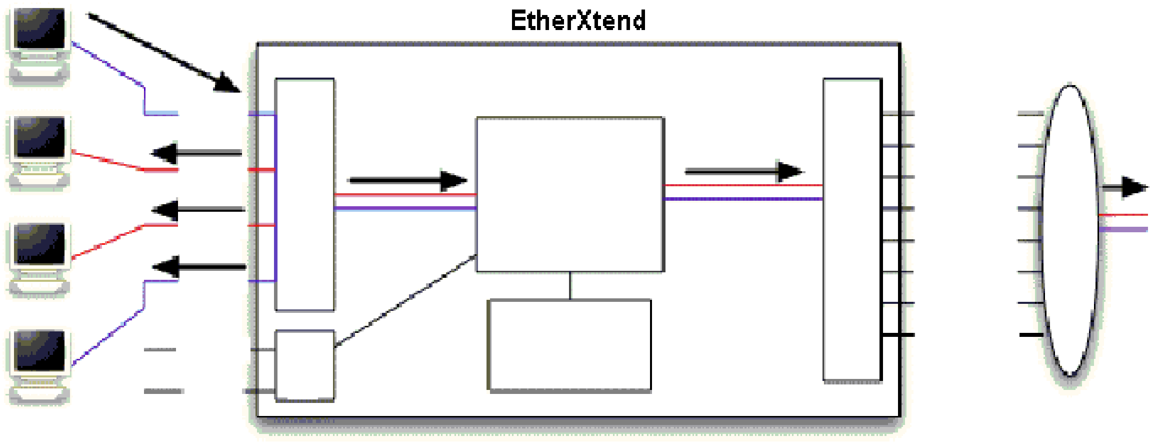
Distance (feet)	CO-to-CPE (1 Pair)	CO-to-CPE (2 Pair)	CO-to-CPE (4 Pair)	CO-to-CPE (8 Pair)
5,000	5,704	11,408	22,816	45,632
7,000	5,704	11,408	15,936	28,800
9,000	3,856	5,792	9,536	18,560
12,000	2,192	4,384	7,488	10,368
15,000	1,040	2,080	3,392	5,248
18,000	464	928	1,856	3,200

While the total achievable distance is not increased through loop bonding, the maximum bandwidth that can be achieved at any distance is increased, making even very long DSL deployments capable of high data rates.

EtherXtend bonding implementation

The following sections detail various implementations of EtherXtend. The following implementations are provided.

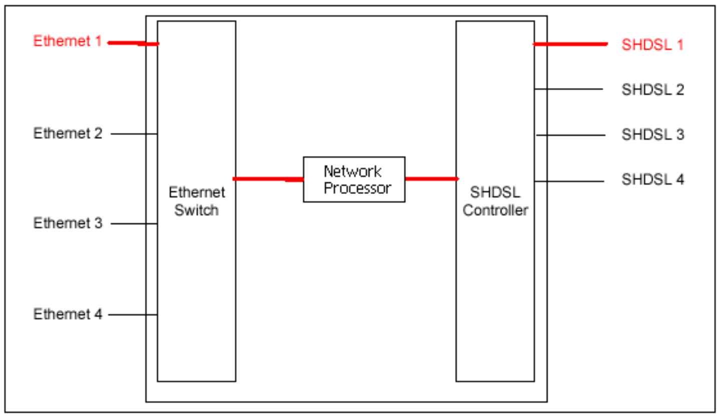
- One Ethernet port mapped to one SHDSL port on page 28
- One Ethernet port mapped to an SHDSL bonded group on page 29
- Ethernet ports mapped to different SHDSL bonded groups on page 29
- Multiple Ethernet ports mapped to multiple SHDSL ports on page 30



Unicast traffic destined for remote (SHDSL) ports will not be forwarded to other local Ethernet ports. However, in wire mode, packets are not processed by SLMS code within the NPU; they are cut-through to the SHDSL ports directly.

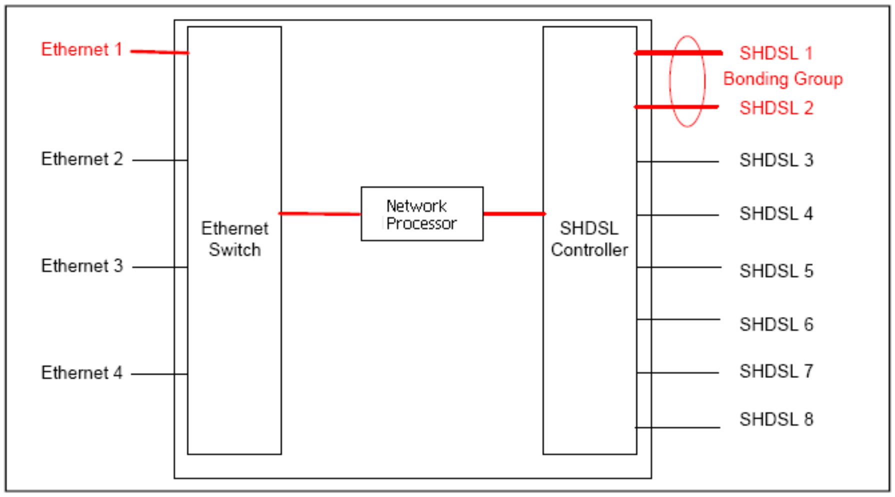
One Ethernet port mapped to one SHDSL port

This configuration is a simple LAN extension application. In this mode, the device is simply a bridge. Backhaul from a cell tower might use this configuration. Instead of an Ethernet to T1 conversion, this approach would allow native Ethernet throughout the network.



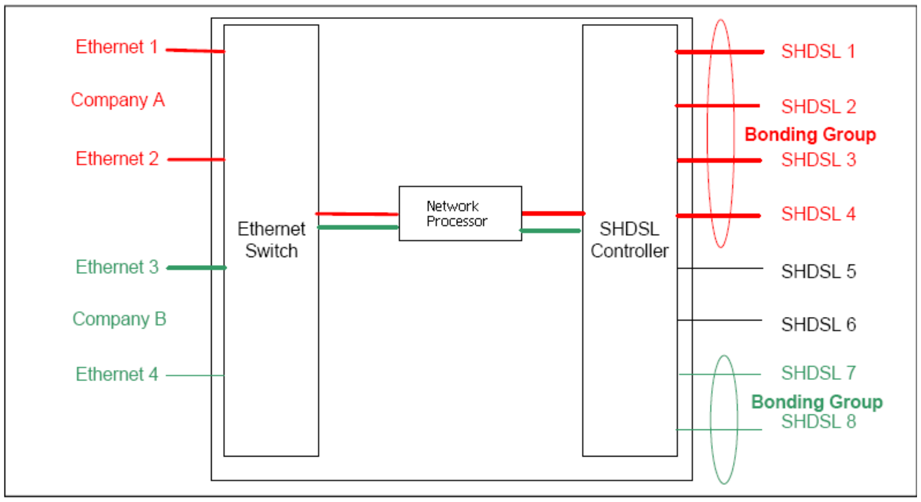
One Ethernet port mapped to an SHDSL bonded group

This configuration increases the speed of the WAN link. This configuration applies to the same scenarios as the instance where one Ethernet port is mapped to one SHDSL port. However, you use this scenario when there is a need for greater bandwidth. With two ports bonded together, you can raise your throughput to 11.4 Mbps. This is appropriate for bursty LAN traffic because the Ethernet port support throughput at 100 Mbps and the DSL port supports throughput at 11 Mbps.



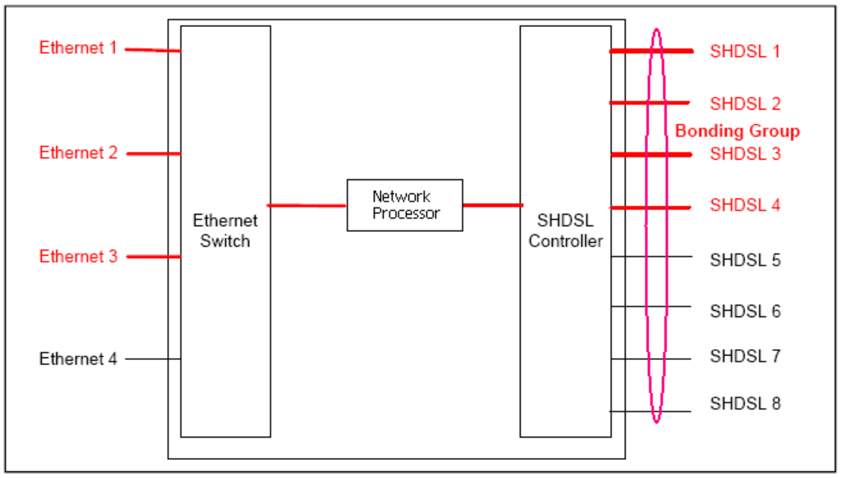
Ethernet ports mapped to different SHDSL bonded groups

In this configuration, two independent Ethernet ports are connected to different bonded SHDSL groups. This configuration operates as two Ethernet extenders in the same physical unit. This feature allows for transparent LAN services. It could be possible for Company A to use Ethernet port 1 and Company B to use Ethernet port 3. With features such as Plans, the traffic from each company would be separated. A better solution could be to use just two units. Then the data is physically as well as logically separated.



Multiple Ethernet ports mapped to multiple SHDSL ports

This configuration allows multiple PCs to connect to the EAD eliminating the need for an additional router or switch for multiple users. In addition, the WAN link speed is significantly improved. This configuration is similar to the scenario where multiple Ethernet switch ports map to one SHDSL port. However, the WAN link speed is much faster. with al eight ports bonded, the WAN speed can be up to 45 Mbps.



Scenarios

This section provides details on possible usage scenarios for the EtherXtend device:

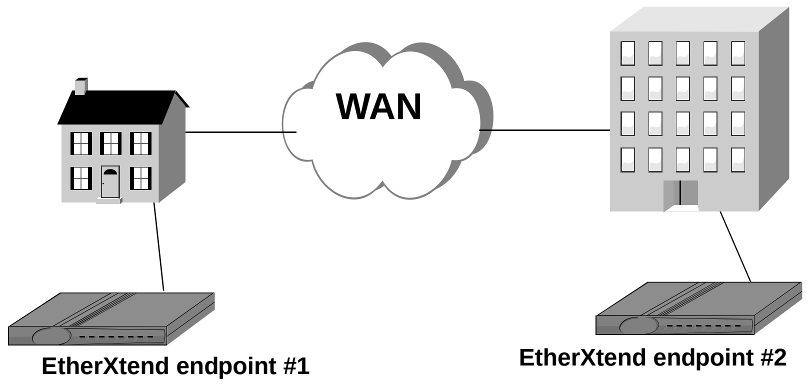
- Endpoint-to-endpoint CPEs on page 31
- Multiple endpoints to multiple cards on a chassis on page 31

- CO mode with subtended devices, page 32

Endpoint-to-endpoint CPEs

The basic scenario is a pair of EtherXtend devices acting as endpoints. This pairing is where an explicit endpoint is directly connected to another explicit endpoint. This topology is known as back-to-back mode. The following figure details this.

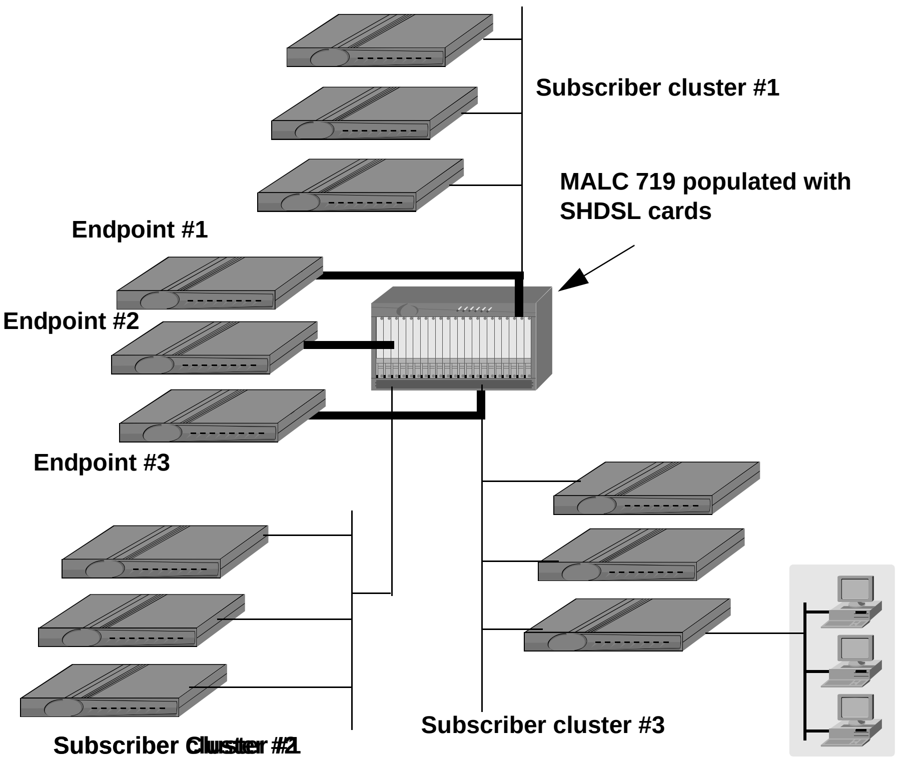
Figure 5: EtherXtend in back-to-back mode



Multiple endpoints to multiple cards on a chassis

This scenario enables the most subscribers. This scenario is multiple EtherXtend endpoints connecting to multiple SHDSL cards in a chassis. Each card has 24 ports and can connect to multiple endpoints on the subscriber side. The following figure shows three EtherXtend endpoints connecting to a containing three SHDSL cards. Each of those cards connects to three subscribers side EtherXtend endpoints, each with four ports.

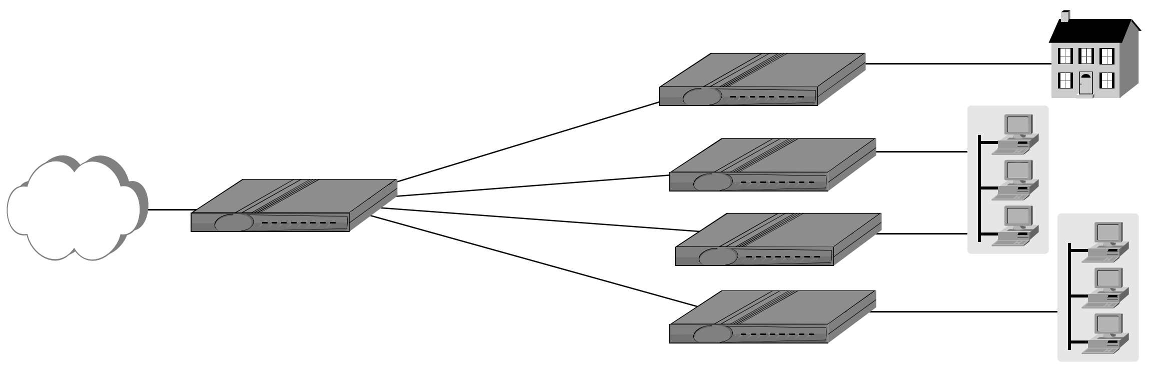
Figure 6: Multiple endpoints connecting to multiple SHDSL cards on a MALC



CO mode with subtended devices

In this scenario, one EtherXtend device functions in CO as the aggregation point to the network, while other EtherXtend devices are subtended in various locations and connected to CPEs.

Figure 7: EtherXtend in CO mode



Product specifications

The following are product specifications for the ETHX-SHDSL-4 and ETHX-SHDSL-8.

Specifications for the ETHX-SHDSL-4

Table 5: ETHX-SHDSL-4 specifications

Specification Type	Specification
Dimensions	1.75” (4.45 cm) High x 10” (21.6 cm) Wide x 7.5” (19.1 cm) Deep
Weight	3.35 lbs (1.52 kg)
Power	-48V DC and Universal AC power options available.
Interfaces	4 extended-rate SHDSL (2Base-TL) interfaces; 4 10/100 Ethernet interfaces.
Standards Support	ITU G.994.1 G.handshake; IEEE 802.3 Ethernet; IEEE 802.3ah Ethernet in the First Mile (2Base-TL); IEEE 802.3ah OAM, IEEE 802.1Q/p.
Protocol Support	Host-based routing for per-interface single IP address assignments; network-based routing for per-interface IP subnet address assignments; IP host and gateway support; RFC 1483/2684 Encapsulation; VLAN 802.1Q support.
Management	Serial terminal and Telnet for command line interface; inband IP via 10/100 Ethernet or WAN port.
Bandwidth/Distance	Data rates up to 5.7 Mbps symmetrical; distances up to 24,000 ft/7,320m; cross-talk cancellation within bonded groups.
Operating Requirements	Temperature: -40^0 F to 149^0 F (-40^0 C to 65^0 C); Non-operating temperature: -40^0 F to 185^0 F (-40^0 C to 85^0 C); humidity: 5% to 95%, non-condensing; altitude: -200 ft. to 16,500 ft. (-60m to 5,000m).

Specifications for the ETHX-SHDSL-8

Table 6: ETHX-SHDSL-8 specifications

Specification Type	Specification
Dimensions	1.75” (4.45 cm) High x 10” (21.6 cm) Wide x 7.5” (19.1 cm) Deep
Weight	3.35 lbs (1.52 kg)
Power	-48V DC and Universal AC power options available.
Interfaces	8 extended-rate SHDSL (2Base-TL) interfaces; 4 10/100 Ethernet interfaces.
Standards Support	ITU G.994.1 G.handshake; IEEE 802.3 Ethernet; IEEE 802.3ah Ethernet in the First Mile (2Base-TL); IEEE 802.3ah OAM, IEEE 802.1Q/p.
Protocol Support	Host-based routing for per-interface single IP address assignments; network-based routing for per-interface IP subnet address assignments; IP host and gateway support; VLAN 802.1Q support.
Management	ZMS via SNMPv2c for GUI and CORBA IDL machine interface; serial terminal and Telnet for command line interface; inband IP via 10/100 Ethernet or WAN port.
Operating Requirements	Temperature: -40^0 F to 149^0 F (-40^0 C to 65^0 C); Non-operating temperature: -40^0 F to 158^0 F (-40^0 C to 70^0 C); humidity: 5% to 95%, non-condensing; altitude: -200 ft. to 16,500 ft. (-60m to 5,000m).

4 INSTALLATION PREPARATION

This chapter describes how to prepare your site for the installation of the EtherXtend platform. It includes the following topics:

Tools you need

The required equipment listed in Table 7 should be available before beginning the installation of the EtherXtend system.

Table 7: Equipment required to install the EtherXtend system

Qty	Equipment	Details	Use
1	VT-100-compatible terminal or PC used as a VT-100 terminal emulator	Connected to the EtherXtend through RJ45 craft port.	Commission and configuration
1	Pliers		General installation
1	Cable prep tools	Pressfit and crimpers	Cable installation
-	Cables		System connections
2	#1 and #2 Phillips-head and 1/8-inch flat-blade screwdrivers	N/A	Locking and unlocking cards, front panels and chassis brackets
2	Antistatic wrist strap	N/A	Static electricity prevention

Selecting the system location

Ensure that the environment is free of dust and excessive moisture, not exposed to the elements or temperature extremes, and has sufficient ventilation.

Install the system in reasonable proximity to all equipment with which it will connect. Ensure that proper cable grades are used for all system and network connections. For best results, use the cables and connectors recommended in this document.

Environmental specifications

Table 8 describes the EtherXtend chassis environmental specifications and shows the EtherXtend dimensions.

Table 8: EtherXtend environmental specifications

Description	Specification
Weight	3.35 lbs. (1.52 kg) fully loaded
Operating temperature	0^0C to +40^0C (32^0F to +104^0F).
Storage temperature	0^0C to +40^0C (32^0F to +104^0F)
Operating relative humidity	5% to 95% noncondensing
Storage relative humidity	Up to 95% noncondensing
Altitude	Operating altitude: Up to 4,000 m (13,123 ft.)
Airflow	EtherXtend (working at front of unit): Left to right

Power requirements and specifications

–48V DC power sources to be connected to the EtherXtend system. The Return (+) terminals are common.

Table 9 describes the EtherXtend power specifications.

Table 9: EtherXtend power supply specifications

Description	Specification
Rated voltage	-41.75V to -60.0V DC
Maximum power consumption	EtherXtend: 4-port watts, 19.4 watts EtherXtend 8-port watts, 19.4 watts
DC-input cable	AWG 18 (5.27 mm^2) maximum

5 EtherXtend Installation

This chapter explains how to install the EtherXtend hardware. It includes the following sections:

- Unpacking the system, page 37
- Port and LED descriptions, page 38
- Connect the power supply, page 40
- Connect the WAN SHDSL lines, page 41
- Ground the device, page 44

Unpacking the system

Use the following procedure to unpack the EtherXtend system components from the shipping cartons.

- On system receipt, check the shipping cartons for physical damage.
- Unpack the shipping cartons, and check the contents for physical damage.
- If the equipment appears damaged, immediately contact the shipping company to file a claim.

The shipping company representative will give instructions on how to submit a claim, where to send the unit, and any special instructions that may be required.

If you need to return the equipment, pack the equipment in its original packing materials and send it by prepaid freight to the address given by the claims representative. If the original packing materials are unavailable, ship the equipment in a sturdy carton, wrapping it with shock-absorbing material.

Port and LED descriptions

This section provides descriptions of the ports on the rear panel of the EtherXtend as follows:

- *EtherXtend rear panel ports* on page 38
- *LED descriptions* on page 38
- *LED states for the EtherXtend* on page 39

EtherXtend rear panel ports

The following graphic shows where the ports are located.

Figure 8: Location of rear panel ports

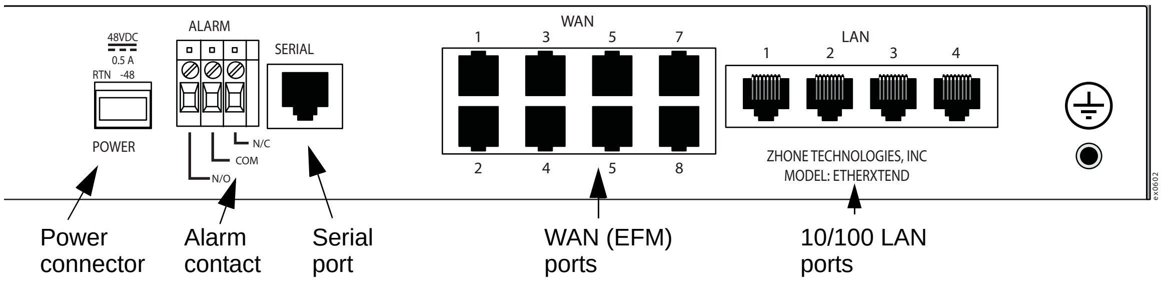


Table 10: Description of rear panel ports

Port	Description	Type	Speed/Protocol
Serial	Enables a serial modem connection for establishing out-of-band management sessions from outside of the network.	RJ-45	9600 Bps/RS-232
WAN (SHDSL)	Enables a SHDSL connection.	RJ-45	5,696 Kbps
10/100	Enables a Fast Ethernet connection. Four 10/100 ports.	RJ-45	10 Mbps/Ethernet 100 Mbps/Ethernet

LED descriptions

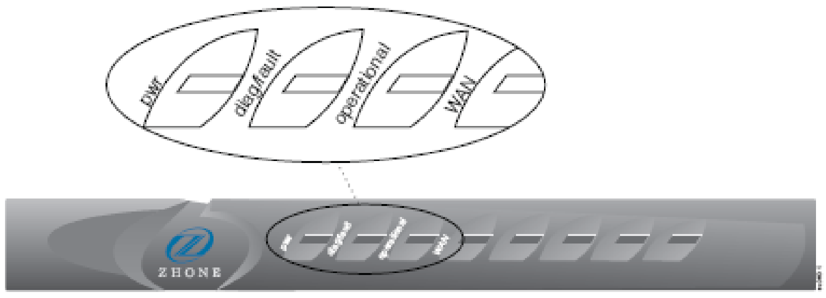
This section describes the EtherXtend LEDs.

The two types of LEDs found on the EtherXtend are:

- **Status LEDs** Located on the front of the device to show system-wide states
- **Port LEDs** Located in the ports on the back of the device to show the states that exist for a specific port, for example the status of a link connection.

The following table describes the both the system and port LEDs that appear on EtherXtend. The LED port lights are located on the port.

Figure 9: LEDs on the EtherXtend Device



LED states for the EtherXtend

The state of the LEDs show how the device is operating. The following table describes the LED states.

Table 11: LED states on the EtherXtend

LED	LED Color	Solid/Blinking	Meaning of the LED
Power	green	solid	Battery voltage is within tolerance.
Diagnostics	amber	blinking	Occurs during the Post process if any alarms are present.
Operational	green	blinking	Device is initializing.
WAN	green	solid	Indicates whether any activity occurs on the EFM ports or in Data mode. Training has occurred. Speed negotiation has occurred.
WAN (SHDSL) (Left) - Port State	green	solid	The port link state is up.
		off	None of the ports are in data mode.
		blinking	The port link state is down.
LAN (10/ 100) (left side)	green	solid	The port links to the network.
		off	The port does not link to the network.
		blinking	The port has activity occurring on it.

Table 11: LED states on the EtherXtend (Continued)

LED	LED Color	Solid/Blinking	Meaning of the LED
LAN (10/ 100) (right side)	green	off	The port operates in 10BASE-T mode where it transmits and receives packets at 10Mbps.
		on	The port operates in 100BASE-T mode where it transmits and received packets at 100Mbps.

Connect the power supply

Connect the power supply to the EtherXtend by plugging either the AC power supply or the DC power supply into the power adaptor port on the back of the device.The power adaptor port has a plus (+) and a minus (-). The plus side connects to the RTN side of the power outlet and the minus side connects to the -48 side of the power outlet. Verify that the power LED on the front of the device illuminates. After startup, Ethernet links remain disabled until at least one of the SHDSL connection has been established. Figure 10 shows the AC power supply.

Figure 10: Connect AC power

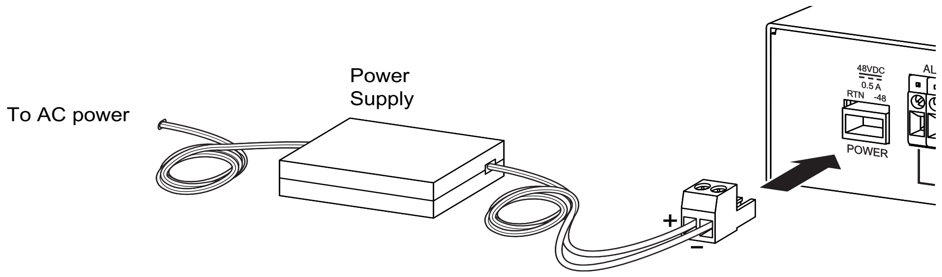
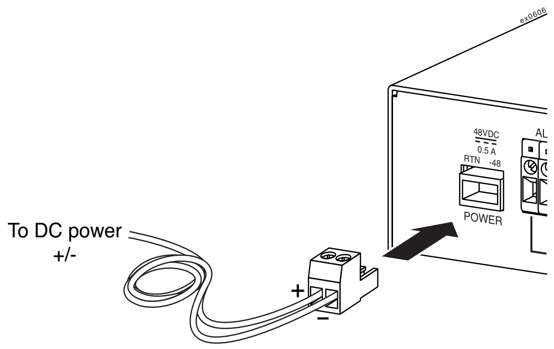


Figure 11 shows a DC power connection.

Figure 11: Connect DC power



Connect the WAN SHDSL lines

An important feature of the EtherXtend device is the loop bonding capability among all four SHDSL ports. However, both the provider and the subscriber units can function with a single SHDSL connection as follows:

- *Establish a loop bonded connection* on page 41
- *Connect the LAN Ethernet line* on page 41

Establish a loop bonded connection

Using up to eight SHDSL lines for one network connection (loop bonding) will net up to eight times the speed and data passing capability as a single SHDSL connection. Multiple SHDSL lines used for one connection provide backup for each other should one or more of the lines become disabled.

Establishing a loop bonded connection

1. Plug your SHDSL cables into the SHDSL RJ-45 ports (any combination of SHDSL 1, 2, 3, and 4) on the rear of the device. The order of the connection is not important.
2. Verify the connections. The SHDSL link LED for each connected port flashes green when the connection is established and operational.

Establishing a single line connection

1. Plug your SHDSL cable into one of the four SHDSL RJ-45 ports on the back of the device. Any of the four ports may be used.
2. Verify your connection. The SHDSL LED corresponding to the connected port flashes green when the connection is established and operational.

Connect the LAN Ethernet line

If an SHDSL connection has not yet been made, the Ethernet link remains disabled as indicated by no illumination of the LEDs until at least one of the four SHDSL lines are established.

The 10/100 Ethernet port auto-negotiates speed and duplex mode in accordance with the remote equipment to which it is connected.

- Half Duplex - Receive and transmit functions are mutually exclusive; data transmission occurs in only one direction at a time. Packet collisions are unusual.
- Full Duplex - Receive and transmit functions occur simultaneously, effectively doubling aggregate bandwidth and preventing packet collisions.

For the best connection results, the remote device should be set to autonegotiate speed and duplex mode as well. If the remote device cannot be configured to autonegotiate, speed may be hard set at either 10 Mbps or 100 Mbps. But duplex mode must be hard set to half duplex. A 10/100 Ethernet connection will not operate properly if the remote device is hard set to full duplex.

Hard setting the speed and mode of an Ethernet port

In order to manually provision the speed and duplex of the Ethernet port, the autonegstatus must be set to disabled and any of the following values entered in the mauType field:

Table 12: mauType field values

mauType	Setting
mauType = mau10basethd	10 Mbps half-duplex
mauType = mau10basetfd	10 Mbps full-duplex
mauType = mau100basetxhd	100 Mbps half-duplex
mauType = mau100basetxfd	100 Mbps full-duplex

To change the speed and mode of the Ethernet port, enter the following commands:

1 To view the Ethernet port interfaces, enter **list ether**:

```
zSH> list ether
ether  1-1-1-0/eth
ether  1-1-2-0/eth
ether  1-1-3-0/eth
ether  1-1-4-0/eth
4 entries found.
```

2 To view the Ethernet port interface parameter defaults, enter **get ether** *interface/type*:

```
zSH> get ether 1-1-1-0/eth
ether  1-1-1-0/eth
autonegstatus: ---->  {enabled}
mauType: ---------->  {mau100basetxfd}
restart: ---------->  {norestart}
ifType: ----------->  {mau100basetxfd}
autonegcap: ------->  {b10baseT+b10baseTFD+b100baseTX+b100baseTXFD}
remotefault: ------>  {noerror}
clksrc: ----------->  {automatic}
pauseFlowControl: ->  {disabled}
```

3 To view the ether port interface parameter variables, enter **show ether**:

```
zSH> show ether
autonegstatus:---->   enabled  disabled
mauType:---------->   mauother  mau10baset  mau10basethd  mau10basetfd  mau100basetxhd
  mau100basetxfd  mau1000baselxhd  mau1000baselxfd  mau1000basesxhd  mau1000basesxfd
  mau1000basethd  mau1000basetfd
restart:---------->   restart  norestart
ifType:----------->   mauother  mau10baset  mau10basethd  mau10basetfd  mau100basetxhd
  mau100basetxfd  mau1000baselxhd  mau1000baselxfd  mau1000basesxhd  mau1000basesxfd
  mau1000basethd  mau1000basetfd
autonegcap:------->   bOther+b10baseT+b10baseTFD+b100baseT4+b100baseTX+b100baseTXFD+b1
  00baseT2+b100baseT2FD+b1000baseX+b1000baseXFD+b1000baseT+b1000baseTFD+b100baseX+b1ba
  seXFD+b100baseT+b100baseTFD
remotefault:------>   noerror  offline  linkfailure  autonegerror
clksrc:----------->   unused  automatic  master  slave
pauseFlowControl:->   disabled  asymmetricTx  asymmetricRx  symmetric
```

4 To change the speed/mode of the Ethernet port, disable auto negotiate and change the mauType as follows using **update ether** *interface/type*:

a To set Ethernet port 1 to 10Mbps half-duplex enter:

```
zSH> update ether 1-1-1-0/eth
 autonegstatus:---->{enabled}: disabled
  mauType:---------->{mau100basetxfd}: mau10basethd
Save changes? [s]ave, [c]hange or [q]uit: s
```

b To set Ethernet port 1 to 10Mbps full-duplex enter:

```
zSH> update ether 1-1-1-0/eth
 autonegstatus:---->{enabled}: disabled
    mauType:---------->{mau100basetxfd}: mau10basetfd
Save changes? [s]ave, [c]hange or [q]uit: s
```

c To set Ethernet port 1 to 100Mbps half-duplex enter:

```
zSH> update ether 1-1-1-0/eth
autonegstatus:---->{enabled}: disabled
mauType:---------->{mau100basetxfd}: mau100basetxhd
Save changes? [s]ave, [c]hange or [q]uit: s
```

d To set Ethernet port 1 to 100Mbps full-duplex enter:

```
zSH> update ether 1-1-1-0/eth
autonegstatus:---->{enabled}: disabled
    mauType:---------->{mau100basetxfd}: 100basetxfd
Save changes? [s]ave, [c]hange or [q]uit: s
```

5 To set the Ethernet port 1 back to auto-negotiate enter:

```
zSH> update ether 1-1-1-0/eth
    autonegstatus:---->{disabled}: enabled
    mauType:---------->{mau100basetxfd}: mau100basetxfd
Save changes? [s]ave, [c]hange or [q]uit: s
```

Ground the device

When the AC plug is used, but not grounded it is recommended to ground the device using minimum 16-guage wire to a building or earth ground. The ground screw is located on the far-right on the back of the device.

6 BASIC CONFIGURATION

This chapter covers basic configuration of the EtherXtend:

Device management

This section describes how the EtherXtend can be managed either through the the serial interface or the WAN Ethernet ports.

- *EtherXtend local management interface* on page 46
- *EtherXtend remote management interfaces* on page 46

EtherXtend devices provide interfaces for four or eight WAN SHDSL ports, four LAN 10/100 Ethernet ports, and a serial port for local management.

After connecting the MALC Ethernet ports on a SHDSL card and the Ethernet WAN ports on the EtherXtend, the default Autoconfig automatically configures the interface as a N2N bond group on VLAN 7 as a DCHP client.

Table 13 lists the EtherXtend system default settings.

Table 13: EtherXtend default system settings

Parameter	Default setting
Mode	CPE Mode
IP Address	DHCP enabled on bond group port.
SHDSL Speed	5,696 Adaptive

Table 13: EtherXtend default system settings (Continued)

Parameter	Default setting
Ethernet Interface	Autosensing Enabled Autonegotiating Enabled
Bonding Mode	N2N
Login	admin
Password	zhone
DHCP Client (WAN)	Enabled

EtherXtend local management interface

The EtherXtend unit provides an out-of-band RS232 D serial (craft) interface to manage the unit locally. To access the serial port, configure the terminal interface software with the following settings:

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

You must perform the initial configuration of the system using the serial (craft) interface. After completing the initial configuration, you can manage the EtherXtend device remotely through Telent over the network through the Ethernet management interface or over the management PVC.

Note: The EtherXtend supports a maximum of two concurrent telnet sessions and one serial session.

EtherXtend remote management interfaces

This section describes how the EtherXtend can be managed remotely from the MALC using the default management interface and how to change the management interface to manage the EtherXtend from a different IP address:

EtherXtend uses the *shelf-slot-port-subport/type* syntax to identify system interfaces. For the EtherXtend, the convention is always 1 for the *shelf* and 1 for the *slot* value. The *subport* is always 0. The *type* is based on the Internet Assigned Numbers Authority interface type (IANAiftype) definitions.

By default, Autoconfig assigns a bond group number to the N2N bond group which may vary depending on which port receives the link first. The management interface is built on the default N2N bond group, 1-1-1-*bondgroupnumber*/n2nbond, as a DHCP client using VLAN 7.

Managing EtherXtend using the default management interface

The MALC that connects to the EtherXtend can be configured as a local DCHP server or as a client with a bridge to an external DHCP server.

1 Configure Telnet access from the MALC to the EtherXtend by creating a new dhcp-server subnet 1 on the MALC, with the following parameters:

```
zSH> get dhcp-server-subnet  1
dhcp-server-subnet  1
network: ---------------->  {10.250.1.0}
netmask: ---------------->  {255.255.255.0}
domain: ----------------->  {0}
range1-start: ---------->  {10.250.1.1}
range1-end: ------------>  {10.250.1.250}
range2-start: ---------->  {0.0.0.0}
range2-end: ------------>  {0.0.0.0}
range3-start: ---------->  {0.0.0.0}
range3-end: ------------>  {0.0.0.0}
range4-start: ---------->  {0.0.0.0}
range4-end: ------------>  {0.0.0.0}
default-lease-time: ---->  {-1}
min-lease-time: -------->  {-1}
max-lease-time: -------->  {-1}
boot-server: ----------->  {0.0.0.0}
bootfile: -------------->  {}
default-router: -------->  {10.250.1.254}
primary-name-server: --->  {0.0.0.0}
secondary-name-server: ->  {0.0.0.0}
domain-name: ----------->  {}
subnetgroup: ----------->  {1}
stickyaddr: ------------>  {enable}
external-server: ------->  {0.0.0.0}
```

2 Create the management interface on the MALC.

```
zSH> interface add float management 10.250.1.254 255.255.255.0
```

3 Enter **host add** on the MALC to configure a host connection between the EtherXtend and the bond group associated with the management interface

```
zHS> host add 1-10-203-0/n2nbond vlan 7 dynamic 1 1
```

You can now Telnet to the MALC and then Telnet to the EtherXtend device.

Refer to the MALC documentation for detailed procedures on DHCP configuration

Managing EtherXtend using a non-default management interface

To manage the EtherXtend through an interface different than the default AutoConfig address, delete the default AutoConfigIP address and then add the desired interface. If the new IP address is not compatible with the address of the management PC, the connection to the device will be lost. To restore the connection, change the address of the management PC to be compatible with the device address.

The following example configures the IP address for the system:

1 Delete the AutoConfig address.

```
zSH> delete ip-interface-record AutoConfig/ip
```

2 Add the desired interface to a bond port group.

```
zSH> interface add 1-1-4-0/n2nbond 172.24.200.133/24
Created ip-interface-record ethernet1/ip
```

Note: The default interface is reset if a **set2default** is performed without the restore option.

Verifying the interface

Enter **interface show** to verify that the Ethernet interface was configured correctly on the EtherXtend:

```
zSH> interface show
1 interface
Interface      Status  Rd/Address            Media/Dest Address      IfName
---------------------------------------------------------------------------
1/1/4/0/ip     UP      1 172.24.200.133/24       00:01:47:07:ef:ee   AutoConfig
---------------------------------------------------------------------------
```

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

To verify that the routes were added, tenter **route show**:

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

To verify connectivity to the default gateway, enter **ping**:

```
zSH> ping 192.168.8.1
PING 192.168.10.1: 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.

EtherXtend auto-provisioning

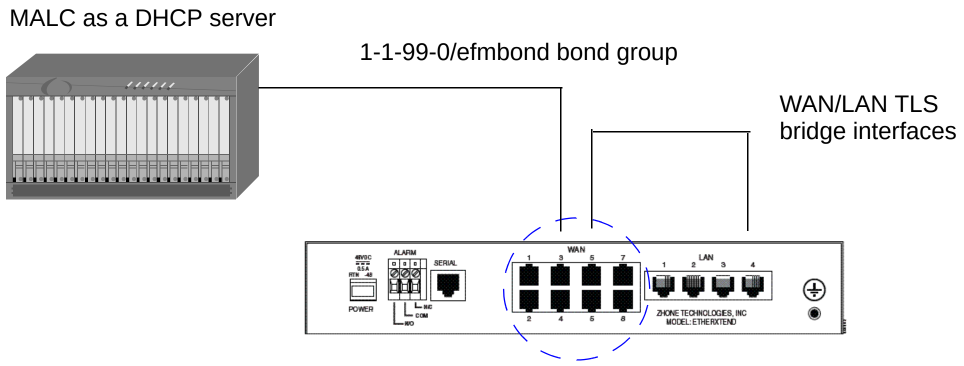
The auto-provisioning feature of the EtherXtend provides a factory default configuration of EtherXtend interfaces. Auto-provisioning runs during the initial boot of the EtherXtend or after entering a **set2default** command.

Note: In order not to run auto-provisioning and to use your own customized configuration settings, you must either have a *default* file or a *restore* file located in the *onreboot* directory. To create a *default* file see Customize the EtherXtend default settings, page 53. To create a restore file see

During the initial EtherXtend boot, auto-provisioning automatically:

- Creates an efmbond group, 1-1-99-0/efmbond with all SHDSL interfaces as members.

- Adds a DHCP client interface using VLAN 7 on the bond group for device management over the WAN.
- Creates a TLS bridge for tagged VLAN traffic (VLAN 0) and a TLS bridge for untagged traffic on each of the four LAN Ethernet interfaces and the SHDSL efmbond group.



EtherXtend as a DHCP client with IP address

Auto-provisioning automatically runs when an EtherXtend is upgraded to a version of software that runs auto-provisioning from a version of the software that did not run auto-provisioning. When auto-provisioning runs after an upgrade, some of the existing configuration will be overwritten, specifically bridges and the IP interfaces.

Auto-configuration runs just once during the initial boot of the EtherXtend and runs once after a **set2default**. When upgrading the EtherXtend software from a verison of the software that supports auto-provisioning, auto-provisioning does not run again except after using the **set2default** command.

For a description of how to create user-defined defaults and bypass EtherXtend auto-provisioning to factory defaults, see *Customize the EtherXtend default settings* on page 53.

To verify the version of the software running on an EtherXtend enter **swversion**:

```
zSH> swversion
Zhone ethXShdsl software version ETHX 1.14.2.119
```

To view the auto-provisioned configuration, enter **bridge show**:

```
zSH> bridge show
Type VLAN       Bridge                         St  Table Data
-------------------------------------------------------------------------------------
tls           7 1-1-99-0-efmbond/bridge        UP
tls Tagged      1-1-99-0-efmbond-0/bridge      UP
tls           7 1-1-1-0-eth/bridge             DWN
tls Tagged      1-1-1-0-eth-0/bridge           DWN
tls           7 1-1-4-0-eth/bridge             DWN
tls Tagged      1-1-4-0-eth-0/bridge           DWN
tls            7 1-1-3-0-eth/bridge            DWN
tls Tagged       1-1-3-0-eth-0/bridge          DWN
tls            7 1-1-2-0-eth/bridge            DWN
tls Tagged       1-1-2-0-eth-0/bridge          DWN
zSH>
```

To view the bond group and the members of the bond group, enter **bond show group** *interface/type*:

```
zSH> bond show group 1-1-99-0/efmbond
                  Bond Groups
  Slot   GrpId      Name            Type         State
   1       99       1-1-99-0        efmbond       ACT
                  Group Members
  Slot    Port      Name            Type         State
   1        3       1-1-3-0         shdsl         ACT
   1        2       1-1-2-0         shdsl         ACT
   1        1       1-1-1-0         shdsl         ACT
   1        8       1-1-8-0         shdsl         ACT
   1        7       1-1-7-0         shdsl         ACT
   1        6       1-1-6-0         shdsl         ACT
   1        5       1-1-5-0         shdsl         ACT
   1        4       1-1-4-0         shdsl         ACT
```

To verify the bond group interface enter **interface show**:

```
zSH> interface show
1 interface
Interface      Status  Rd/Address           Media/Dest Address      IfName
--------------------------------------------------------------------------------
1/1/99/0/ip    UP      1 0.0.0.0/0          00:01:47:f6:48:27       AutoConfig
--------------------------------------------------------------------------------
```

AutoConfig under the IfName column indicates that the IP management interface was created using VLAN 7.

To verify these auto-provisioning settings and verify that the Etherxtend was configured as a DHCP client with a VLAN 7 on the EFM bond group IP interface, enter **get ip-interface-record** *interface:*

```
zSH> get ip-interface-record 1/1/99/0/ip
ip-interface-record  1/1/99/0/ip
vpi: ------------------------->  {0}
vci: ------------------------->  {0}
rdindex: --------------------->  {1}
dhcp: ------------------------>  {client}
addr: ------------------------>  {0.0.0.0}
netmask: --------------------->  {0.0.0.0}
bcastaddr: ------------------->  {255.255.255.255}
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: --------------->  {dhcpclient}
dhcpserverenable: ------------>  {false}
subnetgroup: ----------------->  {0}
unnumberedindex: ------------->  {0}
mcastcontrollist: ------------>  {}
vlanid: ---------------------->  {7}
maxVideoStreams: ------------->  {0}
tosOption: ------------------->  {disable}
tosCOS: ---------------------->  {0}
vlanCOS: --------------------->  {0}
s-tagTPID: ------------------->  {0x8100}
s-tagId: --------------------->  {0}
s-tagIdCOS: ------------------>  {0}
```

Customize the EtherXtend default settings

When installing EtherXtend software for the first time with auto-provisioning, auto-provisioning creates default bridges and configures the IP interface. (For a description of EtherXtend auto-provisioning, see *EtherXtend auto-provisioning* on page 49).

✓ **Note:** When upgrading an EtherXtend from versions of the software without auto-provisioning to a version of the software with auto-provisioning, auto-provisioning will overwrite existing bridges and the IP interface with default settings.

After running auto-provisioning and changing the configuration of the EtherXtend, you can save that configuration to a *default* file to restore your configuration later when needed. This *default* file will override factory default settings created by auto-provisioning. The system will use this file after a **set2default** command when there is no configuration currently on the system. You can also use this *default* file created on one system as a common base configuration for other systems by uploading the file to a server and downloading the *default* file to the other systems.

Using customized EtherXtend default settings

Follow these steps to create a new *default* file that the EtherXtend recognizes every time the device is set to defaults. You can also use the *default* file to configure multiple EtherXtends.

1. Configure the EtherXtend with your unique settings for bridging, hosts, VLANS, IP addresses, etc. as needed.
2. Verify that you are in the *card1* directory before creating the *default* file:

```
zSH> cd /card1
zSH> pwd
/card1
```

3. Create the *default* file that stores your configuration and place that file in the *onreboot* directory with **dump file** *directory/filename*:

```
zSH> dump file /onreboot/default
```

4. Place the *default* file on multiple EtherXtends, if needed, by uploading the *default* file to a TFTP server, then downloading that *default* file from the TFTP server onto each EtherXtend:

   a. Upload the *default* file with **file upload** [*TFTP server ipAddr*] [*source filename*] [*destination filename*]:

```
zSH> file upload 172.16.100.123 /onreboot/default /
default
```

   The source filename can include the directory path to the file.

b Download the *default* file into the onreboot directory of the EtherXtend with **file download** [*TFTP server ipAddr*] [*source filename*] [*destination filename*]:

Note: The EtherXtend will not recognize the destination file as the default configuration file unless the file is named *default.*

```
zSH> file download 172.16.100.123 /default /
onereboot/default
```

The destination directory and filename is always */onreboot/default*.

5 Reboot the EtherXtend and let it completely finish the reboot process.

Note: The EtherXtend does not use auto-provisioning or Zhone's factory defaults whenever you place a file named *default* in the *onreboot* directory.

```
zSH> reboot
Rebooting...
```

6 Set your defaults after the EtherXtend is completely booted and respond to several prompts after you log in using the login: *admin*, and password: *zhone* by entering **set2default**:

```
login: admin
password:
NOV 27 20:01:42: alert  : 1/1/1027: clitask0:
CliUserConnect(): l=1695: tCliInit0: User admin logged
in on slot 1
zSH> set2default
No restore file (/card1/onreboot/restore) found.
Setting to default will result in an empty database.
Continue? (yes or no) [no]: yes
Ok to reset to default (system will reboot) ? [yes] or
[no]: yes
Do you want to exit from this request? (yes or no)
[yes] no
Are you sure? (yes or no) [no] yes
```

7 At this point the EtherXtend should be rebooting. To verify that the *default* file is loaded, notice that the EtherXtend will actually reboot twice. The first time it loads the default file, the second time it boots with the new configuration.

During the first boot, the load of the default will list various configurations being loaded. This is an example of what the configurations may look like:

```
restore partial file /card1/onreboot/default
if-translate (28)
if-next-index (1)
ether (4)
```

```
alarm-config (1)
bridge-interface-record (4)
dhcp-client-lease-resource (3)
dhcp-client-resource (1)
dhcp-server-options (1)
dsl-alarm (4)
dsl-config (4)
efm-port (4)
ether-oam (1)
info-reconcile (9)
ip-interface-record (2)
ipsla-cos-act (8)
ipsla-cos-map (64)
ipsla-global (1)
ntp-client-config (1)
pat-bind (1)
pme-profile (4)
rip-global-config (1)
rip-if-config (2)
system (1)
user-profile (2)
if-stack (22)
static-route (1)
card-profile (1)
(restore completed)
```

The first and last lines of this example should always be displayed on a successful load of the *default* file. All the other lines depend on what is in the *default* file and may or may not be present.

Shortly, after the (restore completed) line, the second reboot is started.

Deleting your customized configuration and returning to factory defaults

If you need to delete your customized configuration and return to EtherXtend factory defaults:

1 Remove your custom configuration by removing the *default* file:

```
zSH> del /onreboot/default
```

2 Reset the EtherXtend to factory settings and run auto-provisioning:

```
zSH> set2default
```

Entering **set2default** erases the configuration in the EtherXtend's database and restores factory default settings.

System settings

This section describes the following system settings:

- System security
- Radius support
- System logging

System security

There are several methods to guard against unauthorized access to your EtherXtend, such as changing the default user password. You can also set up SNMP access lists to restrict access to your system. See Creating community access lists on page 86 for more information about setting SNMP access lists.

System login

Follow this example to login to a 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 the *Zhone CLI Reference Guide* for information on the **timeout** command.

Changing the default user password

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

```
jsmith> changepass
Current Password: enter current password
New Password : enter new password
Confirm New Password : confirm new password
User record updated.
Password change successful.
```

Note: Passwords are case sensitive.

Radius support

The EtherXtend supports local and RADIUS (Remote Authentication Dial In User Service) access authentication. The EtherXtend 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 14 shows the mapping of service-type to EtherXtend permissions.

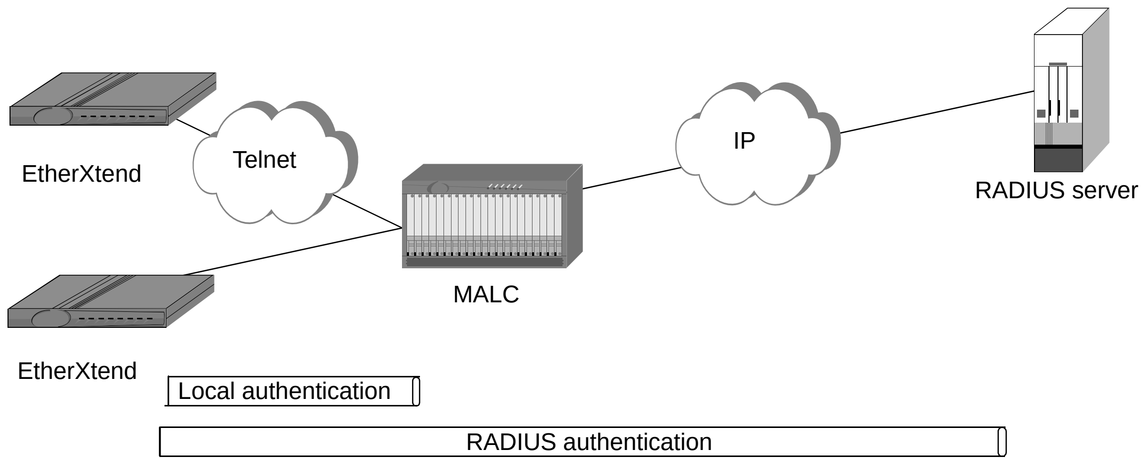
Table 14: Service type mapping to EtherXtend permissions

Service-Type Attribute	EtherXtend 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 EtherXtend with RADIUS authentication, the EtherXtend passes RADIUS information securely to the RADIUS server. The RADIUS server then authenticates the user and either allows or denies access to the EtherXtend. If access is denied and the local authentication option is also configured, the EtherXtend 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 EtherXtend 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 12: EtherXtend RADIUS authentication



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

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

Configuring RADIUS support

The EtherXtend 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 EtherXtend attempts authentication 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 EtherXtend 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 EtherXtend 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 EtherXtend. 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*. The IP address is leased from a DHCP server so 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 additional radius-client profiles for each additional RADIUS server to be assigned to this EtherXtend. The index number is incremented (for example 1/2 for the second RADIUS server in group 1) to specify the sequence in the profile group.

```
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.
```

4 In the system profile on the EtherXtend, 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 EtherXtend first attempts authentication using the server specified in radius-client 1/1. If this authentication fails, the EtherXtend attempts authentication using radius-client 1/2 server. If this authentication also fails, the EtherXtend then attempts authentication based on the authentication mode setting in the system profile. This example uses **radiusthenlocal**.

Caution: If the *radius* authentication mode is used, local authentication is disabled so the EtherXtend may become inaccessible if IP connectivity to the RADIUS server is lost or other changes prevent the EtherXtend from receiving RADIUS authentication.

```
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: -------------->  {EtherXtend1}:
syslocation: ---------->  {Oakland}:
enableauthtraps: ------>  {disabled}:
setserialno: ---------->  {0}:
zmsexists: ------------>  {true}:
zmsconnectionstatus: -->  {inactive}:
zmsipaddress: --------->  {172.16.49.76}:
configsyncexists: ----->  {false}:
configsyncoverflow: --->  {false}:
configsyncpriority: --->  {high}:
configsyncaction: ----->  {noaction}:
configsyncfilename: --->  {172.16.88.14_4_1178142210378}:
configsyncstatus: ----->  {synccomplete}:
configsyncuser: ------->  {zmsftp}:
configsyncpasswd: ----->  {** private **}:  ** read-only **
numshelves: ----------->  {1}:
shelvesarray: --------->  {}:
numcards: ------------->  {3}:
ipaddress: ------------>  {172.16.88.14}:
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}: radiusthenlocal
radiusauthindex: ------>  {0}: 1
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
zSH>
```

After completing the RADIUS configuration, the EtherXtend 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 EtherXtend using the default Zhone EtherXtend system name will appear as basicuser@Zhone EtherXtend. The system name is configured using the sysname parameter in the System 0 profile.

System logging

System logs can be enabled to record session activity and user access.

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, enter:

```
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 enter:

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

This command setting persists across system reboots.

Device interface

Although the EtherXtend does not have cards, the EtherXtend device settings are stored in the card-profile parameter. Update the card-profile to modify the device settings. The device type number for the EtherXtend is 7101.

```
zSH> get card-profile 1/1/7101

card-profile  1/1/7101
sw-file-name: ----------->  {ethxshdsl.bin}
admin-status: ----------->  {operational}
upgrade-sw-file-name: --->  {}
upgrade-vers: ----------->  {}
admin-status-enable: ---->  {enable}
sw-upgrade-admin: ------->  {reloadcurrrev}
sw-enable: -------------->  {true}
sw-upgrade-enable: ------>  {false}
card-group-id: ---------->  {1}
hold-active: ------------>  {false}
weight: ----------------->  {nopreference}
card-line-type: --------->  {unknowntype}
card-atm-configuration: ->  {notapplicable}
card-line-voltage: ------>  {not-used}
maxvpi-maxvci: ---------->  {notapplicable}
card-init-string: ------->  {}
wetting-current: -------->  {disabled}
```

LAN interfaces to CPEs

The EtherXtend provides 4 Ethernet LAN ports for 10/100 Ethernet connections to CPEs or subtended devices.

Use the following command to display the available Ethernet LAN interfaces.

```
zSH> list ether
ether  1-1-1-0/eth
ether  1-1-2-0/eth
ether  1-1-3-0/eth
ether  1-1-4-0/eth
4 entries found.
```

A profile is available for each Ethernet LAN interface to configure Ethernet parameters. Use the following command to configure the Ethernet LAN port settings. This example changes the pauseFlowControl setting to symmetric.

```
zSH> update ether 1-1-1-0/eth

ether  1-1-1-0/eth
Please provide the following: [q]uit.
autonegstatus: ---->  {enabled}
mauType: ---------->  {mau100basetxfd}
restart: ---------->  {norestart}
ifType: ----------->  {mau100basetxfd}
autonegcap: ------->  {b10baseT+b10baseTFD+b100baseTX+b100baseTXFD}
remotefault: ------>  {noerror}
clksrc: ----------->  {automatic}
pauseFlowControl: ->  {disabled} symmetric
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

Moving PC cables on Ethernet ports

If a PC cable is moved from one Ethernet port to another on the EtherXtend, a **bridge flush** command may be given from the EtherXtend CLI to force re-learning the PC's MAC address on the new Ethernet port. If the **bridge flush** command is not given, the MAC address timeout is 3600 seconds.

Configure an interface on an Ethernet port

Configuring IP interfaces involves creating an **ip-interface-record** on the LAN/WAN interface.

This profile specifies the basic IP parameters of the LAN interface. These include the IP address and netmask, and the services enabled on the interface. Each **ip-interface-record** is associated with a specific physical interface.

To create an IP interface, you need to know the logical address of the physical interface over which IP will run.

Tip: If you use the address format (with slashes instead of dashes) when creating the IP interface, the system will recognize the physical address and automatically bind the Ethernet line group to the new IP interface over the Ethernet port.

Table 15: Interface Parameters

Parameter	Description
addr	The IP address of the EtherXtend device in dotted-decimal format.

Table 15: Interface Parameters (Continued)

Parameter	Description
netmask	The subnet mask associated with the IP interface. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.
bcastaddr	The IP broadcast address used for sending datagrams on the (logical) interface associated with the IP interface. The broadcast address is determined by the IP address and the netmask. It should always be set to an IP address that is the network address of the interface with all ones in the host portion of the address.
mru	The size, in octets, of the largest packet that can be received on the IP interface. For interfaces used for network datagrams, this is the size of the largest network datagram that can be received on the interface.

```
zSH> new ip-interface-record 1/1/4/0/ip
Please provide the following: [q]uit.
vpi: --------------->  {0}:
vci: --------------->  {0}:
rdindex: ----------->  {1}:
dhcp: -------------->  {none}:  ** read-only **
addr: -------------->  {0.0.0.0}: 192.168.88.200
netmask: ----------->  {0.0.0.0}: 255.255.255.0
bcastaddr: --------->  {0.0.0.0}: 192.168.88.255
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}:
ingressfilterid: --->  {0}:
egressfilterid: ---->  {0}:
ipaddrdynamic: ----->  {static}:
dhcpserverenable: -->  {false}:
subnetgroup: ------->  {0}
unnumberedindex: --->  {0}
mcastcontrollist: -->  {}:
vlanid: ------------>  {0}:
maxVideoStreams: --->  {0}:
```

```
tosOption: --------->  {disable}:
tosCOS: ------------>  {0}:
vlanCOS: ----------->  {0}:
s-tagTPID: --------->  {0x8100}:
s-tagId: ----------->  {0}:
s-tagIdCOS: -------->  {0}:
....................
Save new record? [s]ave, [c]hange or [q]uit: s

This IP Interface has been automatically bound to
1-1-4-0-eth
New record saved.
```

To verify that the Ethernet interface has been set up, enter **interface show**:

```
zSH> interface show
1 interface
Interface      Status  Rd/Address            Media/Dest Address      IfName
-------------------------------------------------------------------------------
1/1/4/0/ip     UP      1 172.24.200.133/24 00:01:47:f6:48:1c       1-1-4-0-eth
-------------------------------------------------------------------------------
```

IP on a bridge

IP on a bridge allows you to put an IP address on a bridged VLAN. This allows VLANs to be used to manage multiple EtherXtends or other devices. One IP on a bridge can be created on a EtherXtend. The following example provides a typical example of how you would configure IP on a bridge.

Creating the IP on a bridge interface

Create an IP on a bridge interface using the IP address of 10.11.12.13/24, and a logical port interface 6 with a VLAN 200

✓ **Note:** The logical port interface for IP on a bridge *must* be *1-1-6-0/ipobridge* for correct transmission of IP packets.

1 Enter **interface add** *interface/type* with the type as *ipobridge*:

```
zSH> interface add 1-1-6-0/ipobridge vlan 200 10.11.12.13/24
Created ip-interface-record ipobridge-200/ip.
```

This command creates the new IP interface as well as a new bridge. The bridge created will be a Transparent LAN Service (TLS) bridge.

2 Enter **interface show** to verify the IP interface:

```
zSH> interface show
2 interfaces
Interface      Status  Rd/Address           Media/Dest Address      IfName
--------------------------------------------------------------------------------
1/1/6/0/ip     UP      1 10.11.12.13/24     00:01:47:f6:48:25       ipobridge-200
1/1/99/0/ip    UP      1 0.0.0.0/0          00:01:47:f6:48:27       AutoConfig
--------------------------------------------------------------------------------
```

Enter **bridge show** to verify the ipobridge:

```
zSH> bridge show
Type VLAN        Bridge                        St  Table Data
---------------------------------------------------------------------------------
tls            7 1-1-99-0-efmbond/bridge       UP
tls Tagged       1-1-99-0-efmbond-0/bridge     UP
tls            7 1-1-1-0-eth/bridge            DWN
tls Tagged       1-1-1-0-eth-0/bridge          DWN
tls            7 1-1-4-0-eth/bridge            DWN
tls Tagged       1-1-4-0-eth-0/bridge          DWN
tls            7 1-1-3-0-eth/bridge            DWN
tls Tagged       1-1-3-0-eth-0/bridge          DWN
tls            7 1-1-2-0-eth/bridge            DWN
tls Tagged       1-1-2-0-eth-0/bridge          DWN
tls Tagged 200   ipobridge-200/bridge          UP
```

3 Create another bridge on an uplink port to manage traffic going to the uplink connection with **bridge add**:

```
zSH> bridge add 1-1-99-0/efmbond tls vlan 200 tagged
Adding bridge on 1-1-99-0/efmbond
Created bridge-interface-record 1-1-99-0-efmbond-200/bridge
```

The uplink connection is now reachable from the upstream, and IP 10.11.12.13/24 can reach other upstream devices on the same VLAN.

4 Enter **bridge show** to verify the IP on a bridge and the upstream connection bridge on VLAN 200:

```
zSH> bridge show
Type VLAN       Bridge                        St  Table Data
---------------------------------------------------------------------------------
tls           7 1-1-99-0-efmbond/bridge       UP
tls Tagged      1-1-99-0-efmbond-0/bridge     UP
tls           7 1-1-1-0-eth/bridge            DWN
tls Tagged      1-1-1-0-eth-0/bridge          DWN
tls           7 1-1-4-0-eth/bridge            DWN
tls Tagged      1-1-4-0-eth-0/bridge          DWN
tls           7 1-1-3-0-eth/bridge            DWN
tls Tagged      1-1-3-0-eth-0/bridge          DWN
tls           7 1-1-2-0-eth/bridge            DWN
tls Tagged      1-1-2-0-eth-0/bridge          DWN
tls Tagged 200  ipobridge-200/bridge          UP
tls Tagged 200  1-1-99-0-efmbond-200/bridge   UP
```

Follow the same steps to create an IP on a bridge and bridges for downstream devices.

The IP on a bridge feature does not support SNMP.

Deleting IP on a bridge and the upstream connection bridge

1 Delete the IP on a bridge interface when necessary:

```
zSH> interface delete 1/1/6/0/ipobridge vlan 200
Delete complete
```

Enter **interface show** to verify the ipobridge interface is deleted.

```
zSH> interface show
1 interface
Interface      Status  Rd/Address            Media/Dest Address      IfName
--------------------------------------------------------------------------------
1/1/99/0/ip    UP      1 0.0.0.0/0           00:01:47:f6:48:27       AutoConfig
--------------------------------------------------------------------------------
```

2 Delete the IP on a bridge and the upstream bridge connection when necessary.

```
zSH> bridge delete ipobridge-200/bridge vlan 200
ipobridge-200/bridge Delete complete
zSH> bridge delete 1-1-99-0/efmbond vlan 200
1-1-99-0/efmbond Delete complete
```

3 Verify that the bridges are deleted with **bridge show**:

```
zSH> bridge show
Type VLAN        Bridge                        St  Table Data
--------------------------------------------------------------------------------
tls            7 1-1-99-0-efmbond/bridge       UP
tls Tagged       1-1-99-0-efmbond-0/bridge     UP
tls            7 1-1-1-0-eth/bridge            DWN
tls Tagged       1-1-1-0-eth-0/bridge          DWN
tls            7 1-1-4-0-eth/bridge            DWN
tls Tagged       1-1-4-0-eth-0/bridge          DWN
tls            7 1-1-3-0-eth/bridge            DWN
tls Tagged       1-1-3-0-eth-0/bridge          DWN
tls            7 1-1-2-0-eth/bridge            DWN
tls Tagged       1-1-2-0-eth-0/bridge          DWN
```

7

ETHERXTEND SHDSL WAN INTERFACES

This chapter describes the WAN SHDSL interfaces on the EtherXtend.

WAN SHDSL interfaces

The EtherXtend device can have either 4 or 8 SHDSL WAN interfaces for use as individual interfaces or as members of a bond group. The EtherXtend supports 2-wire SHDSL cards.

The EtherXtend uses the *shelf-slot-port-subport*/*type* syntax to identify system interfaces. The EtherXtend is always **1** for the *shelf*, ***1*** for the *slot* values and **0** for the *subport* value. The *type* is based on the Internet Assigned Numbers Authority interface type (IANAiftype) definitions.

This section describes how to set the following profiles for SHDSL interface configuration:

- *Setting pme-profile settings* on page 69
- *Setting DSL profile settings* on page 71
- *Configuring efm-port settings* on page 74

Setting pme-profile settings

A *pme-profile* (Physical Medium Entities) is available for each SHDSL WAN port. PME profiles are used to set link rates. To display PME profiles, enter **list pme-profile**:

```
zSH> list pme-profile
pme-profile  1-1-1-0/shdsl
pme-profile  1-1-2-0/shdsl
pme-profile  1-1-3-0/shdsl
pme-profile  1-1-4-0/shdsl
pme-profile  1-1-5-0/shdsl
pme-profile  1-1-6-0/shdsl
pme-profile  1-1-7-0/shdsl
pme-profile  1-1-8-0/shdsl
```

```
8 entries found.
```

To display the PME parameters in their default state, enter **get pme-profile**:

```
zSH> get pme-profile 1-1-1-0/shdsl
pme-profile  1-1-1-0/shdsl
efmCuPmeAdminSubType: ----------->  {ieee2basetlr}
efmCuPmeAdminProfile: ----------->  {0}
efmCuPAFRemoteDiscoveryCode: ---->  {}
efmCuPmeThreshLineAtn: ---------->  {0}
efmCuPmeThreshSnrMgn: ----------->  {0}
efmCuPmeLineAtnCrossingEnable: -->  {false}
efmCuPmeSnrMgnCrossingEnable: --->  {false}
efmCuPmeDeviceFaultEnable: ------>  {false}
efmCuPmeConfigInitFailEnable: --->  {false}
efmCuPmeProtocolInitFailEnable: ->  {false}
efmCuPme2BProfileDescr: --------->  {}
efmCuPme2BRegion: --------------->  {region1}
efmCuPme2BDataRate: ------------->  {0}
efmCuPme2BPower: ---------------->  {0}
efmCuPme2BConstellation: -------->  {adaptive}
efmCuPme2BProfileRowStatus: ----->  {active}
efmCuPmeNtr: -------------------->  {ntr-local-osc}
```

For the *efmCuPme2BRegion* parameter, the regions are set as specified in the relevant Regional Annex of [G.9991.2]. Regional settings place limitation on the max allowed data rate, power, and constellation. The possible values for this parameter are:

- region 1

  Annex A and F (North America)

- region 2

  Annex B and G (Europe)

Regions can only be changed when the link is down.

For the *efmCuPme2BDataRate*, setting the parameter to 0 sets the data rate to auto-negotiate. Entering a range between 192 and 5696 defines a specific range of the data rate.

Table 16 provides the settings for the *efmCuPme2BConstellation* parameter.

Table 16: efmCuPme2BConstellaltion settings

Constellation settings	Rate range
TCPAM16	192 to 3840
TCPAM32	768 to 5696
Adaptive	192 to 5696

To change *pme-profile* values, enter **update pme-profile** *interface/type*:

```
zSH> update pme-profile 1-1-3-0/shdsl
pme-profile  1-1-3-0/shdsl
Please provide the following: [q]uit.
efmCuPmeAdminSubType: ----------->  {ieee2basetlr}:
efmCuPmeAdminProfile: ----------->  {0}:
efmCuPAFRemoteDiscoveryCode: ---->  {}:
efmCuPmeThreshLineAtn: ---------->  {0}:
efmCuPmeThreshSnrMgn: ----------->  {0}:
efmCuPmeLineAtnCrossingEnable: -->  {false}:
efmCuPmeSnrMgnCrossingEnable: --->  {false}:
efmCuPmeDeviceFaultEnable: ------>  {false}:
efmCuPmeConfigInitFailEnable: --->  {false}:
efmCuPmeProtocolInitFailEnable: ->  {false}:
efmCuPme2BProfileDescr: --------->  {}:
efmCuPme2BRegion: --------------->  {region1}:
efmCuPme2BDataRate: ------------->  {0}:
efmCuPme2BPower: ---------------->  {0}:
efmCuPme2BConstellation: -------->  {adaptive}:
efmCuPme2BProfileRowStatus: ----->  {active}:
efmCuPmeNtr: ------------------->  {ntr-local-osc}:
....................
Save changes? [s]ave, [c]hange or [q]uit:
```

Setting DSL profile settings

The dsl-profile provides settings for DSL options, such as co/cpe mode, line-type, unit-mode, and others. The following table summarizes the commands required to configure SDSL interfaces on the EtherXtend:

Action	Command
Verify the type of SHDSL interface.	**`update dsl-config`** ***`index/`*****`shdsl`** Where index is of the form shelf-slot-port-subport or a user-defined string.
Verify the interface is active.	**`showlinestatus`** ***`shelf slot port`***

Automatic baud rate adaption and fixed rate settings

When you select the **shdsl-2btl** line type for an SHDSL interface, the EtherXtend can perform automatic baud rate adaption. This allows receiving devices to communicate with transmitting devices operating at different baud rates without the need to establish data rates in advance. By determining the baud rate from the transmitting device, the receiving EtherXtend automatically trains to match the line rate of the incoming data.

The automatic baud rate adaption process may take several minutes. This is because the CO and CPE device modems use an algorithm to step through a sequence of baud rates, where the devices establish a connection at each line rate and then move to the next higher rate until they reach the final rate they agree upon.

The adaptive [fixed-rate=0] and fixed line rate settings are defined in the efmCuPme2BDataRate entry of the pme-profile.

Table 17: Fix-bit-rate settings and modem train rates

CO	CPE	Then
Disabled	Disabled	Highest available rate is negotiated.
Disabled	Enabled	Modems train at CPE’s fixed rate.
Enabled	Disabled	Modems train at CO’s fixed rate.
Enabled	Enabled	Modems train at lowest fixed rate.

Specifying the type of DSL interface

The *dsl-config* profile supports the following parameters:

Parameter	Description
line-type	The DSL type supported on this interface. Values: **shdsl-2btl** Supports SHDSL-bonded connections. Default: **shdsl-2btl**
unit-mode	Specifies whether the unit is configured as a CO or CPE device. Values: **co** **cpe** Default: **cpe**
line-status-trap-enable	Specifies whether a line status trap should be sent whenever the DSL line goes up or down. Note that this setting does not apply to line status traps sent during system bootup. During bootup, line status traps are not sent. A DSL link down trap has a moderate severity level and a link up trap has a low severity. Default: **disabled**

To specify the interface as an SDSL line, set the *line-type* in the *dsl-config* profile, enter the **update dsl-config** *interface/type*:

```
zSH> update dsl-config 1-1-1-0/shdsl
dsl-config  1-1-1-0/shdsl
Please provide the following: [q]uit.
line-type: --------------->  {shdsl-2btl}:shdsl-2btl
unit-mode: --------------->  {cpe}:
line-status-trap-enable: ->  {disabled}:
```

```
admin-up-line-alarm: ----->  {disabled}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

Verifying the interface

Entering **showlinestatus** displays the status of the interfaces in the system. The following example displays some of the information returned by this command.

```
zSH> showlinestatus
--------- N2NBOND RP ---------
.........................
Type ---------------> N2NBOND (22)
Registered lines ---> 1
        .................
 Line Type-------> N2NBOND (22)
 GroupId --------> 22
 Redundancy -----> NONE (0)
 TxClk ----------> NONE (1)
 RefClkSrc ------> NO
 If_index -------> 21
 Shelf ----------> 1
 Slot -----------> 1
 Port -----------> 201
 SubPort --------> 0
--------- EFMBOND RP ---------
.........................
Type ---------------> EFMBOND (23)
Registered lines ---> 0
--------- SHDSL RP ---------
.........................
Type ---------------> SHDSL (18)
Registered lines ---> 4
 Line Type-------> SHDSL (18)
 GroupId --------> 10
 Status ---------> ACTIVE (1)
 Redundancy -----> NONE (0)
 TxClk ----------> NONE (1)
 RefClkSrc ------> NO
 If_index -------> 9
 Shelf ----------> 1
 Slot -----------> 1
 Port -----------> 1
 SubPort --------> 0
 Status ---------> ACTIVE (1)
```

To display the status of the interface, enter **dslstat** *interface/type*:

```
zSH> dslstat 1-1-4-0/shdsl
General Stats:
-------------
AdminStatus.................................UP
```

```
DslUpLineRate (bitsPerSec)..................0
DslDownLineRate (bitsPerSec)................0
DslMaxAttainableUpLineRate (bitsPerSec)......5696000
DslMaxAttainableDownLineRate (bitsPerSec)....5696000
Out Octets..................................0
Out Pkts/Cells..............................0
Out Discards................................0
Out Errors..................................0
In Octets...................................0
In Pkts/Cells...............................0
In Discards.................................0
In Errors...................................0
DSL Physical Stats:
------------------
DslLineSnrMgn (tenths dB)...................0
DslLineAtn (tenths dB)......................0
DslCurrOutputPwr (tenths dB)................0
LOFS........................................0
LOLS........................................0
LOSS........................................0
ESS.........................................352902
CRC Errors..................................0
Inits.......................................0
```

Verifying the type of DSL interface

The system creates **dsl-config** profiles for SHDSL cards with the appropriate settings. To view the dsl-config profiles, enter **get dsl-config** *interface/type*:

```
zSH> get dsl-config 1-1-4-0/shdsl
dsl-config  1-1-4-0/shdsl
line-type: --------------->  {shdsl-2btl}
unit-mode: --------------->  {cpe}
line-status-trap-enable: ->  {disabled}
admin-up-line-alarm: ----->  {disabled}
```

Configuring efm-port settings

The efm-port profile provides settings for EFM port, such as admin state, snr mode, and others. To view EFM ports, enter **list efm-port**:

```
zSH> list efm-port
efm-port  1-1-1-0/shdsl
efm-port  1-1-2-0/shdsl
efm-port  1-1-3-0/shdsl
efm-port  1-1-4-0/shdsl
efm-port  1-1-5-0/shdsl
efm-port  1-1-6-0/shdsl
efm-port  1-1-7-0/shdsl
efm-port  1-1-8-0/shdsl
8 entries found.
```

To display the range or options of EFM port parameters, enter **show efm-port**. The following values are available on the efm-port interface.

```
zSH> show efm-port
efmCuPAFAdminState:---------------->   enabled  disabled
efmCuPAFDiscoveryCode:------------->   {260}
efmCuAdminProfile:----------------->   {8}
efmCuTargetDataRate:--------------->   {1 - 999999}
efmCuTargetWorstCaseSnrMgn:-------->   {-10 - 21}
efmCuThreshLowBandwidth:----------->   {0 - 100000}
efmCuLowBandwidthEnable:----------->   true  false
efmCuTargetCurrentConditionMode:--->   true  false
efmCuTargetCurrentConditionSnrMgn:->   {-10 - 21}
efmCuTargetWorstCaseMode:---------->   true  false
```

To change or update the efm-port parameter, enter the **update- efm-port** *interface/type*:

```
zSH> update efm-port 1-1-1-0/shdsl
efm-port  1-1-1-0/shdsl
Please provide the following: [q]uit.
efmCuPAFAdminState: ---------------->  {enabled}:
efmCuPAFDiscoveryCode: ------------->  {}:
efmCuAdminProfile: ----------------->  {0x01}:
efmCuTargetDataRate: --------------->  {50000}:
efmCuTargetWorstCaseSnrMgn: -------->  {0}:
efmCuThreshLowBandwidth: ----------->  {0}:
efmCuLowBandwidthEnable: ----------->  {false}: true
efmCuTargetCurrentConditionMode: --->  {false}:
efmCuTargetCurrentConditionSnrMgn: ->  {6}:
efmCuTargetWorstCaseMode: ---------->  {true}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

Updating efmCuTargetWorstCaseSnrMgn

When the rate selection algorithm is designed to use efmCuTargetWorstCaseSnrMgn, the modem will select a more conservative connect rate based on a minimum Noise Level that is artificially determined. When efmCuTargetWorstCaseSnrMgn is set to 0dB, the modem automatically selects the connect rate that would result in 0dB SNR margin if the crosstalk noise level was equal to the noise in a binder fully loaded with SHDSL links. The result is that on lightly loaded loops, the connect rate is lower than it could be and the SNR Margin is much higher than it needs to be for reliable operation. However, when the loop plant fills up with SHDSL connections, the originally selected rate will still be valid, and there will be no service interruptions caused by retraining of loops in the binder as the noise profile changes when more links are added.

If the user would rather not use the efmCuTargetWorstCaseSnrMgn setting, then set efmCuTargetWorstCaseMode to false. The default setting for efmCuTargetWorstCaseMode is TRUE and efmCuTargetWorstCaseSnrMgn is 0.

Note: Previous releases of EtherXtend show the efmCuTargetWorstCaseSnrMgn to be 1. Enter the **set2dafault** command to view the current default setting of 0.

To view current parameter settings, enter **get efm-port** *interface/type*:

```
zSH> get efm-port 1-1-1-0/shdsl
efm-port  1-1-1-0/shdsl
efmCuPAFAdminState: ---------------->  {enabled}
efmCuPAFDiscoveryCode: ------------->  {}
efmCuAdminProfile: ----------------->  {0x01}
efmCuTargetDataRate: --------------->  {50000}
efmCuTargetWorstCaseSnrMgn: -------->  {0}
efmCuThreshLowBandwidth: ----------->  {0}
efmCuLowBandwidthEnable: ----------->  {false}
efmCuTargetCurrentConditionMode: --->  {false}
efmCuTargetCurrentConditionSnrMgn: ->  {6}
efmCuTargetWorstCaseMode: ---------->  {true}
```

EtherXtend EFM 802.3ah bonding

EFM (Ethernet in the First Mile) extends Ethernet signaling between the EtherXtend-EFM-SHDSL-24 card and EtherXtend or other EFM-enabled CPEs.

By default, all ports are configured in N2N bond groups and can be re-configured for EFM bonding.

This section describes the following:

- *Creating bond groups* on page 76
- *Displaying bond groups* on page 77
- *Changing bond group type* on page 78
- *Deleting bond groups* on page 78

Creating bond groups

The **bond add** and **list if-translate** commands are used to add and verify a single N2N or EFM bond group. If the bond group already exists, adding an identical group with a different bond type changes the bond group type.

Note: Bond groups created with CLI commands must be greater than 24 and less than 100. 100-series bond group IDs are used by ZMS and 200-series bond groups are auto-provisioned/discovered.

```
zSH> bond add group 1-1-40-0/n2nbond
zSH> bond add group 1-1-50-0/efmbond
```

To add a new member to an existing bond group:

```
zSH> bond add member 1-1-40-0/n2nbond 1-1-1-0/shdsl
zSH> list if-stack
  .....
if-stack  1-1-40-0/n2nbond/1-1-1-0-shdsl/n2nlink
```

To create a bond group with multiple members and view the bond groups:

```
zSH> bond add member 1-1-40-0/n2nbond 1-1-3-0/shdsl
1-1-4-0/shdsl
zSH> list if-translate
  .....
if-translate  1-1-40-0/n2nbond
if-translate  1-1-40-0-n2nbond/linegroup
```

Displaying bond groups

Bond groups can be displayed for all existing groups, a specific group, a specific slot, or link.

To display all configured bond groups:

```
zSH> bond show all
                  Bond Groups
  Slot   GrpId      Name            Type         State
   1      40        1-1-40-0        n2nbond       OOS
   1      102       1-1-102-0       efmbond       OOS
   1      101       1-1-101-0       n2nbond       OOS
```

To display a specific bond group:

```
zSH> bond show group 1-1-40-0/n2nbond
                  Bond Groups
  Slot   GrpId      Name            Type         State
   1      40        1-1-40-0         n2nbond       OOS
                  Group Members
  Slot    Port      Name            Type         State
   1        3       1-1-3-0         shdsl         OOS
   1        4       1-1-4-0         shdsl         OOS
```

To display bond groups by slot:

```
zSH> bond show slot 1
                  Bond Groups
  Slot   GrpId      Name            Type         State
   1      40        1-1-40-0        n2nbond       OOS
   1      102       1-1-102-0       efmbond       OOS
   1      101       1-1-101-0       n2nbond       OOS
```

To display bond groups for a specific link:

```
zSH> bond show link 1-1-40-0/shdsl
```

```
                    Bond Groups
Slot    GrpId         Name            Type         State
 1        40          1-1-40-0         n2nbond       OOS
                    Group Members
Slot     Port         Name            Type         State
 1        3           1-1-3-0         shdsl         OOS
 1        4           1-1-4-0         shdsl         OOS
```

Changing bond group type

Bond group type can be changed for individual bond groups or all bond groups used in a specified slot using **bond move** and **bond modify**.

```
zSH> bond move 1-1-102-0/efmbond 1-1-101-0/n2nbond
1-1-2-0/shdsl

zSH> bond modify n2n group 1-1-101-0/n2nbond

zSH> bond modify efmbond slot 1
```

Deleting bond groups

Bond groups can be deleted by individual member or entire group.

```
zSH> bond delete member 1-1-101-0/n2nbond 1-1-3-0/shdsl


zSH> bond show group 1-1-101-0/n2nbond
```

Bond group/physical line stats

Data in the **dslstat** command is provided for bond groups. The data is collected differently for N2N and EFM ports and bond groups. This section describes:

- *Packet counts* on page 78
- *Bond group bandwidth* on page 79

Packet counts

EFM bonding fragments packets across multiple lines so that packet counts for EFM ports indicate the number of EFM packet fragments for that port. At the physical port level, EFM unicast packet counts show the number of packet fragments for that port. Octets at the EFM physical port include all bytes received, including those from errored packet fragments and protocol overhead.

The packet count for N2N bond groups show the number of complete packets that traversed the bond group and indicate the number of unicast, multicast, and broadcast packets for that bond group. Octets at the N2N bond group include all bytes received from all valid packets; bytes from errored packets and protocol overhead are not included.

To display the aggregate statistics for a specified bond group interface or if-index, use **bond stats**.

Use **bond show** to view the type of bond group and the interface name for the bond group to gather statistics. The EtherXtend is always slot 1.

```
zSH> bond show slot 1
                   Bond Groups
  Slot    GrpId      Name            Type          State
   1       201       1-1-201-0       efmbond        ACT

zSH> bond stats 1-1-201-0/efmbond
****************** Bond group statistics ******************
                      Group Info
        Slot     GrpId      Interface Name      IfIndex
          1       201     1-1-201-0/efmbond       33

          UP               UP            17152000       0.00:10:15
                   Group Members
        Port      Interface Name      IfIndex
          4          1-1-4-0/shdsl      15
          6          1-1-6-0/shdsl      19
          1          1-1-1-0/shdsl      5
          5          1-1-5-0/shdsl      17
          7          1-1-7-0/shdsl      21
          2          1-1-2-0/shdsl      8
          8          1-1-8-0/shdsl      23
          3          1-1-3-0/shdsl      12
                Statistics (Received)
        Octets                           2955877408
        Ucast                            42200684
        Mcast                            0
        Bcast                            28
        Discards                         0
        Errors                           0
                Statistics (Transmitted)
        Octets                           2118780630
        Ucast                            252843879
        Mcast                            252843808
        Bcast                            0
        Discards                         0
```

Bond group bandwidth

Table 18 shows the bond group bandwidth rates for EFM 4-port bond groups.

Table 18: Bond group bandwidth

Frame Size	Downstream (pks/sec)	Upstream (pks/sec)	Total
64	40584	40584	81168
128	21478	21478	42956
256	11105	11105	22210
512	5547	5547	11094
1024	2826	2826	5652
1280	2269	2269	4538
1480	1967	1967	3934

8 CONFIGURING BRIDGING

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

Bridging overview

Bridges are configured with **bridge add** and the desired bridge type (uplink, downlink, intralink, tls for TLS, hub, and no type for transparent). This command creates a **bridge-interface-record** profile for the specified interface and sets the default values for the profile based on the bridge type. The **bridge add** command also supports uplink and downlink bridges that use VLANs.

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

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

- Asymmetrical bridge types are uplink, downlink, and intralink.
  - Uplink bridge

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 either a specific VLAN or globally for the system onto the uplink bridge. If an uplink is missing this configuration, traffic will not flow across the asymmetric VLAN.

– Downlink bridge

  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.

– Intralink bridge

  An intralink bridge is used in conjunction with an uplink bridge, where the uplink bridge is the path upstream to the network, and the intralink forwards traffic with unknown MAC addresses or multicasts to the configured bridge interface without attempting to learn the addresses of the attached devices or network. Traffic coming into this interface is forwarded to the uplink regardless of the destination MAC address. Broadcasts, multicasts, 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 will not have their source MAC addresses learned since this interface is not used when a MAC is known.

Intralink bridge interfaces require an additional configuration to take effect, which is a bridge-path. The bridge-path sets a default intralink path for either a specific VLAN or a global intralink for the system onto the intralink bridge. If an intralink is missing this configuration, traffic will not flow across the asymmetric VLAN.

- Symmetrical bridge types are transparent, transparent LAN service (TLS), and hub.
  - 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.
  - Transparent LAN service

    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.

Bridges also utilize VLAN tagging for tagged and untagged traffic segregation.

- Tagged bridging

  Tagged or Virtual LANs (VLANs) bridging that forward traffic based on MAC addresses and allows the segregation of a single Ethernet network into multiple virtual network segments by mapping physical ports to VLAN IDs.
- Untagged bridging

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.

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

Note: The EtherXtend ports can support both IP termination or bridging on different virtual circuits. However, each virtual circuit must be configured for either IP termination or bridging and cannot support both at the same time.

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.

Bridge enhancements to flood unknowns and multicasts

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

- *FloodUnknown parameter* on page 84
- *FloodMulticast parameter* on page 85

FloodUnknown parameter

The FloodUknown parameter provides the ability to toggle the flooding of unknown unicast destination frames. When this parameter is set to true, the EtherXtend 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 EtherXtend 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 EtherXtend to flood all multicast traffic received on a bridge out to all other ports in the VLAN. This is useful for architectures where the EtherXtend 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 change a parameter, enter **update bridge-interface-record** *interface/type*:

```
zSH> update bridge-interface-record 1-1-201-0-n2nbond/bridge
bridge-interface-record  1-1-201-0-n2nbond/bridge
Please provide the following: [q]uit.
vpi: --------------------------------->  {0}:
vci: --------------------------------->  {0}:
vlanId: ------------------------------>  {0}:
stripAndInsert: ---------------------->  {true}:
customARP: --------------------------->  {false}:
filterBroadcast: --------------------->  {false}:
learnIp: ----------------------------->  {false}:
learnUnicast: ------------------------>  {true}:
maxUnicast: -------------------------->  {100}:
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: ------------------------>  {true}:
floodMulticast: ---------------------->  {false}:true
bridgeIfEgressPacketRuleGroupIndex: -->  {0}:
bridgeIfTableBasedFilter: ------------>  {NONE(0)}:
bridgeIfDhcpLearn: ------------------->  {NONE(0)}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record created.
```

Broadcasts and bridging

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

In general, broadcasts sent to 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 EtherXtend. 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 EtherXtend 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 EtherXtend 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 EtherXtend 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.

VLANs

Figure 13 shows a typical VLAN configuration. On the access (subscriber) side, VLANs 1 and 2 are separate DSL networks connected to the EtherXtend via EtherXtend devices. 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.

The side of the connection closest to the subscriber is called the downlink interface. The upstream egress is called the uplink interface. When the EtherXtend 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 the downlink interface as tagged. Tagged downlink interfaces can be used for subtended EtherXtends or subscribers expecting tagged traffic with Transparent LAN Server (TLS) service.

Note: The EtherXtend supports VLAN IDs from 1 to 4096. Multiple VLAN interfaces can be added to the same physical port and VC.

Figure 13: Typical VLAN network

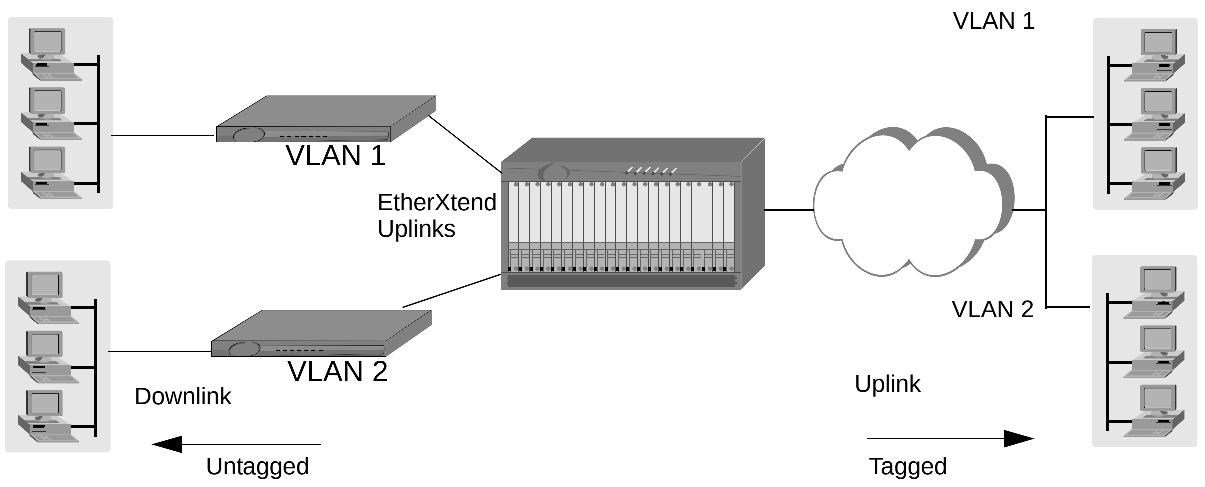
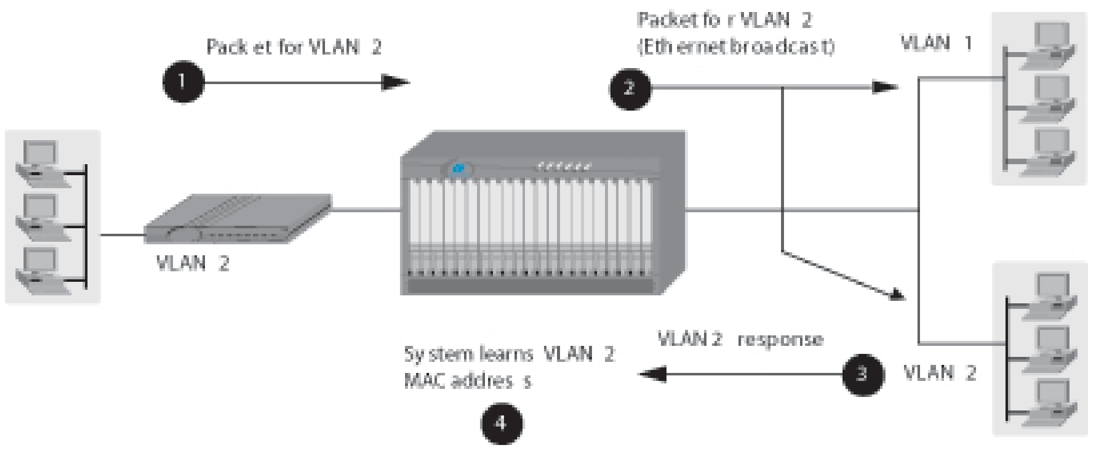


Figure 14: Learning a MAC address



Configuring bridges using VLANs

To configure a downstream bridge that directs traffic on a VLAN:

1 To add a bridge for the downstream connection, enter **bridge add** *1-1-port-interface/type downlink vlan vlan id*. Multiple VLAN interfaces can be added to the same physical port and bond group.

```
zSH> bridge add 1-1-1-0/eth downlink vlan 60
```

This command adds a downlink to an Ethernet port on the LAN that uses VLAN 60.

2 To verify the bridge interface, enter **bridge-interface-record** *interface/type*:

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

```
zSH> get bridge-interface-record 1-1-1-0-eth/bridge
bridge-interface-record  1-1-1-0-eth/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {60}
stripAndInsert: ---------------------->  {true}
customARP: --------------------------->  {false}
filterBroadcast: --------------------->  {false}
learnIp: ----------------------------->  {false}
learnUnicast: ------------------------>  {true}
maxUnicast: -------------------------->  {100}
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: ------------------------>  {true}
floodMulticast: ---------------------->  {true}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------>  {NONE(0)}
bridgeIfDhcpLearn: ------------------->  {NONE(0)}
```

3 To create a bridge interface on the WAN bond group for the upstream connection, enter **bridge add** *1-1-bondgroup-0/type uplink vlan vlan id*:

```
zSH> bridge add 1-1-201-0/efmbond uplink vlan 1
```

This creates a bridge interface on the WAN bond group interface.

4 To create a bridge path for this uplink, enter **bridge-path add** *1-1-bondgroup-0-interface/type global*:

```
zSH> bridge-path add 1-1-201-0-efmbond/bridge global
```

The **global** setting specifies that the EtherXtend should send all VLAN traffic to this port. A VLAN ID can also be used when the EtherXtend should send only traffic from a specific VLAN ID to this port. It is recommended not to change the default settings unless advanced bridge configuration is required.

```
fm1> get bridge-interface-record 1-1-201-0-efmbond/bridge
bridge-interface-record  1-1-201-0-efmbond/bridge
vpi: ---------------------------------->  {0}
vci: ---------------------------------->  {0}
vlanId: ------------------------------->  {0}
stripAndInsert: ----------------------->  {true}
customARP: ---------------------------->  {false}
filterBroadcast: ---------------------->  {false}
learnIp: ------------------------------>  {false}
learnUnicast: ------------------------->  {true}
maxUnicast: --------------------------->  {100}
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: ------------------------->  {true}
floodMulticast: ----------------------->  {true}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ------------->  {NONE(0)}
bridgeIfDhcpLearn: -------------------->  {NONE(0)}
```

Note: To delete a downlink bridge with a VLAN, the VLAN ID must be specified in the **bridge delete** command.

Bridging behavior for untagged, tagged, and s-tagged

This section provides a discussion and examples of various types of bridges and their settings:

- *Overview* on page 90
- *Untagged bridging examples* on page 91

- *Tagged and s-tagged bridging examples* on page 93

Overview

Bridges also utilize VLAN and SLAN tagging for untagged, tagged, and s-tagged, traffic segregation.

- 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.

- Tagged bridging

  Tagged or Virtual LANs (VLANs) 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.

  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.

- s-tagged

  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.

Untagged bridging examples

Configuring untagged or transparent bridging allows traffic to be forwarded from a downlink interface through the EtherXtend uplink interface based on the destination MAC address without tagging or modification to the frame. Refer to the *Zhone 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 To add an untagged bridge to the upstream SHDSL interface, enter **bridge add** *interface/type:*

```
zSH> bridge add 1-1-40-0/efmbond
Adding bridge on 1-1-40-0/efmbond
Created bridge-interface-record 1-1-40-0-efmbond-0/bridge
```

This example adds a default transparent bridge interface to the SHDSL card on the MALC and sets the parameters to the default transparent bridge interface settings.

The following example shows the default **bridge-interface-record** settings defaults. It is recommended not to change the default settings unless advanced bridge configuration is required. To view the defaults, enter **get bridge-interface-record** *interface/type*:

```
zSH> get bridge-interface-record 1-1-40-0-efmbond-0/bridge
bridge-interface-record  1-1-40-0-efmbond-0/bridge
vpi: ---------------------------------->  {0}
vci: ---------------------------------->  {0}
vlanId: ------------------------------->  {0}
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)}
```

2 To add a transparent bridge that accepts transparent/untagged traffic on the EtherXtend units's downstream Ethernet port, enter **bridge add** *interface/type*:

```
zSH> bridge add 1-1-3-0/eth
Adding bridge on 1-1-3-0/eth
Created bridge-interface-record 1-1-3-0-eth/bridge
```

3 To show the default transparent **bridge-interface-record** settings for the uplink, enter **get bridge-interface-record** *interface/type*. Unless advanced bridge configuration is required, it is recommended not to change the default settings.

```
zSH> get bridge-interface-record 1-1-3-0-eth/bridge
bridge-interface-record  1-1-3-0-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)}
```

4 To verify that both sides of the bridge are present, enter:

```
zSH> bridge show
Typ VLAN        Bridge                        St  Table Data
--------------------------------------------------------------------------------
               0 1-1-40-0-efmbond/bridge       UP D 00:00:00:00:a5:03
               0 1-1-3-0-eth/bridge            UP D 00:00:00:00:05:03
```

Tagged and s-tagged bridging examples

When adding bridges for VLAN tagged (single tagged) bridges, the bridge interface name includes the VLAN ID, even when the default VLAN ID of 0 is not explicitly added. By entering **bridge add** *interface/type*, the 0 is automatically included:

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

Entering **bridge add** *interface/type vlan 4000 tagged* shows a tagged bridge with VLAN 4000:

```
zSH> bridge add 1-1-1-0/eth vlan 4000 tagged
Adding bridge on 1-1-1-0/eth
Created bridge-interface-record 1-1-1-0-eth-4000/bridge

zSH> bridge add 1-1-1-0/eth vlan 1000 slan 17 tagged
Adding bridge on 1-1-1-0/eth
Created bridge-interface-record 1-1-1-0-eth-1000/bridge
```

Enter **bridge show** to view the bridges just created:

```
zSH> bridge show
Typ VLAN        Bridge                        St  Table Data
------------------------------------------------------------------------------------
dwn         123 1-1-4-0-eth/bridge            UP  D 00:1a:6d:13:19:8f
                                                S VLAN 123 default [U: 3600 sec, M: 150 sec, I: 0 sec]
dwn        123 1-1-40-0-efmbond/bridge       UP  S VLAN 123 default [U: 3600 sec, M: 150 sec, I: 0 sec]
    Tagged      1-1-1-0-eth-0/bridge          DWN
    Tagged 4000  1-1-1-0-eth-4000/bridge       DWN
    Tg 1000/17   1-1-1-0-eth-1000/bridge       DWN
```

Bridge 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 **show bridge-interface-record**:

```
zSH> show bridge-interface-record
vpi:--------------------------------->   {0 - 4095}
vci:--------------------------------->   {0 - 65535}
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
bridgeIfIngressPacketRuleGroupIndex:->   {0 - 2147483647}
vlanIdCOS:--------------------------->   {0 - 7}
outgoingCOSOption:------------------->   disable  all
outgoingCOSValue:-------------------->   {0 - 7}
s-tagTPID:--------------------------->   {33024 - 37376}
s-tagId:----------------------------->   {0 - 4095}
s-tagStripAndInsert:----------------->   false  true
s-tagOutgoingCOSOption:-------------->   s-tagdisable  s-tagall
s-tagIdCOS:-------------------------->   {0 - 7}
s-tagOutgoingCOSValue:--------------->   {0 - 7}
```

```
mcastControlList:-------------------->   {264}
maxVideoStreams:--------------------->   {0 - 210}
isPPPoA:----------------------------->   false  true
floodUnknown:------------------------>   false  true
floodMulticast:---------------------->   false  true
bridgeIfEgressPacketRuleGroupIndex:-->   {0 - 2147483647}
bridgeIfTableBasedFilter:------------>   none+mac+ip
bridgeIfDhcpLearn:------------------->   none+mac+ip
```

To modify a parameter in the bridge-interface-record such as the COS, enter **update bridge-interface-record** *interface/type* and make the changes.

```
zSH> update bridge-interface-record 1-1-3-0-eth/bridge
bridge-interface-record  1-1-3-0-eth/bridge
Please provide the following: [q]uit.
vpi: ---------------------------------->  {0}:
vci: ---------------------------------->  {0}:
vlanId: ------------------------------->  {800}:
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: --------------------->  {2}:
maxVideoStreams: ---------------------->  {1}:
isPPPoA: ------------------------------>  {false}:
floodUnknown: ------------------------->  {false}:
floodMulticast: ----------------------->  {false}:
bridgeIfEgressPacketRuleGroupIndex: -->  {0}:
bridgeIfTableBasedFilter: ------------>  {NONE(0)}:
bridgeIfDhcpLearn: ------------------->  {NONE(0)}:
....................
Save changes? [s]ave, [c]hange or [q]uit:
```

Q-in-Q VLAN tagging

The IEEE 802.1ad (also know as Q-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 EtherXtend 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, intralink, 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 EtherXtend 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.

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.

The **bridge add** command supports adding s-tag IDs from the command line. This example adds interface *1-1-2-0/eth downlink* with VLAN 100, SLAN 200, COS value of 7 and sCOS value of 6.

```
zSH> bridge add 1-1-2-0/eth downlink vlan 100 slan 200 tagged cos 7 scos 6
Adding bridge on 1-1-2-0/eth
Created bridge-interface-record 1-1-2-0-eth-100/bridge
```

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

```
zSH> get bridge-interface-record  1-1-2-0-eth-100/bridge
bridge-interface-record  1-1-2-0-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: -------------------------->  {6}
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:

- *Packet-rule records (Option 82, Forbid OUI, DHCP relay)* on page 98
- *VLAN ID stripandInsert parameter* on page 100
- *Broadcast suppression* on page 101
- *Bridge with DHCP relay* on page 102

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

Packet-rule records (Option 82, Forbid OUI, DHCP relay)

The EtherXtend 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 EtherXtend bridge interfaces:

- bridgeinsertoption82:

  packetRuleValue 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

  packetRuleValue 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

  packetRuleValue 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. EtherXtend supports *bridgeinsertoption82*, *bridgedhcprelay*, and *bridgeforbidoui*.

```
zSH> show packet-rule-record
packetRuleType:--->   bridgeinsertoption82  bridgedhcprelay
bridgeinsertpppoevendortag  bridgeforbidoui  ratelimitdiscard
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}
```

Note: Bridge configuration record settings supersede the global filter settings set using the **bridge-path add filter** command.

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 bridgeIfConfigGroupIndex parameter.

To configure a bridge configuration group:

```
zSH> update bridge-interface-record 1-1-40-0-efmbond/bridge
bridge-interface-record  1-1-40-0-efmbond/bridge
Please provide the following: [q]uit.
vpi: --------------------------------->  {0}:
vci: --------------------------------->  {0}:
vlanId: ------------------------------>  {123}:
stripAndInsert: ---------------------->  {true}:
customARP: --------------------------->  {false}:
filterBroadcast: --------------------->  {false}:
learnIp: ----------------------------->  {true}:
learnUnicast: ------------------------>  {true}:
maxUnicast: -------------------------->  {5}:
learnMulticast: ---------------------->  {true}:
forwardToUnicast: -------------------->  {false}:
forwardToMulticast: ------------------>  {false}:
forwardToDefault: -------------------->  {true}:
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: -------------------->  {}:
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.
```

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 EtherXtend 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}:
```

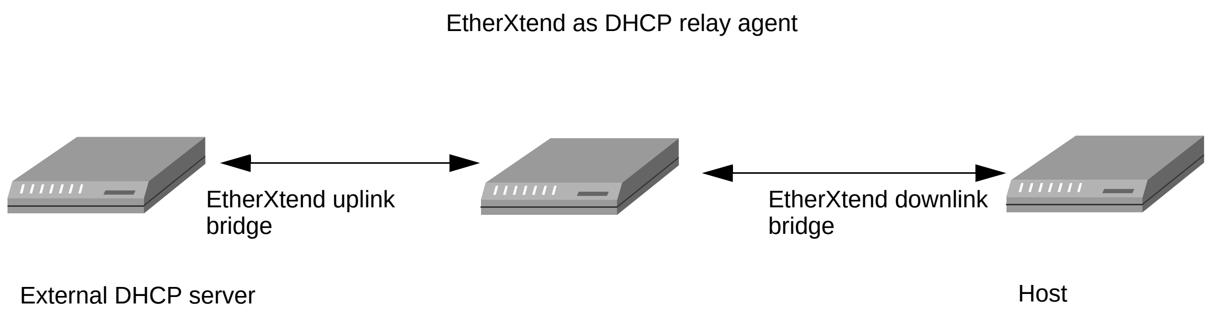
```
bridgeIfCustomDHCP: ------------------>  {false}: **true**
bridgeIfIngressPacketRuleGroupIndex: ->  {0}:
vlanIdCOS: --------------------------->  {0}:
outgoingCOSOption: ------------------->  {disable}:
outgoingCOSValue: -------------------->  {0}:
s-tagTPID: --------------------------->  {0x8100}:
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.
```

Bridge with DHCP relay

The EtherXtend enables bridges to be configured as DHCP relay agents. All DCHP messages on the bridge will have Option 82 information inserted and be passed up through an IP interface to a external DHCP server.

Figure 15 illustrates the traffic flow when the EtherXtend is configured with bridges to support DHCP relay.

Figure 15: Bridge supported DHCP relay



Configuring bridges to support DHCP relay

This procedure describes how to configure bridges on the EtherXtend to support DHCP relay. This procedure assumes the following configuration has already been performed on the EtherXtend.

- Downlink bridge to the host
- Uplink bridge to network
- IP interface on the EtherXtend with a route available to the DHCP server

To configure bridge support for DHCP relay:

1 Configure the packet-rule-record using the group number of the bridge and add the record to either the bridgeIfIngressPacketRuleGroupIndex interface or the bridgeIfEngressPacketRuleGroupIndex interface depending on the type of bridge.

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

2 Verify that the bridge-interface-record contains correct **bridge IfConfigGroupIndex** value. This value represents the bridge configuration group index specified for the bridge-config-record.

```
zSH> get bridge-interface-record  1-1-4-0-eth/bridge
vpi: ---------------------->  {0}
vci: ---------------------->  {35}
vlanId: ------------------->  {0}
stripAndInsert: ----------->  {true}
customARP: ---------------->  {false}
filterBroadcast: ---------->  {false}
learnIp: ------------------>  {true}
learnUnicast: ------------->  {true}
maxUnicast: --------------->  {5}
learnMulticast: ----------->  {true}
forwardToUnicast: --------->  {false}
forwardToMulticast: ------->  {false}
forwardToDefault: --------->  {true}
bridgeIfCustomDHCP: ------->  {false}
bridgeIfConfigGroupIndex: ->  {1} bridge-config-records 1/1, 2/2, etc.
vlanIdCOS: ---------------->  {0}
outgoingCOSOption: -------->  {disable}
outgoingCOSValue: --------->  {0}
s-tagTPID: ---------------->  {0x8100}
```

```
   s-tagId: ----------------->  {0}
   s-tagStripAndInsert: ------>  {false}
   s-tagOutgoingCOSOption: --->  {s-tagdisable}
   s-tagIdCOS: -------------->  {0}
   s-tagOutgoingCOSValue: ---->  {0}
```

Verify the dhcp-server-subnet with subnetgroup index matching the bridgeConfigValue is configured to forward DHCP requests to the desired external DHCP server. In this example, the bridgeConfigValue of 1 in the bridge-config-record matches the subnetgroup value specified in the dhcp-server-subnet profile. These values tell the DHCP relay agent to send the DHCP packets to the specified DHCP external server at 172.16.88.73.

```
zSH> get dhcp-server-subnet 1
   network: --------------->  {10.11.1.0}
   netmask: --------------->  {255.255.255.0}
   domain: ---------------->  {0}
   range1-start: ---------->  {10.11.1.10}
   range1-end: ------------>  {10.11.1.250}
   range2-start: ---------->  {0.0.0.0}
   range2-end: ------------>  {0.0.0.0}
   range3-start: ---------->  {0.0.0.0}
   range3-end: ------------>  {0.0.0.0}
   range4-start: ---------->  {0.0.0.0}
   range4-end: ------------>  {0.0.0.0}
   default-lease-time: ---->  {-1}
   min-lease-time: -------->  {-1}
   max-lease-time: -------->  {-1}
   boot-server: ----------->  {0.0.0.0}
   bootfile: -------------->  {}
   default-router: -------->  {10.11.1.1}
   primary-name-server: --->  {0.0.0.0}
   secondary-name-server: ->  {0.0.0.0}
   domain-name: ----------->  {}
   subnetgroup: ----------->  {1}  matches bridgeConfigValue of 1 in the bridge-config-record
   stickyaddr: ------------>  {enable}
   external-server: ------->  {172.16.88.73}
```

COS in bridges

The EtherXtend supports setting COS values in Ethernet VLAN headers for bridged packets. 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, intralink, 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 EtherXtend sets and preserves the COS settings to ensure these settings are passed to other Ethernet devices in the network for QOS processing.

Adding a bridge

Enter **bridge add** *interface/type* to create a bridge with a VLAN ID of 50.

```
zSH> bridge add 1-1-2-0/eth vlan 50
Adding bridge on 1-1-2-0/eth
```

Verifying bridge settings

To verify bridge settings, enter **get bridge-interface-record** 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 1-1-201-0-efmbond/bridge
vpi: --------------------->  {0}
vci: --------------------->  {0}
vlanId: ------------------>  {0}
stripAndInsert: ----------->  {false}
customARP: ---------------->  {true}
filterBroadcast: ---------->  {true}
learnIp: ------------------>  {false}
learnUnicast: ------------->  {false}
maxUnicast: --------------->  {0}
learnMulticast: ----------->  {false}
forwardToUnicast: --------->  {true}
forwardToMulticast: ------->  {true}
forwardToDefault: --------->  {false}
bridgeIfCustomDHCP: ------->  {true}
bridgeIfConfigGroupIndex: ->  {0}
vlanIdCOS: ---------------->  {0}
outgoingCOSOption: -------->  {disable}
outgoingCOSValue: --------->  {0}
s-tagTPID: ---------------->  {0x8100}
s-tagId: ------------------>  {0}
s-tagStripAndInsert: ------>  {false}
s-tagOutgoingCOSOption: --->  {s-tagdisable}
s-tagIdCOS: --------------->  {0}
s-tagOutgoingCOSValue: ---->  {0}
```

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

```
zSH> get bridge-interface-record 1-1-3-0-eth/bridge
bridge-interface-record  1-1-3-0-eth/bridge
vpi: --------------------------------->  {0}
vci: --------------------------------->  {0}
vlanId: ------------------------------>  {800}
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: ------------------->  {2}
maxVideoStreams: -------------------->  {1}
isPPPoA: ---------------------------->  {false}
floodUnknown: ----------------------->  {false}
floodMulticast: --------------------->  {false}
bridgeIfEgressPacketRuleGroupIndex: -->  {0}
bridgeIfTableBasedFilter: ----------->  {NONE(0)}
bridgeIfDhcpLearn: ------------------>  {NONE(0)}
```

In addition, you can run the **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
```

EtherXtend CO and CPE mode bridge scenarios

EtherXtends can be configured in either CO mode or CPE mode with either Transparent LAN Services (TLS) or uplink/downlink bridges. To enable bridged interfaces on the EtherXtend, different bridge types can be configured on the device depending on the mode of operation.

- *Bridges on EtherXtend CPE mode* on page 107
- *Configure the EtherXtend in CO mode* on page 110

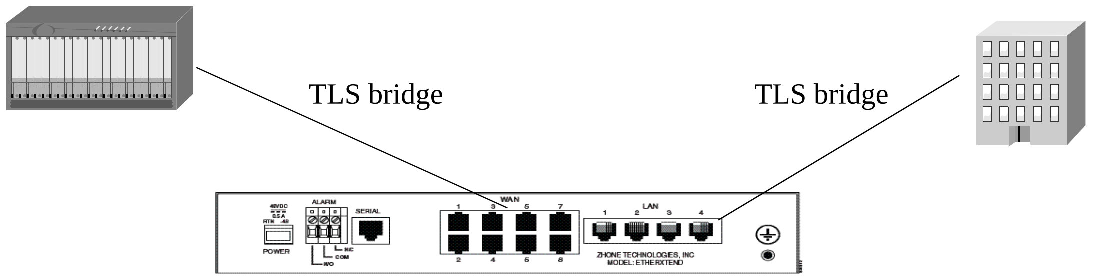
Bridges on EtherXtend CPE mode

When the EtherXtend is configured in CPE mode, the WAN traffic connects to a MALC EFM card through a bonded connection and the LAN traffic connects to subscribers. In this mode, the EtherXtend WAN and LAN interfaces can use either TLS bridging or uplink/downlink bridging.

Configure TLS 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. A TLS bridge is used with only other TLS bridges. This should not be used with any asymmetrical bridges. TLS bridges must be configured on both the WAN and the LAN ports on the EtherXtend as shown in Figure 16.

Figure 16: EtherXtend CPE mode with TLS bridges



Creating the TLS bridge on the EtherXtend

1 To find the interface associated with the default bond group configured on the WAN ports, enter **interface show.**

```
zSH> interface show
1 interface
```

```
Interface      Status  Rd/Address           Media/Dest Address      IfName
--------------------------------------------------------------------------------
1/1/40/0/ip    UP      1 10.250.1.40/24      00:01:47:36:2e:7b 1-1-40-0-efmbond-7
--------------------------------------------------------------------------------
```

The active default EFM interface *1-1-40-0-efmbond-7* is shown.

To display the default bond group, enter **bond show all**:

```
zSH> bond show all
                  Bond Groups
  Slot   GrpId      Name            Type         State
   1      40        1-1-40-0        efmbond       ACT
```

2 To create a TLS bridge on the WAN interface using the default bond group interface, enter **bridge add** *interface/type tls*. In this example, the default interface is *1-1-40-0/efmbond* with a VLAN ID of 7.

```
zSH> bridge add 1-1-40-0/efmbond tls vlan 7
Adding bridge on 1-1-40-0/efmbond
Created bridge-interface-record 1-1-40-0-efmbond/bridge
```

3 Next, create a TLS bridge on the LAN interface with **bridge add** *interface/type tls*. This example configures a TLS bridge on the first Ethernet LAN interface 1-1-1-0-eth with a VLAN ID of 200.

```
zSH> bridge add 1-1-1-0/eth tls vlan 200
Adding bridge on 1-1-1-0/eth
Created bridge-interface-record 1-1-1-0-eth/bridge
```

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

```
zSH> bridge show
Typ        VLAN        Bridge                        St  Table Data
---------------------------------------------------------------------------------
tls         200 1-1-1-0-eth/bridge                   DWN
tls           7 1-1-40-0-efmbond/bridge              UP
```

Configure uplink and downlink 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.

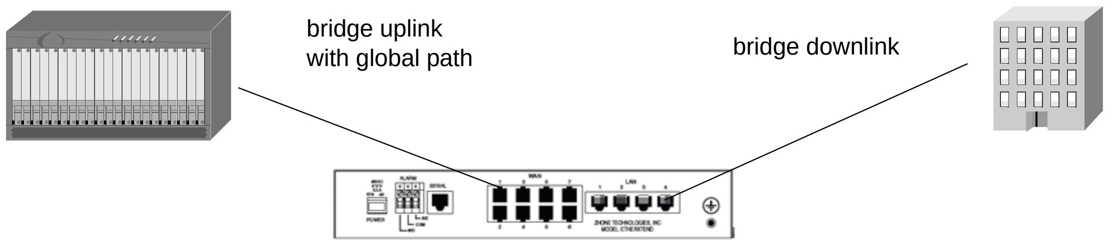
Uplink bridge interfaces require an additional bridge-path configuration to set a default path for either a specific VLAN or globally for the system onto the uplink bridge. If an uplink is missing this configuration, traffic will not flow across the asymmetric VLAN.

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.

Configure the EtherXtend with uplink and downlink bridges when the uplink bridge is the path upstream to the network, and the downlink bridge is the learning interface facing subscribers as shown in Figure 17.

Figure 17: EtherXtend in CPE mode with uplink and downlink bridges



Creating uplink and downlink bridges on the EtherXtend

1 To find the interface associated with the default bond group configured on the WAN ports, enter **interface show**. This examples shows the active default EFM interface *1-1-40-0-efmbond-7*.

```
zSH> interface show
1 interface
Interface      Status  Rd/Address            Media/Dest Address      IfName
--------------------------------------------------------------------------------
1/1/40/0/ip    UP      1 10.250.1.40/24      00:01:47:36:2e:7b 1-1-40-0-efmbond-7
--------------------------------------------------------------------------------
```

To display the default bond group, enter **bond show all**:

```
zSH> bond show all
                  Bond Groups
  Slot   GrpId      Name            Type        State
   1      40        1-1-40-0        efmbond      ACT
```

2 To configure an uplink bridge on the WAN interface enter, **bridge add** *interface/type uplink* using the interface of the bond group.

```
zSH> **bridge add 1-1-40-0/efmbond uplink**
Adding bridge on 1-1-40-0/efmbond
Created bridge-interface-record 1-1-40-0-efmbond-0/bridge
```

To create a bridge path for the uplink enter, **bridge-path add** *interface/type global*:

```
zSH>  bridge-path add 1-1-40-0-efmbond-0/bridge global
Bridge-path added successfully
```

3 To create a downlink bridge on the LAN interface, enter **bridge add** *interface/type downlink* VLAN 100:

```
zSH> bridge add 1-1-1-0/eth downlink vlan 100
Adding bridge on 1-1-1-0/eth
Created bridge-interface-record 1-1-1-0-eth/bridge
```

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

```
zSH> bridge show
Typ VLAN        Bridge                         St  Table Data
---------------------------------------------------------------------------------
dwn         123 1-1-4-0-eth/bridge             UP  D 00:13:72:de:92:2e
                                                 D 00:1a:6d:13:19:8f
                                                S VLAN 123 default [U: 3600 sec,
M: 150 sec, I: 0 sec]
upl Tagged      1-1-40-0-efmbond-0/bridge     UP  S Global default [U: 3600 sec,
M: 150 sec, I: 0 sec]
dwn         100 1-1-1-0-eth/bridge             DWN
```

EtherXtend CO mode with subtended EtherXtends in CPE mode

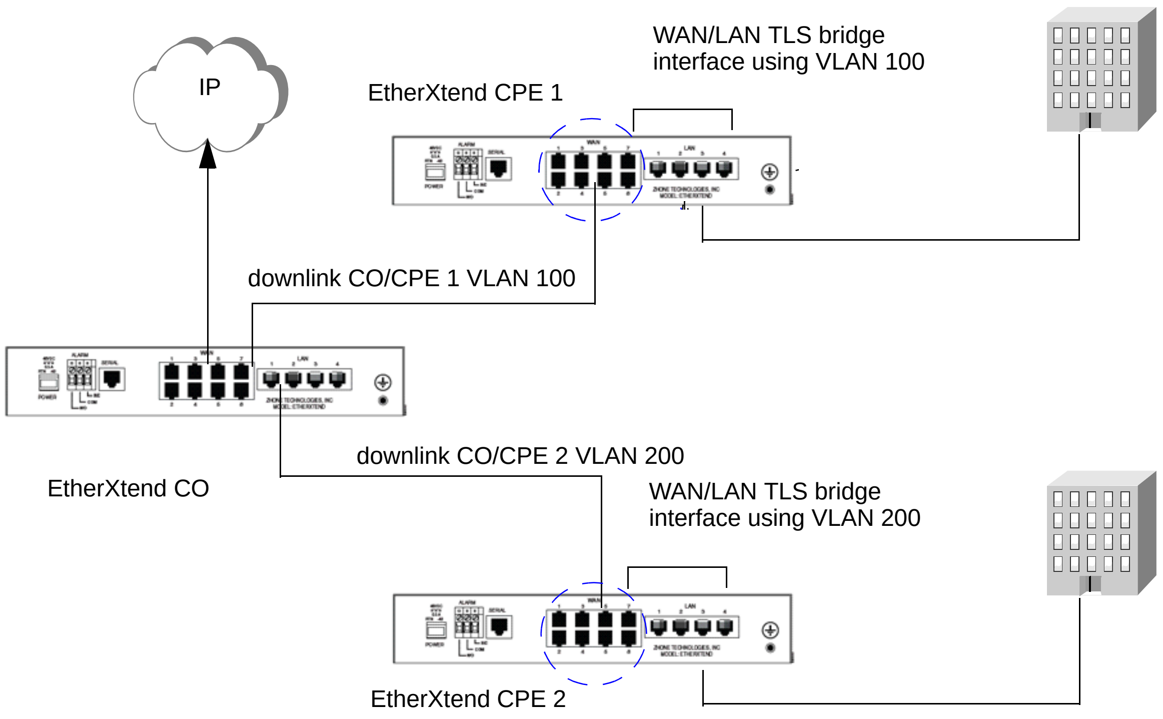
EtherXtend configurations can bridge together devices in both CO mode and CPE mode.

Configure the EtherXtend in CO mode

The EtherXtend can be configured in CO mode which uses an uplink bridge to the outside network and a downlink bridges to subtended EtherXtends in CPE mode.

Figure 18 shows one EtherXtend device in CO mode and two EtherXtend devices configured in CPE mode. The EtherXtend CPE 1 uses a TLS bridge with VLAN 100 to connect the WAN to the LAN. The EtherXtend CPE 2 also uses a TLS bridge with VLAN 200 to connect the WAN to the LAN.

Figure 18: EtherXtend CO scenario



Configure the EtherXtend in CO mode

On EtherXtend in CO mode, create an uplink bridge on a WAN Ethernet port to connect to the external network and create two downlink bridges on the WAN ports that connect to the two subtended EtherXtends in CPE mode. In this example, the downlinks use VLAN 100 and VLAN 200. Then create TLS bridges from the WANs to the LANs.

Note: This procedure assumes that a device in the network is connected to the EtherXtend in CO mode.

1 Display the default bond group, enter **bond show all**:

```
zSH> bond show all
                  Bond Groups
 Slot    GrpId      Name            Type         State
  1        99       1-1-99-0        efmbond       ACT
```

2 To configure an uplink bridge on the WAN interface enter, **bridge add** *interface/type uplink* using the interface of the bond group.

```
zSH> bridge add 1-1-40-0/efmbond uplink
Adding bridge on 1-1-40-0/efmbond
Created bridge-interface-record 1-1-40-0-efmbond-0/bridge
```

To create a bridge path for the uplink enter, :

```
zSH>  bridge-path add 1-1-40-0-efmbond-0/bridge global
Bridge-path added successfully
```

3 To create the downlink bridge on the WAN port for CPE 1 using VLAN 100, enter **bridge add** *interface/type downlink vlan 100*:

```
zSH> bridge add 1-1-40-0/efmbond downlink vlan 100
```

All bridge traffic will be forwarded to the EtherXtend CPE 1 on VLAN 100.

4 To create the downlink bridge for CPE 2 with VLAN 200 on the WAN port, enter **bridge add** *interface/type downlink vlan 200*:

```
zSH> bridge add 1-1-40-0/efmbond downlink vlan 200
```

All bridge traffic will be forwarded to the EtherXtend CPE 2 on VLAN 200.

Configure the EtherXtend in CPE mode

The subtended EtherXtends in CPE mode are configured with TLS bridges to direct traffic to the subscriber.

Creating TLS bridges on the EtherXtend CPE 1

When using TLS bridges on a device, both the WAN port and the LAN port must use TLS bridges.

1 To find the bond group number on the WAN, enter the **bond show all**:

```
zSH> bond show all
                  Bond Groups
  Slot   GrpId      Name            Type         State
   1      40        1-1-40-0        efmbond       ACT
```

2 To create a TLS bridge interface on the WAN port using the bond group number, enter **bridge add** *interface/type tls*:

```
zSH> bridge add 1-1-40-0-efmbond-0/bridge tls
```

3 To create a TLS bridge interface on the LAN port to the business, enter **bridge add** *interface/type tls*:

```
zSH> bridge add 1-1-1-0-eth/bridge tls
```

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

```
fm1> bridge show
Typ VLAN        Bridge                        St  Table Data
--------------------------------------------------------------------------------
tls           0 1-1-1-0-eth/bridge            UP
tls           0 1-1-40-0-efmbond/bridge       UP
```

Creating TLS bridges on the EtherXtend CPE 2

When using TLS bridges on a device, both the WAN port and the LAN port must use TLS bridges.

1 To find the bond group number on the WAN, enter the **bond show all**:

```
zSH> bond show all
                   Bond Groups
  Slot   GrpId       Name            Type         State
   1      40         1-1-40-0        efmbond       ACT
```

2 To create a TLS bridge interface on the WAN port using the bond group number, enter **bridge add** *interface/type tls*:

```
zSH> bridge add 1-1-40-0-efmbond-0/bridge tls
```

3 To create a TLS bridge interface on the LAN port to the business, enter **bridge add** *interface/type tls*:

```
zSH> bridge add 1-1-1-0-eth/bridge tls
```

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

```
fm1> bridge show
Typ VLAN         Bridge                        St  Table Data
---------------------------------------------------------------------------------
tls            0 1-1-1-0-eth/bridge            UP
tls            0 1-1-40-0-efmbond/bridge       UP
```

EtherXtend bridge commands

The EtherXtend supports the following **bridge** commands:

- *Bridge delete command* on page 113
- *Bridge show command* on page 113
- *Bridge stats* on page 114

Refer to the *Zhone 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.

```
zSH> bridge delete 1-1-40-0-efmbond-0/bridge
1-1-40-0-efmbond-0/bridge Delete complete
```

Bridge show command

The **bridge show** command displays either a single bridge path entry or the entire bridge table.

```
zSH> **bridge show**
Typ VLAN        Bridge                       St  Table Data
---------------------------------------------------------------------------------
dwn         123 1-1-4-0-eth/bridge           UP  D 00:1a:6d:13:19:8f
                                               S VLAN 123 default [U: 3600 sec, M: 150
sec, I: 0 sec]
              0 1-1-3-0-eth/bridge           DWN
    Tagged      1-1-1-0-eth-0/bridge         DWN
    Tagged 4000  1-1-1-0-eth-4000/bridge      DWN
    Tg 1000/17   1-1-1-0-eth-1000/bridge      DWN
             50 1-1-2-0-eth/bridge           DWN
            100 1-1-40-0-efmbond/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
1-1-4-0-eth                  11920K 318K   197     6883K  0      0      0
1-1-3-0-eth                  0      0      0       0      0      0      0
1-1-1-0-eth-0                0      0      0       0      0      0      0
1-1-1-0-eth-4000             0      0      0       0      0      0      0
1-1-1-0-eth-1000             0      0      0       0      0      0      0
1-1-2-0-eth                  0      0      0       0      0      0      0
1-1-40-0-efmbond             0      0      0       0      0      0      0

zSH> bridge stats vlan 4000
Interface                    Received Packets      Transmitted Packets
Name                         UCast  MCast  BCast   UCast  MCast  Bcast  Error
1-1-1-0-eth-4000             0      0      0       0      0      0      0
```

9 ADVANCED CONFIGURATION

This chapter covers the following advanced configuration of the EtherXtend:

View EtherXtend statistics

The following table provides a list of commands to provide statistics related to the EtherXtend.

Table 19: Statistics commands

Command	Description
show efm-stats	Displays EFM statistics.
eth-oam stats	Displays OAM statistics.
show pme-stats	Displays PME statistics.

```
zSH> show efm-stats
efmCuPAFSupported:-------->   true  false
efmCuPeerPAFSupported:---->   unknown  true  false
efmCuPAFCapacity:--------->   {1 - 32}
efmCuPeerPAFCapacity:----->   {0 - 32}
efmCuFltStatus:----------->   {0 - 0}
efmCuPortSide:------------>   subscriber  office  unknown
efmCuNumPMEs:------------->   {0 - 32}
efmCuPAFInErrors:--------->   {0 - 0}
efmCuPAFInSmallFragments:->   {0 - 0}
efmCuPAFInLargeFragments:->   {0 - 0}
efmCuPAFInBadFragments:--->   {0 - 0}
efmCuPAFInLostFragments:-->   {0 - 0}
efmCuPAFInLostStarts:----->   {0 - 0}
efmCuPAFInLostEnds:------->   {0 - 0}
```

```
efmCuPAFInOverflows:------>   {0 - 0}

zSH> **eth-oam stats**
*************** dot3OamStatsTable for interface 1-1-201-0/efmbond ***************
        Information Tx                   4
        Information Rx                   4
        UniqueEventNotification Tx       0
        UniqueEventNotification Rx       0
        DuplicateEventNotification Tx    0
        DuplicateEventNotification Rx    0
        LoopbackControl Tx               0
        LoopbackControl Rx               0
        VariableRequest Tx               0
        VariableRequest Rx               0
        VariableResponse Tx              0
        VariableResponse Rx              0
        OrgSpecific Tx                   0
        OrgSpecific Rx                   0
        UnsupportedCodes Tx              0
       UnsupportedCodes Rx              0
       FramesLostDueToOam               0

zSH> **show pme-stats**
efmCuPmeSubTypesSupported:->   {0 - 0}
efmCuPmeOperStatus:-------->   up  downnotready  downready  init
efmCuPmeFltStatus:--------->   {0 - 0}
efmCuPmeOperSubType:------->   ieee2basetlo  ieee2basetlr  ieee10passtso
ieee10passtsr
efmCuPmeOperProfile:------->   {0 - 255}
efmCuPmeSnrMgn:------------>   {-127 - 65535}
efmCuPmePeerSnrMgn:-------->   {-127 - 65535}
efmCuPmeLineAtn:----------->   {-127 - 65535}
efmCuPmePeerLineAtn:------->   {-127 - 65535}
efmCuPmeTCCodingErrors:---->   {0 - 0}
efmCuPmeTCCrcErrors:------->   {0 - 0}
```

EtherXtend bond group statistics

This section describes EtherXtend bond group statistics including:

- View bond group statistics, page 117
- View alarm activity, page 118
- View individual member of bond group statistics, page 119
- Set alarm thresholds, page 120

View bond group statistics

The EtherXtend and other bonding capable devices provide the **bond stats** *interface/type* command to display both the status of the bond group and the status of each individual link in the bond group and to provide statistics for the bond group. A bond group is the aggregate of individual links on a device connected to the same CPE that provides a higher bandwidth than individual links can provide.

To view the statistics for an EtherXtend bond group enter **bond stats** *interface/type*:

```
zSH> bond stats 1-1-99-0/efmbond
****************** Bond group statistics ******************
                      Group Info
        Slot     GrpId      Interface Name
          1        99      1-1-99-0/efmbond
        AdminStatus      OperStatus      Bandwidth      Last Change
          UP               UP            45568000       0.00:06:09
        Threshold Alarm Config
          disabled
                   Group Members
        Port       Interface Name  AdminStatus      OperStatus      Bandwidth
          3        1-1-3-0/shdsl     UP               UP              5696000
          2        1-1-2-0/shdsl     UP               UP              5696000
          1        1-1-1-0/shdsl     UP               UP              5696000
          8        1-1-8-0/shdsl     UP               UP              5696000
          7        1-1-7-0/shdsl     UP               UP              5696000
          6        1-1-6-0/shdsl     UP               UP              5696000
          5        1-1-5-0/shdsl     UP               UP              5696000
          4        1-1-4-0/shdsl     UP               UP              5696000
                Statistics (Received)
        Octets                            480675689
        Ucast                             219070183
        Mcast                             0
        Bcast                             96
        Discards                          0
        Errors                            8649
                Statistics (Transmitted)
        Octets                            1370350049
        Ucast                             220924681
        Mcast                             129866
        Bcast                          58
        Discards                       0
```

Check the output of the Bandwidth column in the Group Info section to view the aggregate train rate of the bond group. This aggregate rate can change depending on the status of the individual links in the bond group as shown in the Group members section.

In the example below, 1-1-6-0/shdsl went down dropping the bandwidth to 0 dropping the bandwidth of the bond group interface 1-1-99-0/shdsl.

```
SH> bond stats 1-1-99-0/efmbond
****************** Bond group statistics ******************
                      Group Info
        Slot     GrpId      Interface Name
          1        99      1-1-99-0/efmbond
        AdminStatus      OperStatus      Bandwidth      Last Change
          UP               UP            39872000       0.00:09:36
        Threshold Alarm Config
          disabled
                   Group Members
        Port      Interface Name  AdminStatus      OperStatus      Bandwidth
          3        1-1-3-0/shdsl     UP               UP             5696000
          2        1-1-2-0/shdsl     UP               UP             5696000
          1        1-1-1-0/shdsl     UP               UP             5696000
          8        1-1-8-0/shdsl     UP               UP             5696000
          7        1-1-7-0/shdsl     UP               UP             5696000
          6        1-1-6-0/shdsl     UP               DOWN                 0
          5        1-1-5-0/shdsl     UP               UP             5696000
          4        1-1-4-0/shdsl     UP               UP             5696000
                Statistics (Received)
        Octets                          480675689
        Ucast                           219070183
        Mcast                           0
        Bcast                           96
        Discards                        0
        Errors                          8649
                Statistics (Transmitted)
        Octets                          1370394534
        Ucast                           220924681
        Mcast                           130519
        Bcast                           58
        Discards                        0
```

View alarm activity

To view alarm activity enter **alarm show**:

```
zSH> alarm show
************    Central Alarm Manager    ************
        ActiveAlarmCurrentCount        :3
        AlarmTotalCount                :9
        ClearAlarmTotalCount           :6
        OverflowAlarmTableCount         :0
ResourceId                AlarmType                 AlarmSeverity
----------                ---------                 -------------

1-1-2-0/eth               linkDown                  critical
1-1-3-0/eth               linkDown                  critical

1-1-99-0/efmbond          threshold_alarm_active    minor
```

The output shows a minor alarm on the 1-1-99-0/efmbond interface.

This change may be important when a loss of the aggregate rate affects service. In that case you can set a parameter in the EtherXtend's alarm-config profile to send a trap when the rate drops below a certain level. See *Set alarm thresholds* on page 120.

View individual member of bond group statistics

If you need to view the statistics of an individual member of a bond group, enter **dslstat** *interface/type*:

```
zSH> dslstat 1-4-1-0/shdsl
General Stats:
-------------
AdminStatus................................UP
LineStatus.................................DATA
Line uptime (DD:HH:MM:SS)..................0:04:35:14
DslUpLineRate (bitsPerSec).................5696000
DslDownLineRate (bitsPerSec)...............5696000
DslMaxAttainableUpLineRate (bitsPerSec)......5696000
DslMaxAttainableDownLineRate (bitsPerSec)....5696000
Out Octets.................................4104990247
Out Pkts/Cells.............................127962354
Out Discards...............................0
Out Errors.................................0
In Octets..................................1189520311
In Pkts/Cells..............................215674806
In Discards................................0
In Errors..................................5104364
DSL Physical Stats:
------------------
DslLineSnrMgn (tenths dB)..................180
DslLineAtn (tenths dB).....................0
DslCurrOutputPwr (tenths dB)...............0
LOFS.......................................0
LOLS.......................................4294967295
LOSS.......................................4294967295
ESS........................................46450
CRC Errors.................................8
Inits......................................153
zSH>
```

Set alarm thresholds

Bonding capable devices provide an alarm-config profile to enable traps and alarms to be sent after meeting certain criteria. You can monitor the bond group bandwidth and have traps and alarms sent whenever the bandwidth drops below a particular threshold for a particular holdtime.

Enter **get alarm-config** *interface/type* to view the alarm-config profile:

```
zSH> get alarm-config 1-1-99-0/efmbond
alarm-config  1-1-99-0/efmbond
bit-rate-threshold: ---------->  {disabled}:
bit-rate-threshold-value: ---->  {0}:
bit-rate-threshold-holdtime: ->  {0}:
status-trap-enable: ---------->  {enabled}:
admin-up-alarm: -------------->  {disabled}:
```

Table 20 describes the alarm-config profile parameters.

Table 20: EtherXtend alarm-group profile

Parameter	Function
bit-rate threshold	Enabled turns this feature on. Disabled turns this feature off.
bit-rate-threshold-value	The bandwidth of the bond group rate, set in Kbps, that generates a trap and alarm when the rate drops below this level.
bit-rate-threshold-holdtime	The time that the bandwidth must remain above or below the bit-rate-threshold-value before a trap is sent and an alarm is set or cleared. The purpose of this parameter is to keep down jitter.
status-trap-enable	Enabled turns the feature on to send a trap indicating the status of the interface. Disabled turns the feature off. Default is enabled. For ZMS.
admin-up-alarm	For MALC.

You can set the threshold alarm by first enabling the bit-rate-threshold parameter, then entering a bit-rate value in Kbps in the bit-rate-threshold-value parameter field for the bond group.

The status-trap-enable parameter defaults to enabled. Enter **update alarm-config** *interface/type* as follows:

```
zSH> update alarm-config 1-1-99-0/efmbond
alarm-config  1-1-99-0/efmbond
Please provide the following: [q]uit.
bit-rate-threshold: ---------->  {disabled}: enabled
bit-rate-threshold-value: ---->  {0}: 45000
```

```
bit-rate-threshold-holdtime: ->  {0}:
status-trap-enable: ---------->  {enabled}:
admin-up-alarm: -------------->  {disabled}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s Record updated.
```

If the aggregate bandwidth of the links in the bond group drops below the bit-rate-threshold-value set in the alarm-config and remains below this value for the bit-rate-threshold-holdtime set in the alarm-config profile, then a trap and an alarm will be sent.

Display EFM profile parameters

EFM supports three commands to display output for valid parameters affected by EFM. The commands are:

- **show efm-bond**
- **show efm-port**
- **show ether-oam**

Display EFM information

To display EFM parameter information, enter **show efm-bond** and **show efm-port**:

```
zSH> show efm-bond
config:------->   auto  manual
dynamic-link:->   enabled  disabled

zSH> show efm-port
efmCuPAFAdminState:---------------->   enabled  disabled
efmCuPAFDiscoveryCode:------------->   {260}
efmCuAdminProfile:----------------->   {8}
efmCuTargetDataRate:--------------->   {1 - 999999}
efmCuTargetWorstCaseSnrMgn:-------->   {-10 - 21}
efmCuThreshLowBandwidth:----------->   {0 - 100000}
efmCuLowBandwidthEnable:----------->   true  false
efmCuTargetCurrentConditionMode:--->   true  false
efmCuTargetCurrentConditionSnrMgn:->   {-10 - 21}
efmCuTargetWorstCaseMode:---------->   true  false
```

To display OAM event configuration, enter **show ether-oam**:

```
zSH> show ether-oam
mode:---------------------------->   passive  active
loopbackStatus:------------------>   noloopback
initiatingloopback  remoteloopback  terminatingloopback  l
```

```
ocalloopback  unknown
ignoreLoopbackCommands:----------->   ignore  process
symbolPeriodSizeHi:--------------->   {0 - 0}
symbolPeriodSizeLo:--------------->   {0 - 0}
symbolPeriodErrorThresholdHi:----->   {0 - 0}
symbolPeriodErrorThresholdLo:----->   {0 - 0}
symbolPeriodErrorNotifyEnable:---->   true  false
framePeriodSize:------------------>   {0 - 0}
framePeriodErrorThreshold:-------->   {0 - 0}
framePeriodErrorNotifyEnable:----->   true  false
intervalWindowSize:--------------->   {0 - 0}
intervalFrameErrorThreshold:------>   {0 - 0}
intervalFrameErrorNotifyEnable:--->   true  false
intervalErrorSummaryWindow:------->   {100 - 9000}
intervalErrorSummaryThreshold:---->   {1 - 900}
intervalErrorSummaryNotifyEnable:->   true  false
dyingGaspEnable:------------------>   true  false
criticalEventEnable:-------------->   true  false
```

Perform EFM updates

The **update** command updates an existing efm-oam profile for the unit.

Update the EFM port profile

To update the EFM port profile using 1-1-1-1 as an example, enter **update efm-port**:

```
zSH>update efm-port 1-1-1-1
profileVersion: ---------->  {1.0}:  ** read-only **
efmCuPAFAdminState: ------>  {enabled}:
efmCuPAFDiscoveryCode: --->  {}:
efmCuAdminProfile: ------->  {0x01}:
efmCuTargetDataRate: ----->  {50000}:
efmCuTargetWorstCaseSnrMgn: ------->  {6}:
efmCuThreshLowBandwidth: ->  {0}:
efmCuLowBandwidthEnable: ->  {false}:
efmCuThreshLowBandwidth: ->  {0}
efmCuLowBandwidthEnable: ->  {false}
efmCuTargetCurrentConditionMode: ---> {false}
efmCuTargetCurrentConditionSnrMgn: -> {6}
efmCuTargetWorstCaseMode: ----------> {true}
```

The efmCurrentConditionMode parameter allows the line to sync using conditions that it *currently* sees on the line at the time of training and takes into consideration *only* the users/disbtubers (noise) seen at that particular moment in time. Because of this, it is recommended that you set this parameter to true only in lab situations or under other special scenarios that would not mind if the line becomes unstable or the link retrains. Instability can occur because over time errors can increase causing retransmissions and lower throughput or cause a retrain of the link resulting in a lower traffic rate.

The efmWorstCaseMode allows the line to sync assuming line conditions are the worst they can be by assuming that there are forty nine users/disburbers even if they do not currently exist. Setting efmWorstCaseMode to true is the most stable setting because the line’s SNR and stability are not effected when other users come onto the line. The line typically trains to a lower rate to allow for any SNR loss when these changes occur.

For n2n bonding, but *not* EFM bonding, you can set both the efmCurrentConditionMode and the efmWorstCaseMode to true. This results in a mixed mode that sets a more liberal train rate than efmWorstCaseMode, but a more conservative train rate than in efmCurrentConditionMode.

Update the EFM bond

To update the efm-bond profile using 1 as an example, enter **update efm-bond**:

```
zSH>update efm-bond
dynamic link: -----------------> {enabled}
```

Modify EFM port interfaces

Perform a list command

EFM ports are automatically created. To view EFM port profiles, enter **list** to view what EFM profiles are available. The following example shows a list of EFM specific EFM profiles that are available.

```
zSH> list
efm-bond: ifIndex
efm-port: ifIndex
efm-stats: ifIndex
```

EFM port default

To display the EFM port parameters in their default state enter **get efm-port** *interface/type*:

```
zSH> get efm-port 1-1-1-0/shdsl
efm-port  1-1-1-0/shdsl
efmCuPAFAdminState: ---------------->  {enabled}
efmCuPAFDiscoveryCode: ------------->  {}
efmCuAdminProfile: ----------------->  {0x01}
efmCuTargetDataRate: --------------->  {50000}
efmCuTargetWorstCaseSnrMgn: -------->  {1}
efmCuThreshLowBandwidth: ----------->  {0}
efmCuLowBandwidthEnable: ----------->  {false}
```

```
efmCuTargetCurrentConditionMode: --->  {false}
efmCuTargetCurrentConditionSnrMgn: ->  {6}
efmCuTargetWorstCaseMode: ---------->  {true}
```

Modify an EFM port

To modify an EFM port, enter **update efm-port** *interface/type* to display the profile parameters.

```
zSH> update efm-port 1-1-1-0/shdsl
efm-port  1-1-1-0/shdsl
Please provide the following: [q]uit.
efmCuPAFAdminState: ---------------->  {enabled}:
efmCuPAFDiscoveryCode: ------------->  {}:
efmCuAdminProfile: ----------------->  {0x01}:
efmCuTargetDataRate: --------------->  {50000}:
efmCuTargetWorstCaseSnrMgn: -------->  {1}:
efmCuThreshLowBandwidth: ----------->  {0}:
efmCuLowBandwidthEnable: ----------->  {false}:
efmCuTargetCurrentConditionMode: --->  {false}:
efmCuTargetCurrentConditionSnrMgn: ->  {6}:
efmCuTargetWorstCaseMode: ---------->  {true}:
```

Create a new EFM bond

To create a new EFM bond group, enter **new efm-bond** to display the profile parameters.

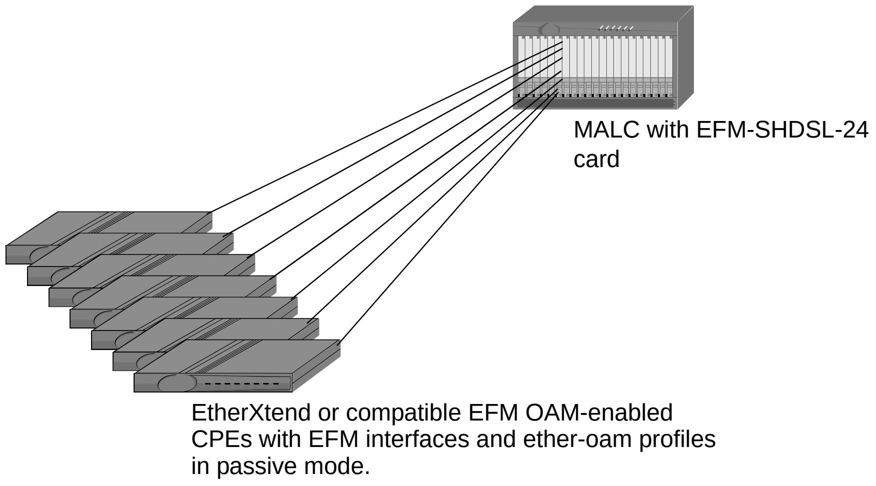
```
zSH>new efm-bond 1-1-1-0/shdsl
Please provide the following: [q]uit.
config: ------->  {auto}:
dynamic-link: ->  {enabled}:dynamic
```

802.3ah EFM OAM

EFM OAM uses an in-band link layer OAM packet exchange between MALC EFM interfaces and OAM capable CPEs such as the EtherXtend. In this case the EtherXtend functions as a remote peer to provide discovery, OAM pdu statistics, and peer information.

When OAM is configured on a MALC EFM interface in active mode, the discovery process initiates a search for a peer OAM-enabled EtherXtend configured in passive mode and physically connected to the MALC EFM port of that interface. If the discovery process does not find an EtherXtend peer, the discovery process begins again after a five second pause and continues until a peer OAM-enabled EtherXtend is found.

Release 1.14.1 introduces OAM support on MALC-EFM-SHDSL-24, MALC-EFM-SHDSL-24NT, MALC-EFM-SHDSL-24NTP card interfaces connected to an EtherXtend.


MALC with EFM-SHDSL-24 card
EtherXtend or compatible EFM OAM-enabled CPEs with EFM interfaces and ether-oam profiles in passive mode.

OAM modes

EFM/CPEs incorporating the OAM sublayer support active and/or passive mode. When OAM is enabled, a EFM/CPE capable of both active and passive mode will select either active or passive. The following table indicates the behavior of active and passive mode EFM/CPEs.

Table 21: Active and passive mode

Capability	Active EFM/CPE	Passive EFM/CPE
Initiates OAM discovery process.	Yes	No
Reacts to OAM discovery process initiation.	Yes	Yes
Required to send information OAMPDUs	Yes	Yes
Permitted to send event notification OAMPDUs	Yes	Yes
Permitted to send variable request OAMPDUs	Yes	No
Permitted to send variable response OAMPDUs	Yes	Yes
Permitted to send loopback control OAMPDUs	Yes	No

Table 21: Active and passive mode

Capability	Active EFM/CPE	Passive EFM/CPE
Reacts to loopback control OAMPDUs	Yes	
Permitted to send organization specific OAMPDUs	Yes	Yes
Requires the per EFM/CPE to be in active mode	Yes	Yes

Active mode

EFM/CPEs configured in active mode initiate the exchange of information OAMPDUs as defined by the discovery state. Once the discovery process completes, active EFM/CPEs are permitted to send any OAMPDU while connected to a remote OAM peer entity in active mode.

Active EFM/CPEs operate in a limited capacity if the remote OAM entity is operating in passive mode. Active devices should not respond to OAM remote loopback commands and variable requests from a passive peer.

Passive mode

EFM/CPEs configured in passive mode do not initiate the discovery process. Passive EFM/CPEs react to the initiation of the discovery process by the remote EFM/CPE. This eliminates the possibility of passive to passive links.

OAM commands

This sections describes commands used to add OAM functionality to a bridge interface, and commands to view information on that interface. The OAM commands are:

- **eth-oam add**
- **eth-oam stats**
- **eth-oam show**

Add OAM to a bridge interface

Use **eth-oam add** to add OAM functionality to a bridge interface.

Adding OAM functionality to a bridge interface

1 To add OAM functionality to a EtherXtend bridge interface, enter **show bridge** to view the available interfaces:

```
zSH> bridge show
Typ VLAN        Bridge                        St  Table Data
-----------------------------------------------------------------------------
upl Tagged      1-1-201-0-efmbond-0/bridge    UP  S Global default [U: 3600 sec, M:
                                              150 sec, I: 0 sec]
dwn           2 1-1-1-0-eth/bridge            UP
```

2 Enter the **eth-oam add** command to add OAM functionality to the bridge interface.

```
zSH> eth-oam add 1-1-201-0/efmbond
```

Display OAM statistics

Displaying eth-oam statistics

Enter **eth-oam stats** to display the OAM statistics associated with a bridge interface:

```
zSH> eth-oam stats
*************** dot3OamStatsTable for interface 1-1-201-0/efmbond ***************
        Information Tx                    4
        Information Rx                    4
        UniqueEventNotification Tx        0
        UniqueEventNotification Rx        0
        DuplicateEventNotification Tx     0
        DuplicateEventNotification Rx     0
        LoopbackControl Tx                0
        LoopbackControl Rx                0
        VariableRequest Tx                0
        VariableRequest Rx                0
        VariableResponse Tx               0
        VariableResponse Rx               0
        OrgSpecific Tx                    0
        OrgSpecific Rx                    0
        UnsupportedCodes Tx               0
        UnsupportedCodes Rx               0
        FramesLostDueToOam                0
```

Show the OAM profile

Enter **eth-oam show** to display the OAM profile.

```
zSH> eth-oam show
****************** OAM Profile for interface 1-1-201-0/efmbond ******************
        OperationalState                       Operational
        OamMode                                passive
        MaxOamPduSize                          1518
        ConfigurationRevision                  1
        FunctionsSupported                     None
        OamLoopbackStatus                      no Loopback
        OamLoopbackIgnoreRx                    ignore
```

```
ErroredFrame Window                  10
ErroredFrame Threshold               1
ErroredFrame Notify                  enabled
ErroredFramePeriod Window            4294967295
ErroredFramePeriod Threshold         1
ErroredFramePeriod Notify            disabled
ErroredFrameSecondsSummary Window    100
ErroredFrameSecondsSummary Threshold   1
ErroredFrameSecondsSummary Notify    disabled
DyingGaspEnable                      disabled
CriticalEventEnable                  disabled
```

Configuring OAM support

The OAM interface is defined by an **ether-oam** profile that specifies the options for active/passive mode, loopback, and notification for events. By default, OAM is disabled on all MALC uplink and Ethernet interfaces.

To configure OAM features:

1. Create a new OAM profile for the desired Ethernet interface. By default, this profile is in passive mode with loopback disabled.

   This example configures Ethernet OAM in active mode on EFM bond group 1-4-50-0/efmbond on a EFM-SHDSL-24 card in slot 4.

```
zSH> eth-oam add 1-4-50-0/efmbond active
```

2. Create a new OAM profile for the desired EtherXtend interface. By default, this profile is in passive mode with loopback disabled.

   This example configures Ethernet OAM in passive mode on EFM bond group 1-1-40-0/efmbond on the peer EtherXtend.

```
zSH> eth-oam add 1-1-40-0/efmbond passive
```

3. Enter commands to modify and display OAM parameters.

   The **eth-oam modify** command provides access to configurable settings in the ether-oam profile.

   The **eth-oam show** command displays configured OAM settings.

   The **eth-oam stats** command displays OAM statistics for a specified physical interface or bond group or all OAM interfaces.

eth-oam add

Configures and enables OAM interface on a physical interface.

Syntax **eth-oam *interface/type* [active | passive]**

Options **interface/type**
Name and type of the physical interface or bond group.

active
Sets OAM to active mode on this interface. The default is passive.

passive
Sets OAM to passive mode on this interface. The default is passive.

eth-oam delete

Deletes and disables the OAM configuration on the specified physical interface. This command does not delete any other configurations on this interface such as bond groups and bridge interfaces.

Syntax **`eth-oam delete`** ***`interface/type`***

Options **interface/type**
Name and type of the physical interface or bond group.

eth-oam modify

Modifies a configured eth-oam interface.

Syntax **`eth-oam modify`** ***`interface/type`*** **`[active | passive]`**

Options **interface/type**
Name and type of the physical interface or bond group.

eth-oam show

Displays configured OAM parameters for the specified interface. If no interface is specified, configured OAM parameters are displayed for all OAM enabled interfaces.

Syntax **`eth-oam show`** ***`interface/type`*** **`[peer]`**

Options **interface/type**
Name and type of the physical interface or bond group.

peer
Displays the learned configuration information of the peer for the given interface. Includes peer MAC address, peer vendor OUI, peer vendor unique info, peer mode, peer max OAM PDU size, peer configuration revision, peer supported functions.

eth-oam stats

Displays OAM statistics for the specified interface. If no option is specified, statistics are displayed for all OAM interfaces.

Syntax **`eth-oam stats`** ***`interface/type`***

Options **interface/type**
Name and type of the physical interface or bond group.

10 IP SERVICE LEVEL AGREEMENT

This chapter covers IP Service Level Agreement (IPSLA) for the EtherXtend.

Overview

The IP Service Level Agreement (IPSLA) feature assists service providers and network operators with enforcing and monitoring access network connections and performance. IPSLA uses ICMP Ping messages over configured IPSLA paths to track Round Trip Times (RTTs) and ECHO REQs/ RSPs between initiator and responder devices to determine network performance and delays. Typically, one initiator device is used to monitor other responder devices in the network. A maximum of 32 IPSLA paths can be configured per MALC and 4 IPSLA paths per EtherXtend.

IPSLA

Initiator devices must be running IPSLA to request data for a responder device. Responder devices must be configured to respond to ICMP messages. Responder devices not running IPSLA display limited statistical data and functionality.

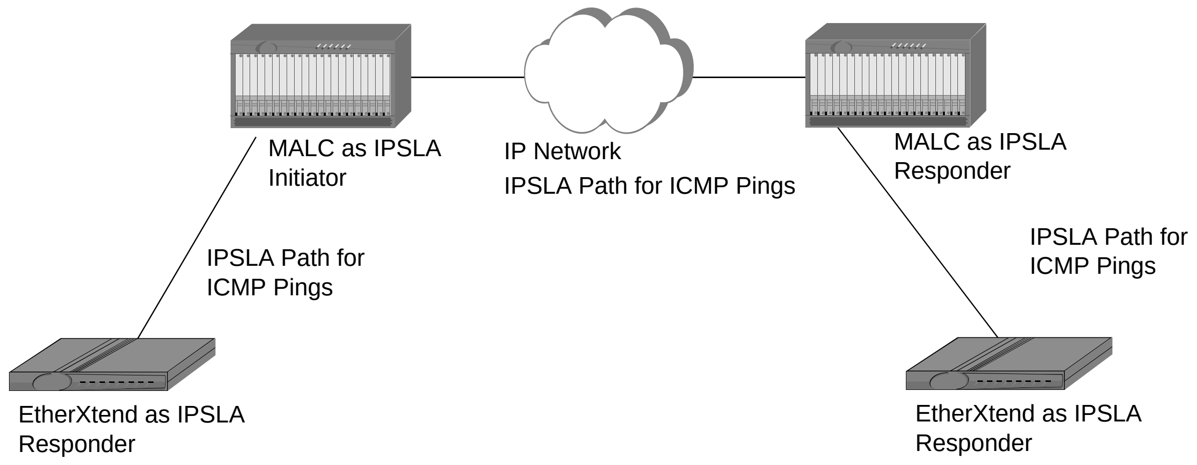
Note: Networks must support CoS queues and DSCP to provide valid per CoS statistics. Otherwise, all statistics are sent to the default CoS queue.

Default CoS-actions are assigned to each CoS queue so threshold crossing alarms can be configured to generate system alarms when thresholds are crossed for uptime, latency, jitter, and packet size.

Data based on received/sent packets and train rates is collected and displayed as real-time statistics for the current 15 minute interval as well as over 96 15-minute intervals for 24 hour historical statistics.

By default, IPSLA is disabled on all EtherXtend, MALC card ports and other SLMS devices.

Figure 19: IPSLA


MALC as IPSLA Initiator
IP Network
IPSLA Path for ICMP Pings
MALC as IPSLA Responder
IPSLA Path for ICMP Pings
IPSLA Path for ICMP Pings
EtherXtend as IPSLA Responder
EtherXtend as IPSLA Responder

Configuring IPSLA

IPSLA requires the following configuration steps:

- Set ipsla-global settings to enable device state and optionally set polling interval
- Create ICMP path between devices
- Optionally modify COS actions for the desired COS queues
- Optionally modify COS map for Diff Server Control Point (DSCP) mappings

To configure IPSLA:

IPSLA global parameters shown in Table 22 determine if the feature is active, and if globally enabled, determine the time interval between outgoing IPSLA Pings.

Table 22: IPSLA global

Parameter	Value	Description
state	enabled/disables	IPSLA global feature activity
pollseconds	60...3600	The interval between outgoing IPSLA pings in seconds.

1 Enable IPSLA and set IPSLA ping interval.

a View the global IPSLA settings by entering **ipsla show global.**

```
zSH> ipsla show global
state: ------->  {disabled}
pollSeconds: ->  {60}
```

The polling interval (60 to 3600 seconds) is used for real-time and historical statistics.

b Enter **iplsa modify global state** *value* **pollseconds** *value* to enable IPSLA and set the pollseconds interval.

```
zSH> ipsla modify global state enabled pollseconds 120
```

c Verify the IPSLA settings.

```
zSH> ipsla show global
state: ------->  {enabled}
pollseconds: ->  {120}
```

2 Create a ICMP path between devices. The device on which this command is entered becomes the initiator device, while the device for which an IP address is entered becomes the responder device. Typically, one initiator device can be used to monitor other responder devices in the network over a maximum of 32 MALC and 4 EtherXtend IPSLA paths per device.

a Enter the **ipsla add path** command to create the path.

```
zSH> ipsla add path ipaddress 172.16.78.11
```

b Enter **ipsla show path** command to verify that the IP address was added.

```
zSH> ipsla show path
Path configuration for ipAddress: 172.16.78.11
forwarding: ->  {disabled}
state: ------>  {enabled}
```

c Modify the path using the IPSLA **modify path** command. This example disables the static path on device 192.168.254.17.

```
zSH> ipsla modify path ipaddress 192.168.254.17 state disabled
```

d Delete a path using the IPSLA **delete** command.

```
zSH> ipsla delete path ipaddress 192.168.254.17
```

Note: Disabling or deleting the path or globally disabling the IPSLA feature will reset historical data.

3 Modify the default CoS actions to specify the response and threshold behavior for each CoS Action Index (1-8). These CoS actions map respectively to the CoS queues (0-7) as shown in Table 23. The following CoS actions are defined by default.

Table 23: CoS action index map to CoS queues

Default Name	CoS Action Index	CoS Queue
Default	1	0
AFClass 1	2	1

Table 23: CoS action index map to CoS queues (Continued)

Default Name	CoS Action Index	CoS Queue
AFClass 2	3	2
AFClass 3	4	3
AFClass4	5	4
Cos-5	6	5
ExpFwd	7	6
NetwCtrl	8	7

Each CoS action contains parameters as shown in Table 24:

Table 24: CoS action parameters

Parameter	Description	Default
Name	Name of the IPSLA CoS action, up to 9 characters in length.	(1) Default, (2) AFClass1, (3) AFClass2, (4) AFClass3, (5) AFClass4, (6) Cos-5, (7) ExpFwd, (8) NetwCtrl
Traps	Specifies whether a trap is issued when any SLA performance error threshold within this CoS is crossed.	disabled/enabled
Timeouts	Specifies the number of consecutive missed IP SLA responses within this CoS before a zhoneIpSLATimeoutTrap is issued.	1...20
Timeout Clear	Specifies the number of consecutive IPSLA responses within this CoS which must be received before the timeout error condition is cleared.	1 sample
Latency	Specifies the 15 sample average roundtrip latency value which must be exceeded within this CoS before a zhoneIpSLALatencyTrap is issued.	50...10000 milliseconds
Latency Clear	Specifies the number of consecutive IPSLA latency samples for which the 15 sample average roundtrip latency must be below the configured SLA latency error threshold within this CoS before the latency error condition is cleared.	1 sample
Jitter	Specifies the 15 sample roundtrip jitter value which must be exceeded within this CoS before a zhoneIpSLAJitterTrap is issued.	50...10000 milliseconds
Jitter Clear	Specifies the number of consecutive IPSLA RTT samples for which the 15 sample roundtrip jitter must be below the configured SLA jitter error threshold within this CoS before the jitter error condition is cleared.	1 sample
Packetsize	Specifies the minimum IPSLA Ping packet size in bytes. The range is 64 thru 2048 if the target IP device is running IPSLA, 64 thru 512 otherwise.	64...2048 bytes

a Display the settings for an individual CoS action. The CoS Action determines IPSLA performance thresholds per CoS and how or whether to react to threshold crossings. The IPSLA Ping packet size may also be set in CoS Action.

```
zSH> ipsla show cos-action cosactionindex 1
Cos Action Configuration for cosActionIndex: 1:
name: ------->  {Default}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}
```

Or

b Display the settings for all CoS actions (1-8).

```
zSH> ipsla show cos-action
Cos Action Configuration for cosActionIndex: 1:
name: ------->  {Default}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}

Cos Action Configuration for cosActionIndex: 2:
name: ------->  {AFClass1}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}

Cos Action Configuration for cosActionIndex: 3:
name: ------->  {AFClass2}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}

Cos Action Configuration for cosActionIndex: 4:
name: ------->  {AFClass3}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}

Cos Action Configuration for cosActionIndex: 5:
name: ------->  {AFClass4}
traps: ------>  {disabled}
```

```
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}

Cos Action Configuration for cosActionIndex: 6:
name: ------->  {Cos-5}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}

Cos Action Configuration for cosActionIndex: 7:
name: ------->  {ExpFwd}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}

Cos Action Configuration for cosActionIndex: 8:
name: ------->  {NetwCtrl}
traps: ------>  {disabled}
timeOuts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetSize: ->  {64}
```

c Modify a cos-action by specifying the desired parameters to change in the command line. This example enables traps for cosActionIndex 1.

```
zSH> ipsla modify cos-action cosactionindex 1 traps enabled
```

d Verify the change by entering **ipsla show.**

```
zSH> ipsla show cos-action cosactionindex 1
Cos Action Configuration for cosactionindex: 1:
name: ------->  {Default}
traps: ------>  {enabled}
timeouts: --->  {3}
latency: ---->  {10000}
jitter: ----->  {10000}
packetsize: ->  {64}
```

e Modify another cos-action, in this case the latency threshold to 5000 milliseconds.

```
zSH> ipsla modify cos-action cosactionindex 1 latency 5000
```

f Verify the change by entering **ipsla show**.

```
zSH> ipsla show cos-action cosactionindex 1
```

```
Cos Action Configuration for cosactionindex: 1:
name: ------->  {Default}
traps: ------>  {enabled}
timeouts: --->  {3}
latency: ---->  {5000}
jitter: ----->  {10000}
packetsize: ->  {64}
```

4 Change alarm-clearing thresholds, if necessary.

The latency-clear parameter sets the criteria for when a trap (alarm) is sent based on how often a sample is taken.

a View the current settings with **get ipsla-cos-act** *index number*.

```
zSH> get ipsla-cos-act 1
ipsla-cos-act  1
name: ---------->  {Default}
traps: --------->  {enabled}
timeouts: ------>  {3}
timeout-clear: ->  {1}
latency: ------->  {5000}
latency-clear: ->  {1}
jitter: -------->  {10000}
jitter-clear: -->  {1}
metrics: ------->  {enabled}
packet-size: --->  {64}
```

b Change the alarm-clearing threshold, in this case for latency-clear, with **update ipsla-cos-act** *index number*.

```
zSH> update ipsla-cos-act 1
ipsla-cos-act  1
Please provide the following: [q]uit.
name: ---------->  {Default}:
traps: --------->  {enabled}:
timeouts: ------>  {3}:
timeout-clear: ->  {1}:
latency: ------->  {5000}:
latency-clear: ->  {1}: 4
jitter: -------->  {10000}:
jitter-clear: -->  {1}:
metrics: ------->  {enabled}:
packet-size: --->  {64}:
....................
Save changes? [s]ave, [c]hange or [q]uit: s
Record updated.
```

c Verify the change by entering **get ipsla-cos-act 1.**

```
zSH> get ipsla-cos-act 1
ipsla-cos-act  1
name: ---------->  {Default}
traps: --------->  {enabled}
timeouts: ------>  {3}
```

```
      timeout-clear: ->  {1}
      latency: ------->  {5000}
      latency-clear: ->  {4}
      jitter: -------->  {10000}
      jitter-clear: -->  {1}
      metrics: ------->  {enabled}
      packet-size: --->  {64}
```

5 Configure the desired COS maps to modify the default DSCP to COS Action Index mappings. By default, DSCP are mapped to COS Action Index entries based of RFC 2599. The following tables shows the default mappings. A COS Action Index of 0 indicates that the DSCP is not used.

DSCP	COS Action Index
1	8
11, 13, 15	7
19, 21, 23,	6
27, 29, 31	5
35, 37, 39	4
41	3
47	2
49, 57	1
2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 17, 18, 20, 22, 24, 25, 26, 28, 30, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 48, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 61, 62, 63, 64	0

Display the CoS map for an individual CoS action or for all CoS actions.

```
zSH> **ipsla show cos-map**
dscpIndex: 1    cosActionIndex: 1
dscpIndex: 2    cosActionIndex: 0
dscpIndex: 3    cosActionIndex: 0
dscpIndex: 4    cosActionIndex: 0
dscpIndex: 5    cosActionIndex: 0
dscpIndex: 6    cosActionIndex: 0
dscpIndex: 7    cosActionIndex: 0
dscpIndex: 8    cosActionIndex: 0
dscpIndex: 9    cosActionIndex: 0
dscpIndex: 10   cosActionIndex: 0
dscpIndex: 11   cosActionIndex: 2
dscpIndex: 12   cosActionIndex: 0
dscpIndex: 13   cosActionIndex: 2
dscpIndex: 14   cosActionIndex: 0
dscpIndex: 15   cosActionIndex: 2
dscpIndex: 16   cosActionIndex: 0
dscpIndex: 17   cosActionIndex: 0
```

```
dscpIndex: 18   cosActionIndex: 0
dscpIndex: 19   cosActionIndex: 3
Type A<CR> to print all, <CR> to continue, Q<CR> to
stop:
```

Specify the desired index values in the command line to change the mapping of the DSCP index to COS action index. This example changes the mapping of DSCP index 1 to CoS action index 7. Because the DSCP indexes (1-64) map respectively to the DSCP values (0-63), and the CoS action index (1-8) map respectively to the CoS queues (0-7), so this change also changes the mapping of DSCP value 0 to CoS queue 6.

```
zSH> ipsla modify cos-map dscpindex 1 cosactionindex 7
```

To clear a CoS map, specify the desired index values in the IPSLA command to delete the mapping of the DSCP index for the COS queue. This example clears the mapping of DSCP index 1 and resets it to the COS queue 0.

```
zSH> ipsla modify cos-map dscpindex 1 cosactionindex 0
```

6 Display real-time statistics for path or COS queue. Real-time statistics represent minimum, maximum, average, and current values over the current 15 minute polling period based on data collected for each polling intervals. For example, if the polling interval is configured for 60 seconds, the real-time statistics display the data compiled from the latest 15 60-second polling intervals contained in the current polling period.

Note: RTT values of 0 (zero) indicate a lack of data, while sub-millisecond RTTs are reported as 1.

These statistics can be displayed individually or collectively for a specified IP address or for all configured paths.

Note: When a card swact occurs, historical data does not failover and data for the15-minute interval during which the swact occurred may be lost.

```
zSH> ipsla stats path ipaddress 192.168.254.15
---------------+--------+--------+-+---------------+-+--------+-+--------+
               |        |        |A|               |C|        |I|   #    |
   Target IP   | Target | Target |C|    Source     |N| UpTime |/|  CoS   |
    Address    |  name  |  type  |T|      IP       |X| (secs) |R|mismatch|
---------------+--------+--------+-+---------------+-+--------+-+--------+
 192.168.254.15| Unknown| Unknown|Y|192.168.254.166|S|    4357|I|       0|

zSH> ipsla stats path
---------------+--------+--------+-+---------------+-+--------+-+--------+
               |        |        |A|               |C|        |I|   #    |
   Target IP   | Target | Target |C|    Source     |N| UpTime |/|  CoS   |
    Address    |  name  |  type  |T|      IP       |X| (secs) |R|mismatch|
---------------+--------+--------+-+---------------+-+--------+-+--------+
   172.16.78.11| Unknown| Unknown|Y|192.168.254.166|S|   14723|I|      30|
  192.168.254.15| Unknown| Unknown|Y|192.168.254.166|S|   14723|I|     171|
  192.168.254.17| Unknown| Unknown|Y|192.168.254.166|S|   14723|I|      30|
```

The table below explains the statistics for the configured paths.

Path Statistic	Description
Target IP Address	IP Address of the device which is at the other end of the path.
Target Name	Name of the remote device.
Target Type	Type of the remote device.
ACT	Availability status of the remote device.
Source IP	IP Address of the discovery source device.
CNX	Type of path either static or dynamic.
UpTime (secs)	Amount of time in seconds that elapsed since the last transition from Inactive to Active.
I/R	Role played by the local device in collection of latency and availability statistics. Initiator - Device that initiates the IPSLA ping packet used for statistics collection; Responder - Device that returns the IPSLA ping packet sent by the Initiator.
CoS Mismatch	Number of IPSLA ping packets received which indicate a mismatch between the Class Of Service (CoS) definitions at the remote unit and those of the source unit.

Display real-time CoS statistics individually or collectively by CoS action index, IP address or all CoS actions.

```
zSH> ipsla stats cos cosActionindex 1
---+-----------------+---+--------+--------+--------+--------+--------+
   |                 | A |        |        |        |        |        |
Cos|    Target IP    | C |  Last  |  Min   |  Avg   |  Max   |  Drop  |
Idx|     Address     | T |  RTT   |  RTT   |  RTT   |  Rtt   |  Resp  |
---+-----------------+---+--------+--------+--------+--------+--------+
 1 |      10.2.1.254 | Y |       0|       0|       0|       0|       2|
 1 |   172.24.94.254 | Y |       0|       0|       0|       0|       1|


zSH> ipsla stats cos ipaddress 10.2.1.254
---+-----------------+---+--------+--------+--------+--------+--------+
   |                 | A |        |        |        |        |        |
Cos|    Target IP    | C |  Last  |  Min   |  Avg   |  Max   |  Drop  |
Idx|     Address     | T |  RTT   |  RTT   |  RTT   |  Rtt   |  Resp  |
---+-----------------+---+--------+--------+--------+--------+--------+
 1 |      10.2.1.254 | Y |       0|       0|       0|       0|       2|
 2 |      10.2.1.254 | Y |       0|       0|       0|       0|       0|
 3 |      10.2.1.254 | Y |       0|       0|       0|       0|       0|
 4 |      10.2.1.254 | Y |      10|      10|      10|      10|       0|
 5 |      10.2.1.254 | Y |       0|       0|       0|       0|       0|
 6 |      10.2.1.254 | Y |       0|       0|       0|       0|       1|
 7 |      10.2.1.254 | Y |       0|       0|       0|       0|       1|
 8 |      10.2.1.254 | Y |       0|       0|       0|       0|       1|


zSH> ipsla stats cos
---+----------------+---+--------+--------+--------+--------+--------+
   |                | A |        |        |        |        |        |
Cos|    Target IP   | C |  Last  |  Min   |  Avg   |  Max   |  Drop  |
Idx|     Address    | T |  RTT   |  RTT   |  RTT   |  Rtt   |  Resp  |
---+----------------+---+--------+--------+--------+--------+--------+
 1 |    172.16.78.11 | Y |       0|      20|      20|      20|       2|
 2 |    172.16.78.11 | Y |       0|      10|      10|      10|       0|
 3 |    172.16.78.11 | Y |       0|      20|      20|      20|       0|
 4 |    172.16.78.11 | Y |       0|       0|       0|       0|       0|
 5 |    172.16.78.11 | Y |       0|     130|     130|     130|       0|
 6 |    172.16.78.11 | Y |       0|     340|     340|     340|       0|
 7 |    172.16.78.11 | Y |       0|       0|       0|       0|       1|
 8 |    172.16.78.11 | Y |       0|     380|     380|     380|       0|
 1 |  192.168.254.15 | Y |       0|       0|       0|       0|      11|
 2 |  192.168.254.15 | Y |       0|       0|       0|       0|       9|
 3 |  192.168.254.15 | Y |       0|       0|       0|       0|       9|
 4 |  192.168.254.15 | Y |       0|       0|       0|       0|       9|
 5 |  192.168.254.15 | Y |       0|       0|       0|       0|      10|
 6 |  192.168.254.15 | Y |       0|       0|       0|       0|       9|
Type A<CR> to print all, <CR> to continue, Q<CR> to stop: a
---+----------------+---+--------+--------+--------+--------+--------+
   |                | A |        |        |        |        |        |
Cos|    Target IP   | C |  Last  |  Min   |  Avg   |  Max   |  Drop  |
Idx|     Address    | T |  RTT   |  RTT   |  RTT   |  Rtt   |  Resp  |
---+----------------+---+--------+--------+--------+--------+--------+
 7 |  192.168.254.15 | Y |       0|       0|       0|       0|       9|
 8 |  192.168.254.15 | Y |       0|       0|       0|       0|       9|
 1 |  192.168.254.17 | Y |       0|      10|      10|      10|       1|
 2 |  192.168.254.17 | Y |       0|      20|      20|      20|       1|
 3 |  192.168.254.17 | Y |       0|       0|       0|       0|       0|
 4 |  192.168.254.17 | Y |       0|      10|      10|      10|       0|
 5 |  192.168.254.17 | Y |       0|      80|      80|      80|       0|
 6 |  192.168.254.17 | Y |       0|     350|     350|     350|       0|
 7 |  192.168.254.17 | Y |       0|     150|     150|     150|       0|
 8 |  192.168.254.17 | Y |       0|     280|     280|     280|       0|
```

The table below explains the CoS Action Index statistics.

COS Action Index Statistic	**Description**
CoS Index	Index number of the CoS Action Index.
Target IP Address	IP Address of the device which is at the other end of the path.
Last RTT	RTT reported in the most recent successful ping attempt.
Min RTT	Smallest RTT since this statistic was last cleared to a zero value.

COS Action Index Statistic	Description
Avg RTT	Average RTT since this statistic was last cleared to a zero value. Calculated as (RTT1 + RTT2 + RTT3 + …….+RTTn)/n where n equals the number of successful ping attempts since this statistic was last cleared to a zero value.
Max RTT	Largest RTT since this statistic was last cleared to a zero value.
Drop Resp	Number of failed pings since this statistic was last cleared to a zero value.

Display historical statistics individually or collectively based on IP address, CoS action index, and index value of a 15 minute interval. Historical statistics are displayed for the latest 24 hour period or a specified 15 minute interval within the latest 24 hour period.

Note: It is required to specify at least two of these three instancing values: cosactionindex, IP address, and index. For example, the combination of cosactionindex and IP address, or the combination of index and cosactionindex, or the combination of index and IP address.

For historical statistics, IPSLA averages values for the most recent 96 15-minute intervals and displays the minimum, maximum, average and current values in a table for a 24 hour summary.

```
zSH> ipsla stats history cosactionindex 1 ipaddress 172.16.78.11
---+---+-----------------+-----------------+-+--------+--------+--------+-------
 I | C |                 |                 |A|        |        |        |
 n | O |    Target IP    |    DateTime     |C|  Min   |  Avg   |  Max   | Drop
 t | S |     Address     |mm/dd/yy,hh:mm:ss|T|  RTT   |  RTT   |  Rtt   | Resp
---+---+-----------------+-----------------+-+--------+--------+--------+-------
 1 | 1 |    172.16.78.11 |09/04/07-13:09:41|Y|       0|       0|       0|      1
 2 | 1 |    172.16.78.11 |09/04/07-13:24:41|Y|       0|       0|       0|      1
 3 | 1 |    172.16.78.11 |09/04/07-13:39:41|Y|       0|       0|       0|      1
 4 | 1 |    172.16.78.11 |09/04/07-13:54:41|Y|       0|       0|       0|      1
 5 | 1 |    172.16.78.11 |09/04/07-14:09:41|Y|       0|       0|       0|      1
Type A<CR> to print all, <CR> to continue, Q<CR> to stop:
```

```
zSH> ipsla stats history ipaddress 10.2.1.254 index 1
---+---+-----------------+-----------------+-+--------+--------+--------+-------
 I | C |                 |                 |A|        |        |        |
 n | O |    Target IP    |    DateTime     |C|  Min   |  Avg   |  Max   | Drop
 t | S |     Address     |mm/dd/yy,hh:mm:ss|T|  RTT   |  RTT   |  Rtt   | Resp
---+---+-----------------+-----------------+-+--------+--------+--------+-------
 1 | 1 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      2
 1 | 2 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      0
 1 | 3 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      0
 1 | 4 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      0
 1 | 5 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      0
 1 | 6 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      1
 1 | 7 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      1
 1 | 8 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      1


zSH> ipsla stats history index 1 cosactionindex 3
---+---+-----------------+-----------------+-+--------+--------+--------+-------
 I | C |                 |                 |A|        |        |        |
 n | O |    Target IP    |    DateTime     |C|  Min   |  Avg   |  Max   | Drop
 t | S |     Address     |mm/dd/yy,hh:mm:ss|T|  RTT   |  RTT   |  Rtt   | Resp
---+---+-----------------+-----------------+-+--------+--------+--------+-------
 1 | 3 |      10.2.1.254 |09/12/07-21:48:58|Y|       0|       0|       0|      0
 1 | 3 |   172.24.94.254 |09/12/07-21:48:58|Y|       0|       0|       0|      0
```

INDEX