ixia

IxNetwork Tcl Development Guide

8.03 EA,

March 2016

Copyright and Disclaimer

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

Ixia, the Ixia logo, and all Ixia brand names and product names in this document are either trademarks or registered trademarks of Ixia in the United States and/or other countries. All other trademarks belong to their respective owners.

The information herein is furnished for informational use only, is subject to change by Ixia without notice, and should not be construed as a commitment by Ixia. Ixia assumes no responsibility or liability for any errors or inaccuracies contained in this publication.

Corporate Headquarters	Ixia Worldwide Headquarters 26601 W. Agoura Rd. Calabasas, CA 91302 USA +1 877 FOR IXIA (877 367 4942) +1 818 871 1800 (International) (FAX) +1 818 871 1805 sales@ixiacom.com	Web site: www.ixiacom.com General: info@ixiacom.com Investor Relations: ir@ixiacom.com Training: training@ixiacom.com Support: support@ixiacom.com +1 818 595 2599
EMEA	Ixia Europe Limited Part 2nd floor, Clarion House, Norreys Drive Maidenhead, UK SL6 4FL +44 (1628) 408750 FAX +44 (1628) 639916 salesemea@ixiacom.com	Support: support-emea@ixiacom.com +40 21 301 5699
Asia Pacific	Ixia Pte Ltd 210 Middle Road #08-01 IOI Plaza Singapore 188994	Support: support-asiapac@ixiacom.com +91 80 4939 6410
Japan	Ixia KK	Support: support-japan@ixi-

	Nishi-Shinjuku Mitsui Bldg 11F 6-24-1, Nishi-Shinjuku, Shinjuku-ku Tokyo 160-0023 Japan	acom.com +81 3 5326 1980
India	Ixia Technologies Pvt Ltd Tower 1, 7th Floor, UMIYA Business Bay Cessna Business Park Survey No. 10/1A, 10/2, 11 & 13/2 Outer Ring Road, Varthur Hobli Kadubeesanahalli Village Bangalore East Taluk Bangalore-560 037, Karnataka, India +91 80 42862600	Support: support-india@ixiacom.com +91 80 4939 6410
China	Ixia Technologies (Shanghai) Company Ltd Unit 3, 11th Floor, Raffles City, Beijing Beijing, 100007 P.R.C.	Support: support-china@ixiacom.com 400 898 0598 (Greater China Region) +86 10 5732 3932 (Hong Kong)

Change: 4150211*, Date:* September 24, 2013

For viewing the FAQs related to the product, go to Ixia Technical Support Online: https://ebsoprod.ixiacom.com/OA_HTML/jtflogin.jsp

Table of Contents

[This page intentionally left blank]

Chapter 1: Introduction

Protocols

This manual discusses the Tcl interface for the use of intelligent routing protocols with Ixia hardware, via the IxNetwork™ GUI. This manual should be an adjunct to the *Tcl Development Guide*, which describes the non-protocol use of the Ixia Tcl API. That manual contains overview material related to the use of the Tcl API, and should be consulted before use of the IxNetwork Tcl API.

NOTE The commands detailed in this manual are legacy Tcl APIs and are no longer being developed or sustained. A new set of API commands that are up to date and allow for much more control over the IxNetwork functionality are detailed in a separate manual called *IxNetwork Tcl API Guide*.

ScriptGen

ScriptGen is an auxiliary Tcl tool that is installed as part of the Tcl client package. Its purpose is to create a Tcl program that reflects the configuration of a particular port, including protocol interfaces and routing protocols. ScriptGen is run from a wish console, and the resulting program is written to disk and shown in the console window. The configuration of the port may have been established through any of the Ixia tools, such as IxNetwork™, IxExplorer®, ScriptMate™, or the Tcl API. The operation of ScriptGen is described in the *IxExplorer User Guide, Appendix A: Using ScriptGen*.

Layout of This Manual

This guide has a number of chapters and appendixes that convey usage and reference information. The chapters of the manual are:

- Chapter 1: Introduction. This chapter.
- Chapter 2: High-Level and Utility API Description. Organizes the high-level and utility APIs into related discussion groups and describes how to use them at a high level.
- Chapter 3: IxTclHal API Description. Organizes the APIs into related discussion groups and describes how to use them at a high level.
- Appendix B: IxTclHAL Protocol Server Commands. An alphabetical set of reference sheets for all protocol-related Tcl commands.
- Appendix A: High-Level API. Commands that perform a combination of functions against a number of ports.

For additional reference information, see:

- *Ixia Reference Manual - Theory of Operations: Protocols* Explains the conceptual model behind the Ixia emulation of routing protocols.
- *Ixia Reference Manual - Available Statistics* A description of available statistics.

What is Deprecated in IxNetwork

Ixia recommends avoiding usage of *Deprecated* commands as they might be superseded in the near future, possibly as soon as the next revision of the API. Deprecated status may also indicate that the feature will be removed in the future. While a deprecated software feature remains in the software, its use may raise warning messages recommending alternative practices. Deprecated features may continue to be available in future releases,until any alternative practice is available.

The following table lists the commands, subcommands, and options that have been deprecated through the lifetime of the IxOS, IxRouter, and IxNetwork Tcl APIs. Refer to the appropriate release manual to determine the reason for the deprecation.

All Deprecated Commands and Options

Command	Subcommands	Options	First Deprecated Release
BGP	bgp4Neighbor	generateStreams	6.10
	bgp4RouteItem	generateStreams	6.10
	bgp4Server	generateStreams	6.10
IGMP	igmpGroupRange	generateStreams	6.10
	igmpHost	generateStreams	6.10
	igmpVxServer	generateStreams	6.10
MLD	mldServer	generateStreams	6.10
	mldHost	generateStreams	6.10
	mldGroupRange	generateStreams	6.10
ISIS	isisServer	generateStreams	6.10
LDP	ldpServer	generateStreams	6.10
OSPF	ospfServer	generateStreams	6.10
RIP	ripServer	generateStreams	6.10
RIPNG	ripngServer	generateStreams	6.10
RSVP	rsvpServer	generateStreams	6.10

Support for External License Server

NOTE For additional information about IxNetwork licensing, refer to the *Getting Started with IxNetwork* manual. General information about licensing of Ixia products is available in the Ixia *License Management User Guide*.

Licensing is implemented in IxNetwork for the GUI and Tcl implementations. IxNetwork licenses can be installed on the chassis or on an external license server. The default location for the IxNetwork license server is on the chassis.

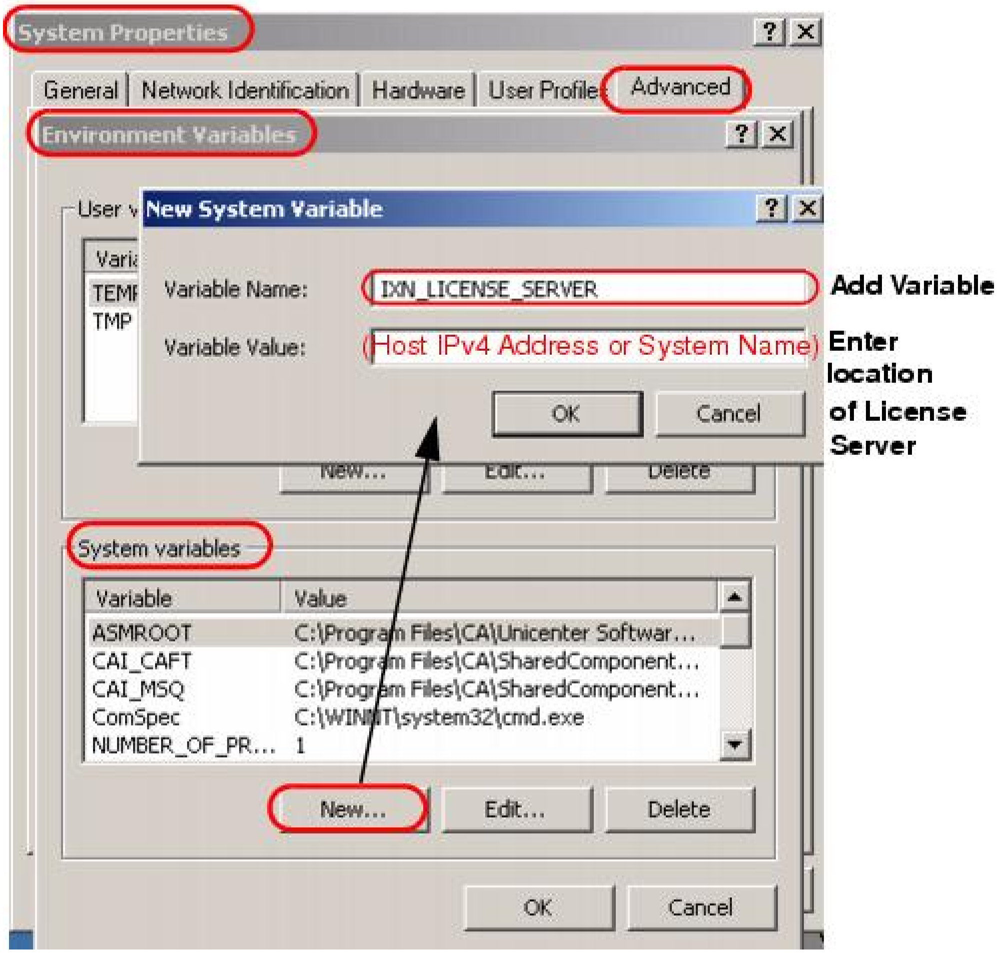
If your license resides on the chassis being used by your Tcl program, no adjustment is necessary. If you are using a license server, please continue reading.

Although the GUI offers the user the ability to designate an independent license server as a menu option, no such option is available to a Tcl program. Instead, an environment variable named IXN_LICENSE_SERVER must be set.

Setting the Environment Variable on a Windows Host

If you are running your Tcl program on a Unix client, the environment variable must be set on the host running the Tcl server. If you are running your Tcl program on a Windows client, it must be set on that client. Environment variables are set on a Unix host through the user's shell initialization script. Environment variables are set on a Windows host by using the following steps.

1. Right-click the **My Computer** icon on the desktop, then left-click **Properties**.
2. Choose the **Advanced tab** from the System Properties dialog box.
3. Click the **Environment Variables** button at the bottom of the page.
4. In either the "User variables for <user" or the "System variables" list, if IXN_LICENSE_SERVER does not exist in the list, click **New**. Otherwise, click **Edit**.
5. The name of the variable should be IXN_LICENSE_SERVER, and the value should be the name or IP address of the license server machine.



Variable Name: IXN_LICENSE_SERVER

Variable Value: (Host IPv4 Address or System Name)

Add Variable

Enter location of License Server

Variable	Value
ASMROOT	C:\Program Files\CA\Unicenter Softwar...
CAI_CAFT	C:\Program Files\CA\SharedComponent...
CAI_MSQ	C:\Program Files\CA\SharedComponent...
ComSpec	C:\WINNT\system32\cmd.exe
NUMBER_OF_PR...	1

Advice to Readers

Readers unfamiliar with the Tcl APIs should refer to chapters in the *Tcl Reference Guide* to understand the concepts of Tcl programming.

The following chapters in the *Tcl Reference Guide* are essential elements in understanding how the APIs are to be used:

- *Quick Start Chapter*
- *Programming Chapter*

People unfamiliar with the Ixia system should read the *Theory of Operation Chapters* in the *Ixia Reference Guide* to understand how the hardware functions and to understand the basics of the protocol emulations that are used.

The API description chapter in this manual should be read, in part, as the elements are needed. For example, you need not read the Border Gateway Protocol Version 4 (BGP4) sections until you need to use the BGP4 protocol.

The appendixes should be used for reference.

Chapter 2: High-Level and Utility API Description

This chapter presents a description of the High-Level API commands organized by protocol:

- IGMP
  - ixTransmitIgmpJoin/ixStartIgmp
  - ixTransmitIgmpJoin/ixStopIgmp
- BGP4
  - ixStartBGP4 / ixStopBGP4
- OSPF
  - ixStartOspf / ixStopOspf
- OSPFv3
  - ixStartOspfV3 / ixStopOspfV3
- ISIS
  - ixStartIsis / ixStopIsis
- LDP
  - ixStartLdp / ixStopLdp
- MPLS OAM
  - ixStartMplsOam / ixStopMplsOam
- MLD
  - ixStartMld / ixStopMld
- RIP
  - ixStartRip / ixStopRip
- RIPng
  - ixStartRipng / ixStopRipng
- RSVP
  - ixStartRsvp / ixStopRsvp
- PIM-SM
  - ixStartPimsm / ixStopPimsm
- STP
  - ixStartStp / ixStopStp
- EIGRP
  - ixStartEigrp / ixStopEigrp
- BFD
  - ixStartBFD / ixStopBFD
- CFM
  - ixStartCfm / ixStopCfm

This chapter provides an overview of the high-level API functions and utility commands. The full details of the commands described herein may be found in the following appendix:

- *Appendix A - High-Level API* includes complete descriptions of each of the high-level commands.

The high-level commands are characterized by one or more characteristics:

- They perform a combination of IxTclHAL commands.
- They perform one or more IxTclHAL commands over a range of ports.
- They control test operation sequences.

Arguments to the high-level APIs are passed in one of two ways:

- **By value** — denoted by (By value) in the Appendix C description. By value arguments are either a constant or a $ variable reference. For example: 32, {{1 1 1} {1 2 1}} or $portList
- **By reference** — denoted by (By reference) in the Appendix C description. By value arguments must be references to variables, **without** the `$'. For example, *pl* after set pl {{1 1 1} [1 1 2}} or one2oneArray.

Read the individual description pages in Appendix C to determine which arguments are passed by reference and by value.

Protocols

IGMP

ixTransmitIgmpJoin/ixStartIgmp

This command sends a message to the IxServer to start transmission of IGMP membership messages for a list of ports. The format of these commands is:

ixTransmitIgmpJoin/ixStartIgmp *portList [groupId]*

where *portList* identifies a number of ports and *groupId* is the optional group ID number to use (101064 is the default).

Refer to ixTransmitIgmpJoin/ixStartIgmp for a full description of this command.

ixTransmitIgmpJoin/ixStopIgmp

This command sends a message to the IxServer to start transmission of IGMP membership leave messages for a list of ports. The format of these commands is:

ixTransmitIgmpJoin/ixStopIgmp *portList [groupId]*

where *portList* identifies a number of ports and *groupId* is the optional group ID number to use (101064 is the default).

Refer to ixTransmitIgmpJoin/ixStopIgmp for a full description of this command.

BGP4

ixStartBGP4 / ixStopBGP4

These commands start and stop the BGP4 component of the protocol server for a list of ports. The format of these commands is:

ixStartBGP4 *portList*

ixStopBGP4 *portList*

Refer to ixStartBGP4 and ixStopBFD for a full description of these commands.

OSPF

ixStartOspf / ixStopOspf

These commands start and stop the OSPF component of the protocol server for a list of ports. The format of these commands is:

ixStartOspf *portList*

ixStopOspf *portList*

Refer to ixStartOspf and ixStopOspf for a full description of these commands.

OSPFv3

ixStartOspfV3 / ixStopOspfV3

These commands start and stop the OSPFv3 component of the protocol server for a list of ports. The format of these commands is:

ixStartOspfV3 *portList*

ixStopOspfV3 *portList*

Refer to ixStartOspfV3 and ixStopOspfV3 for a full description of these commands.

ISIS

ixStartIsis / ixStopIsis

These commands start and stop the ISIS component of the protocol server for a list of ports. The format of these commands is:

ixStartIsis *portList*

ixStopIsis *portList*

Refer to ixStartIsis and ixStopIsis for a full description of these commands.

RSVP

ixStartRsvp / ixStopRsvp

These commands start and stop the RSVP component of the protocol server for a list of ports. The format of these commands is:

ixStartRsvp *portList*

ixStopRsvp *portList*

Refer to ixStartRsvp and ixStopRsvp for a full description of these commands.

LDP

ixStartLdp / ixStopLdp

These commands start and stop the LDP component of the protocol server for a list of ports. The format of these commands is:

ixStartLdp *portList*

ixStopLdp *portList*

Refer to ixStartLdp and ixStopLdp for a full description of these commands.

MPLS OAM

ixStartMplsOam / ixStopMplsOam

These commands start and stop the MPLS OAM component of the protocol server for a list of ports. The format of these commands is:

ixStartMplsOam *portList*

ixStopMplsOam *portList*

MLD

ixStartMld / ixStopMld

These commands start and stop the MLD component of the protocol server for a list of ports. The format of these commands is:

ixStartMld *portList*

ixStopMld *portList*

Refer to ixStartMld and ixStopMld for a full description of these commands.

RIP

ixStartRip / ixStopRip

These commands start and stop the RIP component of the protocol server for a list of ports. The format of these commands is:

ixStartRip *portList*

ixStopRip *portList*

Refer to ixStartRip and ixStopRip for a full description of these commands.

RIPng

ixStartRipng / ixStopRipng

These commands start and stop the RIPng component of the protocol server for a list of ports. The format of these commands is:

ixStartRipng *portList*

ixStopRipng *portList*

Refer to ixStartRipng and ixStopRipng for a full description of these commands.

PIM-SM

ixStartPimsm / ixStopPimsm

These commands start and stop the PIM-SM component of the protocol server for a list of ports. The format of these commands is:

ixStartPimsm *portList*

ixStopPimsm *portList*

Refer to ixStartPimsm and ixStopPimsm for a full description of these commands.

STP

ixStartStp / ixStopStp

These commands start and stop the STP component of the protocol server for a list of ports. The format of these commands is:

ixStartStp *portList*

ixStopStp *portList*

Refer to ixStartStp and ixStopStp for a full description of these commands.

EIGRP

ixStartEigrp / ixStopEigrp

These commands start and stop the EIGRP component of the protocol server for a list of ports. The format of these commands is:

ixStartEigrp *portList*

ixStopEIgrp *portList*

Refer to ixStartEigrp and ixStopEigrp for a full description of these commands.

BFD

ixStartBfd / ixStopBfd

These commands start and stop the BFD component of the protocol server for a list of ports. The format of these commands is:

ixStartBfd *portList*

ixStopBfd *portList*

Refer to ixStartBFD and ixStopBFD for a full description of these commands.

CFM

ixStartCfm / ixStopCfm

These commands start and stop the CFM component of the protocol server for a list of ports. The format of these commands is:

ixStartCfm *portList*

ixStopCfm *portList*

Refer to ixStartCfm and ixStopCfm for a full description of these commands.

Chapter 3: IxTclHal API Description

The protocol server implements a number of intelligent, bidirectional test subcommands and data-gathering routines.

This chapter presents an organized description of the IxTclHAL API commands based on protocols. The protocols covered are:

- ARP—sends ARP requests and maintains an IP address to MAC address correspondence table based on responses.
- IGMP—sends and responds to IGMP messages.
- IGMP(New)—sends and responds to IGMPv3 messages.
- MLD—sends and responds to MLD messages.
- BGP4—simulates one or more BGP4 routers in a network of routers.
- OSPF—simulates one or more OSPF routers in a network of routers.
- OSPFv3—simulates one or more OSPFv3 routers in a network of routers.
- ISIS—simulates one or more IS-IS routers in a network of routers.
- RSVP-TE—simulates one or more RSVP ingress or egress routers. Concentrates on Traffic Engineering parameters.
- LACP—simulates one or more Link Aggregation Control Protocol actors and partners.
- LDP—simulates one or more routers that use the label distribution protocol.
- MPLS OAM—establishes communication channel with BGP Protocol, which sends the information about received labels to the MPLS OAM module which is used to send echo request message.
- Link OAM—simulates Link OAM Protocol for monitoring remote fault indication and remote loopback control on a point to point Ethernet link.
- RIP—simulates one or more RIP routers in a network of routers.
- RIPng—simulates one or more RIPng routers in a network of routers.
- PIM-SM—simulates one or more PIM-SM routers in a network of routers.
- STP—simulates one or more STP/RSTP/MSTP/PVST+/RPVST+ bridges in a network of bridges.
- EIGRP—simulates one or more EIGRP routers in a network of routers.
- BFD—simulates one or more BFD routers in a network of routers.
- CFM—simulates one or more CFM bridges in a network of bridges.
- MPLS-TP—simulates one or more MPLS-TP routers in a network of routers.

NOTE In many of the protocols implemented by the protocol server, lists of items are maintained. These lists are always accessible by one of two mechanisms:

- *getFirst<Item/getNext<Item*—get the first item in the list and then the next and then the next...
- *get<Item*—get an item based on its identifying name.

It is important that the two mechanisms not be mixed on a protocol-by-protocol basis. basis. All items must be accessed by iterating through the list or by named access.

NOTE This also affects the manner in which an <Item is overwritten with a *set<Item* command. These commands take an optional matching name argument. That name must be supplied if the object was fetched with the *get<Item* command and must not be used when it was fetched with the *getFirst<Item/getNext<Item* commands.

All commands are covered within these sections, but only the most significant options and subcommands are discussed. Not all of the options, nor all of the subcommands can be assumed to be discussed in this chapter. In particular, if not otherwise noted the`get`, `cget`, `config`, `set`, `setDefault`, `decode`, and `write` subcommands are assumed to exist and to perform standard functions.

IxTclHAL Protocol Server Commands includes complete descriptions of each of the IxHal commands.

protocolServer

The protocolServer command enables/disables each of the protocol servers and provides the first entry for the IP address table. See protocolServer for full details.

The important options of this command are:

protocolServer Options

protocolServer Options	
Member	**Usage**
enableArpResponse	(Non-POS cards only) Enables ARP requests and responses.
enableBfdService	Enables BFD testing.
enableBgp4Service	Enables BGP4 testing.
enableCfmService	Enables CFM testing.
enableIgmpQueryResponse	Enables IGMP testing.
enableIsisService	Enables ISIS testing.
enableOspfService	Enables OSPF testing.
enableOspfv3Service	Enables OSPFv3 testing.
enablePingResponse	Enables PING requests and responses.
enableRipService	Enables RIP testing.
enableRipngService	Enables RIPng testing.
enableLacpService	Enables LACP testing.
enableLdpService	Enables LDP testing.
enableRsvpService	Enables RSVP testing.
enableMldService	Enables MLD testing.
enablePimsmService	Enables PIM-SM testing.
enableStpService	Enables STP testing.
enableEigrpService	Enables EIGRP testing.
enableMplsTpService	If true the mplsTp protocol is enabled.

Interface Table

The interface table is used to hold a number of logical interfaces that are associated with an Ixia port. Each interface may have none or more IPv4[1] and IPv6 addresses associated with a MAC address and optional VLAN ID.

Please refer to the *Tcl Development Guide* for a discussion of the Ixia protocol server's testing model with respect to interfaces.

[1] Only one IPv4 is currently allowed for interfaces.

ARP

Please refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to ARP.

arpServer

The ARP table is automatically populated when ARP responses are received from automatically generated ARP requests.

The `arpAddressTableEntry` command operates in concert with `arpServer` to show the entries in the table.

The `arpServer` object sets the position in the list and `arpAddressTableEntry` accesses the entry at the current position. The typical series of operations is shown in the table below:

Typical Address Table Operations

Operation	Steps
Look through ARP table	1. Use the `get` subcommand of the `arpServer` command to transfer the data from the hardware to the object. 2. Use the `get` subcommand of the `arpAddressTableEntry` command to get the data into the `arpAddressTableEntry` options. 3. Use the `getNextItem` subcommand of the `arpServer` command to position to the next table item. 4. Repeat steps 2 and 3 until an error is returned from step 3.
Find the ARP table item for an IP address	1. Use the `getItem` subcommand of the `arpServer` command to position the list to the correct entry. 2. Use the `get` subcommand of the `arpAddressTableEntry` command to get the data into the `arpAddressTableEntry` options.

See arpServer for details. The important options and subcommands of this command are:

arpServer Options

Member	Usage
mode	The type of ARP request handling: • Send a single ARP request to each gateway IP address for the first IP address found in the IP address table. Results are saved in the ARP table. • Send ARP requests using all of the addresses found in the IP address table as source addresses. ARP responses are ignored. • Both operations.
rate	ARP frame rates in frames per second.
retries	Number of retries.

arpServer Subcommands

Member	Usage
clearArpTable	Clears the ARP table.
getEntry	Finds the entry for a particular IP address. The data may be retrieved by calling `arpAddressTableEntry.get.`

arpServer Subcommands

Member	Usage
getFirstEntry	Positions to the first entry in the list. The data may be retrieved by calling `arpAddressTableEntry.get`.
getNextEntry	Positions to the next entry in the list. The data may be retrieved by calling `arpAddressTableEntry.get`.
sendArpRequest	Sends ARP requests as per the `mode` member.

arpAddressTableEntry

See *arpAddressTableEntry* for full details. The important options of this command are:

arpAddressTableEntry Options

Member	Usage
ipAddress	IP address for the entry.
macAddress	MAC address for the entry.

ARP

Please refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to ARP.

arpServer

The ARP table is automatically populated when ARP responses are received from automatically generated ARP requests.

The `arpAddressTableEntry` command operates in concert with `arpServer` to show the entries in the table.

The `arpServer` object sets the position in the list and `arpAddressTableEntry` accesses the entry at the current position. The typical series of operations is shown in the table below:

Typical Address Table Operations

Operation	Steps
Look through ARP table	1. Use the `get` subcommand of the `arpServer` command to transfer the data from the hardware to the object. 2. Use the `get` subcommand of the `arpAddressTableEntry` command to get the data into the `arpAddressTableEntry` options. 3. Use the `getNextItem` subcommand of the `arpServer` command to position to the next table item. 4. Repeat steps 2 and 3 until an error is returned from step 3.
Find the ARP table item for an IP address	1. Use the `getItem` subcommand of the `arpServer` command to position the list to the correct entry. 2. Use the `get` subcommand of the `arpAddressTableEntry` command to get the data into the `arpAddressTableEntry` options.

See arpServer for details. The important options and subcommands of this command are:

arpServer Options

Member	Usage
mode	The type of ARP request handling: • Send a single ARP request to each gateway IP address for the first IP address found in the IP address table. Results are saved in the ARP table. • Send ARP requests using all of the addresses found in the IP address table as source addresses. ARP responses are ignored. • Both operations.
rate	ARP frame rates in frames per second.
retries	Number of retries.

arpServer Subcommands

Member	Usage
clearArpTable	Clears the ARP table.
getEntry	Finds the entry for a particular IP address. The data may be retrieved by calling `arpAddressTableEntry.get.`

arpServer Subcommands

Member	Usage
getFirstEntry	Positions to the first entry in the list. The data may be retrieved by calling `arpAddressTableEntry.get`.
getNextEntry	Positions to the next entry in the list. The data may be retrieved by calling `arpAddressTableEntry.get`.
sendArpRequest	Sends ARP requests as per the `mode` member.

arpAddressTableEntry

See *arpAddressTableEntry* for full details. The important options of this command are:

arpAddressTableEntry Options

Member	Usage
ipAddress	IP address for the entry.
macAddress	MAC address for the entry.

IGMP

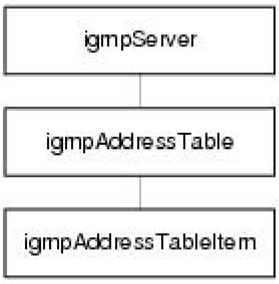
An extended IGMP implementation is available for use with newer Ixia ports containing port CPUs. This extended implementation is listed in the next section (IGMP (New)) and covers IGMP versions 1, 2, and 3. The older IGMP implementation described here will be used with non-CPU based Ixia ports, and may be used with newer ports as well. This older implementation will not handle IGMP version 3, however. Use of the two IGMP implementations should not be mixed for a particular port.

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia's protocol server's testing model with respect to IGMP. The IGMP related commands are:

- igmpServer - configures overall operation of the protocol server's IGMP operation.
- igmpAddressTable - a container used to hold the list of address table items.
- igmpAddressTableItem - an individual item for the IGMP Address Table.

These commands and the data that they maintain are arranged in a hierarchy as shown in the following figure.

IGMP Command Hierarchy



igmpServer

The igmpServer object configures the overall operation of the IGMP protocol server. See igmpServer for full details. The important options of this command are:

igmpServer Options

Member	Usage
version	Version 1 or 2 of the protocol.
sendRouterAlert	Sets the IP header Send Router Alert bit.
reportMode	Basic mode of response: • Report to one when queried — causes each simulated host to respond just to the specific query that it is presented with. • Report to all when queried — causes each simulated host to respond with all of its memberships, regardless of the type of query that it is presented with.
	• Report to all unsolicited — causes each simulated host to automatically send full memberships messages at regular intervals.
reportFrequency	When the mode is *report to all unsolicited,* this is the frequency in

igmpServer Options

Member	Usage
	seconds with unsolicited messages are generated.
enableQueryResponse	Enables responses after initial join message.

igmpAddressTable

The address table is a list of entries, each of which is described in the item command. One positions within the list with the address table object and accesses elements with the list object. The typical series of operations is shown in the table below:

Typical Address Table Operations

Operation	Steps
Add table items	1. Set values in the `igmpAddressTableItem` command. 2. Use the `set` subcommand of the `igmpAddressTableItem` command which transfers the data into a holding area. 3. Use the `addItem` subcommand of the `igmpAddressTable` command to move the data from the holding area to the actual list. 4. Repeat steps 1, 2, and 3 for each table item to be added. 5. Use the `set` subcommand of the `igmpAddressTable` command to send the table to the hardware.
Look through table	1. Use the `get` subcommand of the `igmpAddressTable` command to transfer the data from the hardware to the object. 2. Use the `get` subcommand of the `igmpAddressTableItem` command to get the data into the `ipAddressTableItem` options. 3. Use the `getNextItem` subcommand of the `igmpAddressTable` command to position to the next table item. 4. Repeat steps 2 and 3 until an error is returned from step 3.

See igmpAddressTable for full details. The important subcommands of this command are:

igmpAddressTable Subcommands

Member	Usage
clear	Clears the IGMP address table.
addItem	Adds the table item as set by the last call to `igmpAddressTableItem.set` to the table at the current table position.
delItem	Deletes the address table item at the current position.
getFirstItem	Positions to the first table item.
getNextItem	Moves to the next table item.

igmpAddressTableItem

The `igmpAddressTableItem` is used in concert with the `igmpAddressTable` command. This command holds an individual table item; `igmpAddressTable` takes care of keeping the actual list of address table items. See igmpAddressTableItem for full details. The important options of this command are:

igmpAddressTableItem Options

Member	Usage
fromClientAddress	The client address range `toClientAddress` is read-only.

igmpAddressTableItem Options

Member	Usage
toClientAddress	
fromGroupAddress toGroupAddress	The group address range `toGroupAddress` is read-only.
numClientAddresses	The number of consecutive client addresses.
numGroupAddresses	The number of consecutive group addresses.

IGMP (New)

An extended IGMP implementation is available for use with newer Ixia ports containing port CPUs. This extended implementation covers IGMP versions 1, 2, and 3 and is referred to as IGMPvX. The older IGMP implementation will be used with non-CPU based Ixia ports, and may be used with newer ports as well. The older implementation will not handle IGMP version 3, however. Use of the two IGMP implementations should not be mixed for a particular port.

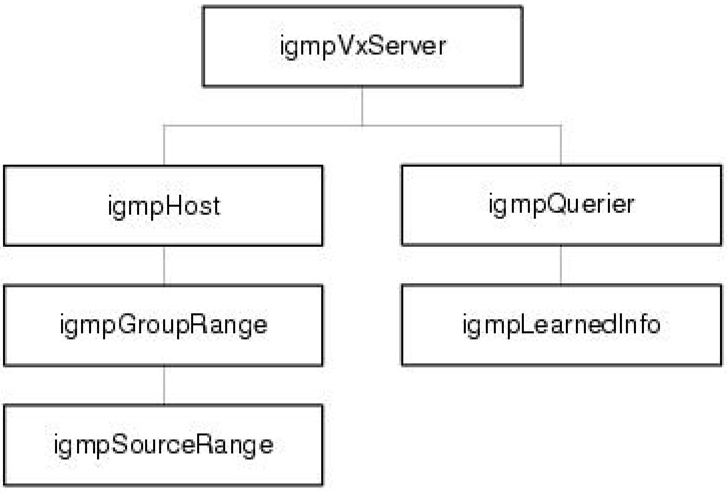
The IGMPvX implementation is very close to the MLD implementation. IGMP deals with IPv4 multicast and MLD deals with IPv6 multicast. Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a full discussion of the MLD implementation.

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to IGMP. The IGMPVx related commands are:

- igmpVxServer - configures overall operation of the protocol server's IGMP operation.
- igmpQuerier - a list of simulated IGMP Queriers.
- igmpLearnedInfo - information learned by the Querier.
- igmpHost - a list of simulated hosts which received multicast traffic.
- igmpGroupRange holds a list of multicast addresses that a particular host is interested in.
- igmpSourceRange - holds a list of source IPv4 addresses from which multicast traffic should be included or excluded.

These commands and the data that they maintain are arranged in a hierarchy, as shown in the following figure.

IGMPvX Command Hierarchy



igmpVxServer

The *igmpVxServer* object configures the overall operation of the IGMP protocol server. See igmpVxServer for full details. The important options and subcommands of this command are:

igmpVxServer Options

Member	Usage
numGroups timePeriod	Provides a means of throttling the amount of IGMP traffic generated by the port.
enableSendLeaveOnStop	If true, enables the Send Leaves on Stop feature (for IGMP versions 2 and 3). *(True / False).*

igmpVxServer Subcommands

Member	Usage
select	Must be used first to select the port being configured.
clearAllHosts	Removes all hosts from the list of hosts.
addHost	Adds a host to the list. The host must have been previously configured through the use of the igmpHost command.
getFirstHost getNextHost getHost	Allows iterative or direct access to all the table items. Each accessed item is available via the igmpHost command.
delHost	Deletes a host that has been accessed directly or iteratively.
generateStreams	Generates traffic streams for all of the hosts.
setHost	Allows the configuration values for a host to be overwritten on the fly.
clearAllQueriers	Removes all queriers from the list of queriers.
addQuerier	Adds a querier to the list. The querier must have been previously configured through the use of the igmpQuerier command.
getQuerier getFirstQuerier getNextQuerier	Allows iterative or direct access to all the table items. Each accessed item is available via the igmpQueriercommand.
setQuerier	Allows the configuration values for a querier to be overwritten on the fly.

igmpQuerier

The *igmpQuerier* object describes an emulated IGMP Querier. See igmpQuerier for full details. The important options and subcommands of this command are:

igmpQuerier Options

Member	Usage
enable	Enables the use of the emulated querier in the IGMP simulation. *(True / False)*
version	Indicates the IGMP protocol version to be used. One of: • igmpQuerierVersion1 • igmpQuerierVersion2 • igmpQuerierVersion3
startupQueryCount	The number of IGMP General Query messages sent at startup. (Integer) (Default = 2).
generalQueryInterval	The amount of time (in seconds) between IGMP General Query messages sent by the router. (Default = 125)
robustnessVariable	Defines the subnet vulnerability to lost packets. IGMP can

igmpQuerier Options

Member	Usage
	recover from robustness variable minus 1 lost IGMP packets. The robustness variable should be set to a value of 2 or greater. (Default = 2)
genQueryResponseInterval	The maximum amount of time (in seconds) that the IGMP querier waits to receive a response to a General Query message. (Default = 10 seconds, and must be less than the Query Interval)
specQueryResponse Variable	The maximum amount of time (in seconds) that the IGMP querier waits to receive a response to a Specific Query message. (Default = 10 seconds, and must be less than the Query Interval)
specQuery TransmissionCount	Indicates the total number of Specific Query messages sent every Specific Query Response Interval (in seconds) before assuming that there is no interested listener for the particular group/source.
discardLearnedInfo	When disabled, the emulated Querier maintains a complete record state for received reports and sent queries (based on timer expiry for received groups and sources). (Default is disabled) When enabled, the Querier does not maintain any database and only sends periodic General Queries. The Specific Query group/source record information is not calculated based on any earlier received report, but solely based on the last received report.
supportElection	Indicates whether the Querier participates in Querier election or not. If disabled, then all incoming Query messages are discarded.
supportOlderVersionHost	Indicates whether the Querier will comply with RFC 3376 Section 7.3.2 and RFC 3810 Section 8.3.2. If disabled, all membership reports with version less than the current version are discarded.
supportOlderVersionQuerie	Indicates whether the Querier downgrades to the lowest version of received Query messages. If disabled, all Query messages with version less than the current version are discarded.
enableRouterAlert	If enabled, sets the "Send Router Alert" bit in the IP header.
Learned Info	
isQuerier	*(Read-only)* If true, indicates that the currently-elected querier is self. If false, indicates that the currently-elected querier is other. (*True / False*)
querierAddress	*(Read-only)* Indicates the IPv4 address of the currently-elected querier. (String)
querierWorkingVersion	*(Read-only)* Indicates the working version of the IGMP querier at that point in time. (Integer).

igmpQuerier Subcommands

Member	Usage
requestLearnedInfo	Requests the learned IGMP information for the respective IGMP Querier.
getLearnedInfoList	Retrieves the list of learned info.
getFirstLearnedInfo getNextLearnedInfo	Retrieves the first learned info entry, then iterates through the list of additional entries.
setDefault	Sets default values for all configuration options.

igmpLearnedInfo

The *igmpLearnedInfo* object describes information learned by the IGMP Querier. See igmpLearnedInfo for full details. The important options and subcommands of this command are:

igmpLearnedInfo Options

Member	Usage
groupAddress	The IPv4 address for the router group. (IPv4-format address)
groupTimer	The number of seconds remaining in the group address timer. (Integer)
compatibilityTimer	The number of seconds remaining in the compatibility timer. (Integer)
filterMode	Whether this group address is included or excluded. One of: • INCLUDE • EXCLUDE
compatibilityMode	What version of IGMP this group address is. One of: • IGMPV1 • IGMPV2 • IGMPV3
sourceAddress	The IPv4 address for the group source. (IPv4-format address)
sourceTimer	The number of seconds remaining in the source address timer. (Integer)

igmpLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values

igmpHost

The *igmpHost* object describes a simulated IGMP host. See igmpHost for full details. The important options and subcommands of this command are:

igmpHost Options

Member	Usage
enable	Enables the use of the host in the IGMP simulation.
enableGeneralQuery enableGroupSpecific	Enables the simulation for the host to respond to general and/or group specific queries.
enableRouterAlert	Sets the router alert bit in listener report messages.

igmpHost Options

Member	Usage
enableUnsolicited reportFrequency	If set, causes the simulation for this host to send unsolicited listener report messages at a particular frequency.
version	Sets the IGMP version number that is to be simulated on the host: 1, 2, or 3.
protocolInterface Description	The name of the defined *interfaceEntry* which describes the host interface to be simulated.

igmpHost Subcommands

Member	Usage
clearAllGroupRanges	Removes all group ranges from the list of group ranges.
addGroupRange	Adds a group range to the list. The group range must have been previously configured through the use of the igmpGroupRange command.
getFirstGroupRange getNextGroupRange getGroupRange	Allows iterative or direct access to all the table items. Each accessed item is available via the igmpGroupRange command.
delGroupRange	Deletes a group range that has been accessed directly or iteratively.
generateStreams	Generates traffic streams for all of the group ranges.
setGroupRange	Allows the configuration values for a group range to be overwritten on the fly.

igmpGroupRange

The *igmpGroupRange* object configures a set of multicast addresses from which a host wishes to receive traffic. See igmpGroupRange for full details. The important options and subcommands of this command are:

igmpGroupRange Options

Member	Usage
enable	Enables the use of the group range in the IGMP simulation.
groupIpFrom groupCount incrementStep	Specifies the set of IPv4 multicast addresses in the group range.
enablePacking recordsPerFrame sourcesPerRecord	If enabled, this option controls how many multicast records and sources will be included in each listener report for this group range. By default, when packing is NOT enabled, all records will be sent in one frame. If the user wants a specified number of records to be sent in each frame, packing should be enabled (enablePacking is true), and the number of records indicated with the *recordsPerFrame* option.
sourceMode	Indicates whether the associated source range is a set of IP addresses to be included or excluded.

igmpGroupRange Subcommands

Member	Usage
clearAllSourceRanges	Removes all source ranges from the list of source ranges.
addSourceRange	Adds a source range to the list. The source range must have been previously configured through the use of the igmpSourceRange command.
getFirstSourceRange getNextSourceRange getSourceRange	Allows iterative or direct access to all the table items. Each accessed item is available via the igmpSourceRange command.
delSourceRange	Deletes a source range that has been accessed directly or iteratively.
generateStreams	Generates traffic streams for this group range.
setSourceRange	Allows the configuration values for a source range to be overwritten on the fly.

igmpSourceRange

The *igmpSourceRange* object configures a set of IPv4 source addresses that a host wishes to receive multicast traffic from. See igmpSourceRange for full details. The important options of this command are:

igmpSourceRange Options

Member	Usage
sourceIpFrom	The first IP address in the source range.
count	The number of IP addresses in the source range.

MLD

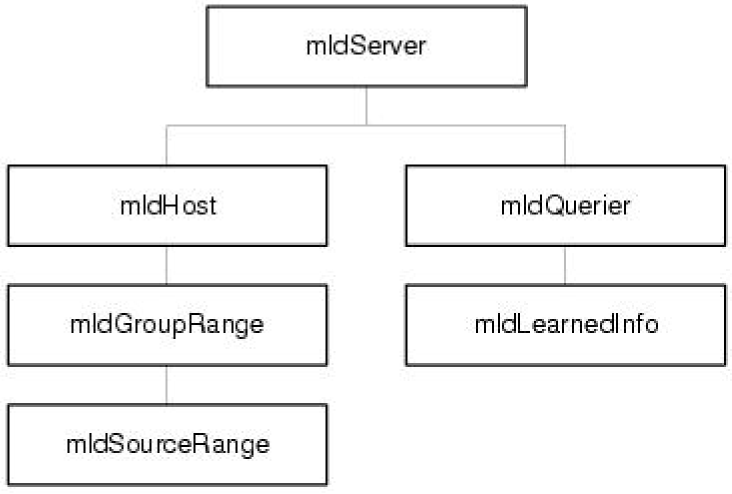
The MLD implementation is very close to the IGMPvX implementation — IGMP deals with IPv4 multicast and MLD deals with IPv6 multicast. Refer to IGMP (New) for a full discussion of the IGMP implementation and *Ixia Reference Manual, Theory of Operations: Protocols* chapter for an MLD overview.

The MLD related commands are:

- mldServer — configures overall operation of the protocol server's MLD operation.
- mldQuerier — list of simulated MLD Queriers.
- mldQuerierLearnedInfo — information learned by the Querier.
- mldHost — a list of simulated hosts which received multicast traffic.
- mldGrpRange — holds a list of multicast addresses that a particular host is interested in.
- mldSrcRange — holds a list of source IPv4 addresses that multicast traffic should be included from or excluded from.

These commands and the data that they maintain are arranged in a hierarchy as shown in the following figure.

MLD Command Hierarchy



mldServer

The *mldServer* object configures the overall operation of the MLD protocol server. See mldServer for full details.

The important options and subcommands of this command are:

mldServer Options

Member	Usage
numGroups timePeriod	Provides a means of throttling the amount of MLD traffic generated by the port.
mldv2ReportType	The version of the MLD report to be generated.
enableSendDoneOnStop	If true, enables the Send Done's on Stop feature. *(True / False)*.

mldServer Subcommands

Member	Usage
select	Must be used first to select the port being configured.
clearAllHosts	Removes all hosts from the list of hosts.
addHost	Adds a host to the list. The host must have been previously configured through the use of the mldHost command.
getFirstHost getNextHost getHost	Allows iterative or direct access to all the table items. Each accessed item is available via the mldHost command.
delHost	Deletes a host that has been accessed directly or iteratively.
generateStreams	Generates traffic streams for all of the hosts.
setHost	Allows the configuration values for a host to be overwritten on the fly.
clearAllQueriers	Removes all queriers from the list of queriers.
addQuerier	Adds a querier to the list. The querier must have been previously configured through the use of the mldQuerier command.
getQuerier getFirstQuerier getNextQuerier	Allows iterative or direct access to all the table items. Each accessed item is available via the mldQuerier command.
setQuerier	Allows the configuration values for a querier to be overwritten on the fly.
write	Sends any changes made with the MLD suite of commands to the protocol server for immediate application. This command **must** be used in order for the changes to have an effect.

mldQuerier

The *mldQuerier* object describes an emulated MLD Querier. See mldQuerier for full details. The important options and subcommands of this command are:

mldQuerier Options

Member	Usage
enable	Enables the use of the emulated querier in the MLD simulation. *(True / False)*
version	Indicates the MLD protocol version to be used. One of: • mldQuerierVersion1 • mldQuerierVersion2
startupQueryCount	The number of MLD General Query messages sent at startup. (Integer) (Default = 2)
generalQueryInterval	The amount of time (in seconds) between MLD General Query messages sent by the router. (Default = 125)
robustnessVariable	Defines the subnet vulnerability to lost packets. MLD can recover from robustness variable minus 1 lost MLD packets. The robustness variable should be set to a value of 2 or greater. (Default = 2)
genQueryResponseInterval	The maximum amount of time (in seconds) that the MLD querier waits to receive a response to a General Query mes-

mldQuerier Options

Member	Usage
	sage. (Default = 10 seconds, and must be less than the Query Interval)
specQueryResponseVariable	The maximum amount of time (in seconds) that the MLD querier waits to receive a response to a Specific Query message. (Default = 10 seconds, and must be less than the Query Interval)
specQueryTransmissionCount	Indicates the total number of Specific Query messages sent every Specific Query Response Interval (in seconds) before assuming that there is no interested listener for the particular group/source.
discardLearnedInfo	When disabled, the emulated Querier maintains a complete record state for received reports and sent queries (based on timer expiry for received groups and sources). (Default is disabled). When enabled, the Querier does not maintain any database and only sends periodic General Queries. The Specific Query group/source record information is not calculated based on any earlier received report, but solely based on the last received report.
supportElection	Indicates whether the Querier participates in Querier election or not. If disabled, then all incoming Query messages are discarded.
supportOlderVersionHost	Indicates whether the Querier will comply with RFC 3376 Section 7.3.2 and RFC 3810 Section 8.3.2. If disabled, all membership reports with version less than the current version are discarded.
supportOlderVersionQuerier	Indicates whether the Querier downgrades to the lowest version of received Query messages. If disabled, all Query messages with version less than the current version are discarded.
enableRouterAlert	If enabled, sets the "Send Router Alert" bit in the IP header.
Learned Info	
isQuerier	*(Read-only)* If true, indicates that the currently-elected querier is self. If false, indicates that the currently-elected querier is other. (*True / False*)
querierAddress	(Read-only) Indicates the IPv6 address of the currently-elected querier. (String)
querierWorkingVersion	(Read-only) Indicates the working version of the MLD querier at that point in time. (Integer).

mldQuerier Subcommands

Member	Usage
requestLearnedInfo	Requests the learned MLD information for the respective MLD Querier.

mldQuerier Subcommands

Member	Usage
getLearnedInfoList	Retrieves the list of learned info.
getFirstLearnedInfo getNextLearnedInfo	Retrieves the first learned info entry, then iterates through the list of additional entries.
setDefault	Sets default values for all configuration options.

mldQuerierLearnedInfo

The *mldQuerierLearnedInfo* object describes information learned by the MLD Querier. See mldQuerierLearnedInfo for full details. The important options and subcommands of this command are:

mldQuerierLearnedInfo Options

Member	Usage
groupAddress	The IPv6 address for the router group. (IPv6-format address)
groupTimer	The number of seconds remaining in the group address timer. (Integer)
compatibilityTimer	The number of seconds remaining in the compatibility timer. (Integer)
filterMode	Whether this group address is included or excluded. One of: • MLD_GROUPMODE_INCLUDE • MLD_GROUPMODE_EXCLUDE
compatibilityMode	What version of MLD this group address is. One of: • MLDV1 • MLDV2
sourceAddress	The IPv6 address for the group source. (IPv6-format address)
sourceTimer	The number of seconds remaining in the source address timer. (Integer)

mldQuerierLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.

mldHost

The *mldHost* object describes a simulated MLD host. See mldHost for full details.

The important options and subcommands of this command are:

mldHost Options

Member	Usage
enable	Enables the use of the host in the MLD simulation
enableGeneralQuery enableGroupSpecific	Enables the simulation for the host to respond to general and/or group specific queries.
enableRouterAlert	Sets the router alert bit in listener report messages.
enableUnsolicited reportFrequency	If set, causes the simulation for this host to send unsolicited listener report messages at a particular frequency.

mldHost Options

Member	Usage
version	Sets the MLD version number that is to be simulated on the host: 1 or 2.
protocolInterface Description	The name of the defined *interfaceEntry* which describes the host interface to be simulated.

mldHost Subcommands

Member	Usage
clearAllGroupRanges	Removes all group ranges from the list of group ranges.
addGroupRange	Adds a group range to the list. The group range must have been previously configured through the use of the mldGrpRange command.
getFirstGroupRange getNextGroupRange getGroupRange	Allows iterative or direct access to all the table items. Each accessed item is available via the mldGrpRange command.
delGroupRange	Deletes a group range that has been accessed directly or iteratively.
generateStreams	Generates traffic streams for all of the group ranges.
setGroupRange	Allows the configuration values for a group range to be overwritten on the fly.

mldGrpRange

The *mldGroupRange* object configures a set of multicast addresses that a host wishes to receive traffic from. See mldGroupRange for full details.

The important options and subcommands of this command are:

mldGroupRange Options

Member	Usage
enable	Enables the use of the group range in the MLD simulation.
groupIpFrom groupCount incrementStep	Specifies the set of IPv6 multicast addresses in the group range.
enablePacking recordsPerFrame sourcesPerRecord	If enabled, this option controls how many multicast records and sources will be included in each listener report for this group range.
sourceMode	Indicates whether the associated source range is a set of IP addresses to be included or excluded.

mldGroupRange Subcommands

Member	Usage
clearAllSourceRanges	Removes all source ranges from the list of source ranges.
addSourceRange	Adds a source range to the list. The source range must have been previously configured through the use of the mldSrcRange command.
getFirstSourceRange getNextSourceRange getSourceRange	Allows iterative or direct access to all the table items. Each accessed item is available via the mldSrcRange command.

mldGroupRange Subcommands

Member	Usage
delSourceRange	Deletes a source range that has been accessed directly or iteratively.
generateStreams	Generates traffic streams for this group range.
setSourceRange	Allows the configuration values for a source range to be overwritten on the fly.

mldSrcRange

The *mldSourceRange* object configures a set of IPv6 sources addresses that a host wishes to receive multicast traffic from. See mldSourceRange for full details.

The important options of this command are:

mldGroupRange Options

Member	Usage
sourceIpFrom	The first IP address in the source range.
count	The number of IP addresses in the source range.

BGP4

NOTE The BGP4 related commands reflect a new API. Older commands, options, and sub-commands are deprecated. Although they are deprecated, tests written with the older API will continue to work; refer to the Ixia Tcl Development Guide for Release 3.55 for a description of that API. The commands which are deprecated as a whole include:

- bgp4ExternalNeighborItem
- bgp4ExternalTable
- bgp4InternalNeighborItem
- bgp4InternalTable
- bgpStatsQuery

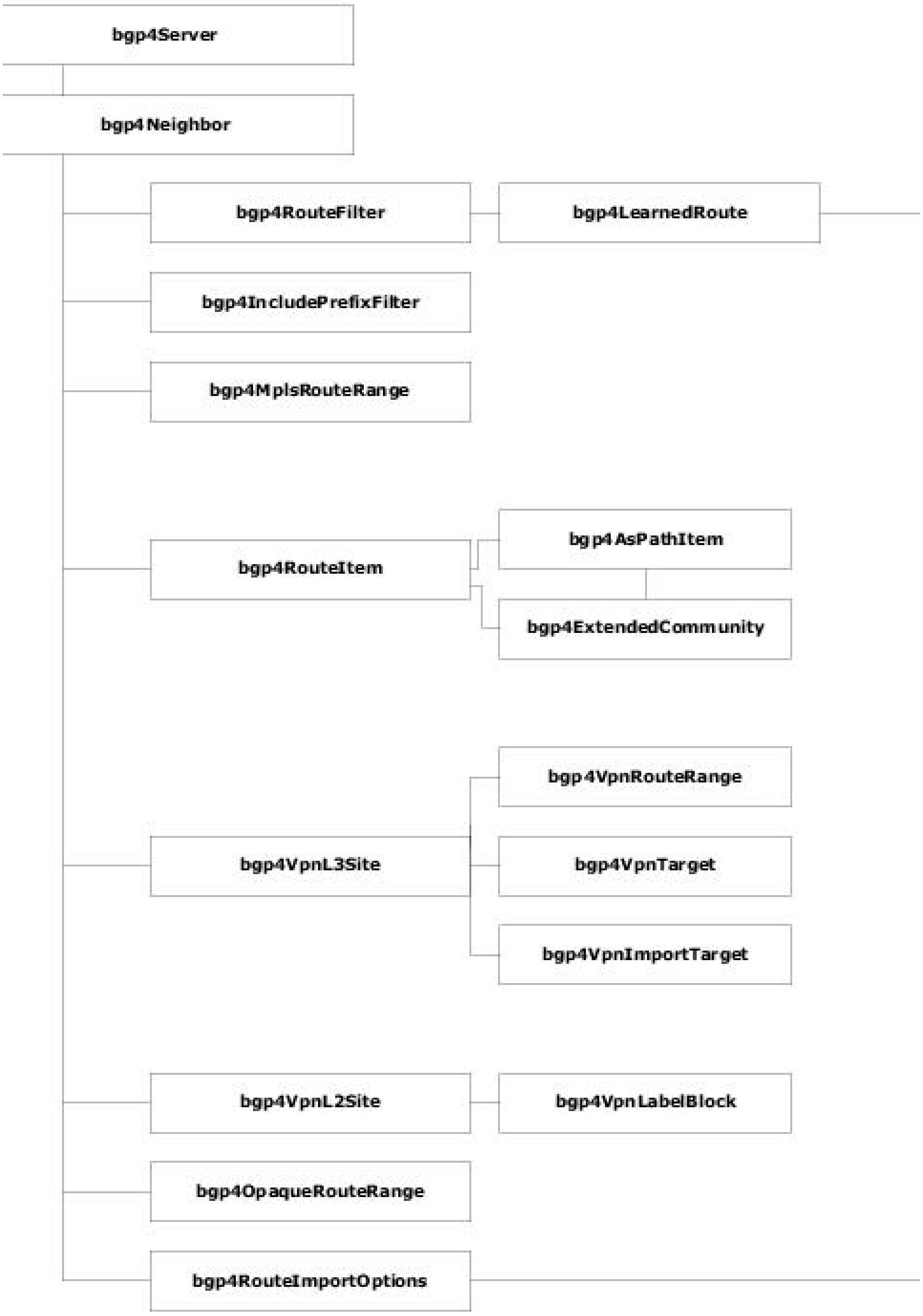
Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to BGP4. The BGP4-related commands are:

- bgp4Server — provides access to the BGP4 part of a port's protocol server. Includes the list of simulated internal and external servers.
- bgp4Neighbor — holds all of the information related to an internal or external simulated server.
- bgp4RouteItem — an individual route range and associated route attributes. More complex list based attributes are held in a list of bgp4AsPathItemitems.
- bgp4AsPathItem — represents list-based route attributes, including AS Set and AS Sequence.
- bgp4ExtendedCommunity — represents an extended community attribute associated with a route item.
- bgp4MplsRouteRange — represents a route range that also advertises an MPLS label and is associated with a route item.
- bgp4RouteFilter — sets up filtering conditions for retrieving learned routes.
- bgp4LearnedRoute — holds the contents of learned routes.
- bgp4IncludePrefixFilter — holds a single prefix filter to be used in filtering which learned routes are retained.
- bgp4VpnL3Site — represents a VPN layer 3 site to be used with internal neighbors.
- bgp4VpnRouteRange — represents a route range present at a VPN site.
- bgp4VpnTarget — represents a L3 VPN target attribute to be associated with routes advertised from a VPN site.
- bgp4VpnL2Site — represents a VPN layer 2 site.
- bgp4VpnLabelBlock — holds the labels to be used with an L2 VPN.
- bgp4StatsQuery — provides access to BGP4-related statistics.
- bgp4OpaqueRouteRange — treats imported route information as opaque data.
- bgp4routeImportOption — adds options for route import.
- bgp4VpnBgpAdVplsRange — adds options for AdVpls range.
- bgp4McastSender Site — adds options for multicast sender.

- bgp4McastReceiver Site — adds options for multicast receiver.
- bgp4UserDefinedAfiSafiRoute — adds options for afi/safi.

These commands, and the data that they maintain are arranged in a hierarchy, as shown in the following figure.

BGP4 Command Hierarchy



The following features are not available on older, non-Linux based load modules:

- IBGP AS setting on a per peer level.
- Next hop increment per route in the route range.
- Specify capabilities option for an advanced neighbor range.

bgp4Server

The bgp4Server command is necessary in order to access the BGP4 component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
bgp4server select 1 3 4
```

will access the BGP4 server for chassis 1, card 3, port 4. Once a BGP4 simulation has been set up with the other commands in this section, streams can be automatically generated which sends traffic to all IP addresses in all defined route ranges. This is done through the use of the *generateStreams* subcommand. See bgp4Server for full details. The important options and subcommands of this command are:

bgp4Server Options

Neighbor Type	Member	Usage
Internal	internalLocalASNum	The Autonomous System number associated with the routers participating in the Internal BGP (IBGP) group.
	internalRetries	The number of times to attempt an OPEN connection with the DUT router(s) before giving up.
	internalRetryDelay	When retries are necessary, the delay between retries.
	enableInternalActive Connect	Causes a Hello message to be actively sent when BGP testing starts.
	triggerVplsPwInitiation	Enables the BGP-LDP communication.
External	externalRetries	The number of times to attempt an OPEN connection with the DUT router(s) before giving up.
	externalRetryDelay	When retries are necessary, the delay between retries.
	enableExternalActive Connect	Causes a Hello message to be actively sent when BGP testing starts.

bgp4Server Subcommands

Member	Usage
clearAllNeighbors	Removes all neighbors from the table.
addNeighbor	Adds a neighbor to the table. The neighbor must have been previously configured through the use of the bgp4Neighbor command.
getFirstNeighbor getNextNeighbor getNeighbor	Allows iterative or direct access to all the table items. Each accessed item is available via the bgp4Neighbor command.
delNeighbor	Deletes a neighbor that has been accessed directly or iteratively.
generateStreams	Generates traffic streams for all of the neighbors.
setNeighbor	Allows the configuration values for a neighbor to be overwritten on the fly.

bgp4Neighbor

The bgp4Neighbor command holds information about a BGP4 internal or external neighbor router. The router may be either a IPv4 or IPv6 router; the type (IPv4 or IPv6) of the router is dictated by the *ipType* option.

In addition to a number of options related to the neighbor itself, this command holds two lists:

- A list of route ranges. A bgp4RouteItem is added to the neighbor with the *addRouteRange* subcommand.
- A list of MPLS route ranges. A bgp4MplsRouteRange is added to the neighbor with the *addMplsRouteRange* subcommand.
- A list of L3 VPN sites. A bgp4VpnL3Site is added to the neighbor with the *addL3Site* subcommand. Only **internal** neighbors may have L3 VPN sites.
- A list of L2 VPN sites. A bgp4VpnL2Site is added to the neighbor with the *addL2Site* subcommand. Only **internal** neighbors may have L2 VPN sites.
- A list of opaque route ranges. A bgp4OpaqueRouteRange information is imported with the *bgp4OpaqueRouteRange* subcommand.
- A list of route import options. A bgp4routeImportOption is added to the neighbor with the *routeImportOption* subcommand.
- A list of BGP AD VPLS Range options. A bgp4VpnBgpAdVplsRange is added to the neighbor.

When all route ranges and L3 or L2 sites are added to the *bgp4Neighbor*, then the neighbor is added to the bgp4Server as either an internal or external neighbor with the *bgp4ServeraddNeighbor* command.

Routes learned from a network are available through the use of the *requestLearnedRoutes* and *getLearnedRoutesList*. The types of routes learned are controlled through the use of the bgp4RouteFilter and bgp4IncludePrefixFilter commands. The latter command is used to establish a single prefix filter, added to a prefix filter list in this command using the *addFilter* subcommand; the *enablePrefixFilter* option must be *true* in order for the prefix list to be used.

See bgp4Neighbor for full details. The important options and subcommands of this command are:

bgp4Neighbor Options

Member	Usage
type	Indicates that the neighbor is either an internal or external router.
ipType	Indicates whether the neighbor is a IPv4 or IPv6 router.
enable	Enables or disables simulation of the router.
enableRouterId routerId	The router ID used in NEXT_HOP messages.
enableBgpId bgpId	The BGP ID used in OPEN messages.
localIpAddress rangeCount	The first IP address for the simulated neighbor routers and the number of routers.
externalNeighborASNum	(This option has been deprecated.)
enable4ByteAsNumber	Enables the 4-byte autonomous system number.
localASNumber	(External only) The first AS Num assigned to the simulated neighbor router. May be set for external neighbors on any port type, but only Linux-based ports may set this for internal neighbors.

bgp4Neighbor Options

Member	Usage
asNumMode	(External only) Indicates that each new session uses a different AS number.
dutIPAddress	The IP address of the DUT router.
holdTimer	The period of time between KEEP-ALIVE messages sent to the DUT.
updateInterval	The frequency with which UPDATE messages are sent to the DUT.
enableLinkFlap linkFlapUpTIme linkFlapDownTime	Controls flapping of the link between the simulated routers and the DUT, including the period and downtime.
enableStaggeredStart staggeredStartPeriod	Controls the staggering and period of initial start messages.
enableOptionalParameters	Controls how an OPEN is conducted in the presence of optional parameters.
enableNextHop	Used for IPv4 traffic. Controls the use of the NEXT_HOP attribute. *(default = disabled)*
nextHop	If *enableNextHop* is *true,* this is the IPv4 address used as the next hop. *(default = 0.0.0.0)*
authenticationType	The cryptographic authentication type used by the neighbor, one of: NULL (no authentication) or MD5. When MD5 is used, an *MD5Key* must be configured by the user.
md5Key	Used with MD5 authentication. A user-defined string; maximum = 255 characters.
enableBfdRegistration	Indicates if a BFD session is to be created to the BGP peer IP address once the BGP session is established. This allows BGP to use BFD to maintain IPv4 connectivity with the BGP peer.
bfdModeOfOperation	Indicates whether to use a single-hop or a multi-hop mode of operation for the BFD session being created with a BGP peer.
tcpWindowSize	*(External neighbor only)* The TCP window used for communications from the neighbor. *(default = 8,192)*
numUpdatesPerIteration	When the protocol server operates on older ports that do not possess a local processor, this tuning parameter controls how many UPDATE messages will be sent at a time. When many routers are being simulated on such a port, changing this value may help to increase or decrease performance. *(default = 1)*
enableGracefulRestart restartTime staleTime enableActAsRestarted	Controls the operation of BGP Graceful Restart.
enableIpV4Mdt	Indicates that BGP will use a new SAFI called the MDT-SAFI (*value 66*) to carry the Data-MDT group address (*IPv4*) in the MP_REACH_NLRI field of the update-packet, instead of using an external-community.
enableIpV4Mpls	Controls the advertisement of these optional parameters in

bgp4Neighbor Options

Member	Usage
enableIpV4MplsVpn enableIpV4Multicast enableIpV4MulticastVpn enableIpV4Unicast enableIpV6Mpls enableIpV6MplsVpn enableIpV6Multicast enableIpV6MulticastVpn enableIpV6Unicast	OPEN statements.
enablePrefixFilter	Enables the use of the prefix filter list, which limits the routes learned.
enableVpls	Enables the use of BGP for setting up a VPLS test topology.

bgp4Neighbor Subcommands

Class	Member	Usage
Route Ranges	clearAllRouteRanges	Removes all route ranges from the table.
	addRouteRange	Adds a range to the neighbor. The route range must have been previously configured using the bgp4RouteItem command.
	getFirstRouteRange getNextRouteRange getRouteRange	Allows iterative or direct access to all route ranges. Each item is available via the bgp4RouteItem command.
	delRouteRange	Deletes a route range either directly or iteratively accessed.
	setRouteRange	Allows a route range's configuration to be overwritten on the fly.
MPLS Route Ranges	clearAllMplsRouteRanges	Removes all MPLS route ranges from the table.
	addMplsRouteRange	Adds an MPLS range to the neighbor. The route range must have been previously configured through the use of the bgp4MplsRouteRange command.
	getFirstMplsRouteRange getNextMplsRouteRange getMplsRouteRange	Allows iterative or direct access to all the MPLS route ranges. Each accessed item is available via the bgp4MplsRouteRange command.
	delMplsRouteRange	Deletes an MPLS route range either directly or iteratively accessed.
	setMplsRouteRange	Allows a MPLS route range's configuration to be overwritten on the fly.
BGP AD VPLS Ranges	addBgpAdVplsRange	Adds a BGP Ad VPLS Range to the BGP4 Neighbor.
	delBgpAdVplsRange	Deletes a BGP Ad VPLS Range to the BGP4 Neighbor.

bgp4Neighbor Subcommands

Class	Member	Usage
	getBgpAdVplsRange	Allows to get a BGP Ad VPLS Range to the BGP4 Neighbor.
	setBgpAdVplsRange	Allows to set a BGP Ad VPLS Range to the BGP4 Neighbor.
	getFirstBgpAdVplsRange	Allows to get the first BGP Ad VPLS Range in the BGP4 Neighbor.
	getNextBgpAdVplsRange	Allows to get the next BGP Ad VPLS Range in the BGP4 Neighbor.
	clearAllBgpAdVplsRanges	Clears all BGP Ad VPLS Range in the BGP4 Neighbor.
VPN L3 Sites	clearAllL3Sites	Clears the L3 VPN sites associated with the neighbor.
	addL3Site	Adds an L3 VPN site to the neighbor. The L3 VPN site must have been previously configured through the use of the bgp4VpnL3Site command.
	getL3Site getFirstL3Site getNextL3Site	Allows direct or iterative access to all the L3 VPN sites. Each accessed item is available via the bgp4VpnL3Site command.
	delL3Site	Deletes a L3 VPN site.
	setL3Site	Allows a L3 VPN site's configuration to be overwritten on the fly.
VPN L2 Sites	clearAllL2Sites	Clears the L2 VPN sites associated with the neighbor.
	addL2Site	Adds a L2 VPN site to the neighbor. The VPN site must have been previously configured through the use of the bgp4VpnL2Site command.
	getL2Site getFirstL2Site getNextL2Site	Allows direct or iterative access to all of the L2 VPN sites. Each accessed item is available via the bgp4VpnL2Site command.
	delL2Site	Deletes an L2 VPN site.
	setL2Site	Allows an L2 VPN site's configuration to be overwritten on the fly.
Learned Routes	addPrefixFilter clearAllPrefixFilters getFirstPrefixFilter getNextPrefixFilter delPrefixFilter	Controls additions and access to the prefix filter list.
	requestLearnedRoutes getLearnedRouteList	Requests the list of routes and then retrieves them. Filter settings must be previously set in bgp4RouteFilter.
Misc	setDefault	Sets default values for all configuration

bgp4Neighbor Subcommands

Class	Member	Usage
		options.
	generateStreams	Generates a data stream for this particular neighbor, as opposed to all neighbors when using the corresponding subcommand in the bgp4Server command.
opaqueRouteRange	delOpaqueRouteRange	Deletes the specified opaque route range.
	setOpaqueRouteRange	Overwrites the specified route range as opaque route range.
	getFirstOpaqueRouteRange	Retrieves the first opaque route range as the current item.
	getNextOpaqueRouteRange	Retrieves the adjacent opaque route range as the current item.
	clearAllOpaqueRouteRange	Clears all the associated opaque route range information.
routeImportOptions	addRouteImportOptions	Adds the specified route import options.
	delRouteImportOptions	Deletes the specified route import options.
	getRouteImportOptions	Retrieves the specified route import options.
	setRouteImportOptions	Overwrites the specified route import options to the route range.
	getFirstRouteImportOptions	Retrieves the first route import option as the current item.
	getNextRouteImportOptions	Retrieves the next route import options as the current item.
	clearAllRouteImportOptions	Clears all the associated route import options.

bgp4RouteItem

The *bgp4RouteItem* holds a route range that is associated with a bgp4Neighbor, bgp4MplsRouteRange or bgp4VpnRouteRange. This command defines a set of routes and associated attributes.

This command defines a set of routes and associated attributes. Two items require lists:

- An AS path item — built in the bgp4AsPathItem command and included using the *addASPathItem* subcommand.
- An extended community item — built with the bgp4ExtendedCommunity command and included using the *addExtendedCommunityList* subcommand.

See bgp4RouteItem for full details. The important options and subcommands of this command are:

bgp4RouteItem Options

Category	Member	Usage
Route Ranges	networkAddress fromPrefix thruPrefix numRoutes fromPacking thruPacking iterationStep enableGenerateUniqueRoutes enableIncludeLoopback enableIncludeMulticast enableProperSafi delay	Controls the range and number of network prefix addresses generated as well as how many are packed in each UPDATE message.
Flapping	enableRouteFlap routeFlapTime routeFlapDropTime flapFrom flapTo	Enables and controls the flapping of routes to the DUT.
Next Hop	enableNextHop nextHopIpAddress nextHopMode	Generates the NEXT HOP attribute.
Origin	enableOrigin originProtocol	Generates the ORIGIN attribute.
Local Preference	enableLocalPref localPref	Generates the LOCAL PREF attribute.
Multi-Exit Discriminator	enableMED MED	Generates the MULTI-EXIT DISCRIMINATOR attribute.
Communities	enableCommunity communityList	Generates a community list.
Aggregator	enableAtomicAggregate enableAggregator aggregatorIPAddress	Generates attributes related to route aggregation: • ATOMIC AGGREGATOR • AGGREGATOR

bgp4RouteItem Options

Category	Member	Usage
	aggregatorASNum aggregatorIDMode	
Originator	enableOriginatorId originatorId	Generates the ORIGINATOR attribute.
Clusters	enableCluster clusterList	Generates a cluster list.
AS Path	enableASPath asPathSetMode	Enables the use of the AS Path attribute and its contents.

bgp4RouteItem Subcommands

Category	Member	Usage
AS Path Item	clearASPathList	Clears the AS Path list associated with the route item.
	addASPathItem	Adds an AS path item to the route item. The AS path item must have been previously configured through the use of the bgp4AsPathItem command.
	getASPathItem getFirstASPathItem getNextASPathItem	Allows direct or iterative access to all the AS Path items. Each accessed item is available via the bgp4AsPathItem command.
Extended Community Item	clearExtendedCommunityList	Clears the extended community list associated with the route item.
	addExtendedCommunity	Adds an extended community item to the route item. The extended community item must have been previously configured through the use of the bgp4ExtendedCommunity command.
	getExtendedCommunity getFirstExtendedCommunity getNextExtendedCommunity	Allows direct or iterative access to all the extended community items. Each accessed item is available via the bgp4ExtendedCommunity command.

bgp4AsPathItem

The bgp4AsPathItem command is used to construct AS list related items. These items must be added to a route item through the use of the bgp4RouteItem *addASPath item*command. See bgp4AsPathItem for full details. The important options of this command are:

bgp4AsPathItem Options

Member	Usage
enableAsSegment	Indicates that this particular list is enabled.
asList	The list of AS numbers.
asSegmentType	The type of AS list in the item.

bgp4ExtendedCommunity

The bgp4ExtendedCommunity command is used to construct an extended community attribute for a route item. This community must be added to an route item through the use of the bgp4RouteItem *addExtendedCommunity*command. See bgp4ExtendedCommunity for full details. The important options of this command are:

bgp4ExtendedCommunity Options

Member	Usage
type	The high-order type byte.
subType	The low-order type byte.
value	The remaining six value bytes of the attribute.

bgp4MplsRouteRange

The *bgp4MplsRouteRange* command holds a route range that is associated with a bgp4Neighbor command. It includes all of the options and subcommands of the bgp4RouteItem command as well as additional MPLS labels which designate a label mapping for the route range.

The labels generated by the options are used to generate corresponding MPLS labels. Each iteration through the route range is matched to an iteration through the label range.

NOTE Only the additional label related options are described for this command. Refer to bgp4RouteItem for the remainder of the options.

See bgp4MplsRouteRange for full details. The important options of this command are:

bgp4MplsRouteRange Options

Member	Usage
labelMode	Indicates whether all MPLS routes receive the same or unique labels.
labelStart labelEnd labelStep	If unique labels are generated, this indicates the lower and upper bound of the generated labels, along with the step size between labels.
labelSpaceId	The label space ID associated with the MPLS labels.
clearASPathList	Clears the AS Path list associated with the route item.

bgp4RouteFilter

The bgp4RouteFilter command is used to set up the types of learned routes that will be retrieved by the *requestLearnedRoutes* and *getLearnedRoutes* subcommands of the bgp4Neighbor command. The options in bgp4RouteFilter must be set before a bgp4Server *addNeighbor*command. The options enable the filtering of routes for one or more addressing types. They may be changed on the fly by using bgp4Server *setNeighbor*. See bgp4RouteFilter for full details. The important options of this command are:

bgp4RouteFilter Options

Member	Usage
enableIpV4Unicast enableIpV4Mdt enableIpV4Multicast enableIpV4MulticastVpn	Enables filtering of routes related to IPV4.

bgp4RouteFilter Options

Member	Usage
enableIpV4Mpls enableIpV4MplsVpn	
enableIpV6Unicast enableIpV6Multicast enableIpV6MulticastVpn enableIpV6Mpls enableIpV6MplsVpn	Enables filtering of routes related to IPV6.
enableAdditional afi, safi	Allows the filter to be specified any AFI (address family indicator) and SAFI (sub-AFI).

bgp4LearnedRoute

bgp4Learned Route is used to retrieve the filtered list of learned routes from the *requestLearnedRoutes* and *getLearnedRouteList* commands of bgp4Neighbor. Learned routes route filtering is set up with bgp4RouteFilter. Each enabled type of route is considered a separate list and must be retrieved with separate calls to the *getFirst/getNext* subcommands. See bgp4LearnedRoute for full details. The important options and subcommands of this command are:

bgp4LearnedRoute Options

Member	Usage
description	A textual description including all of the other items.
neighbor	The local IP address of the neighbor.
routeDistinguisher	The route distinguisher for the route used for IPv4 and IPv6 MPLS VPNs.
label	MPLS label for IPV4 and IPv6 MPLS related routes.
ipAddress	IP prefix for the route.
prefixLength	IP prefix length for the route.
neighborAddress	IP prefix for the route.
remotePeAddress	IP prefix for the route.
remoteVplsId	String prefix for the route.
supportedLocally	The displaying whether VPLS is supported locallly.
remoteVsiId	PE Address or Assigned Number.
routeDistinguisher	IP or AS prefix for the route.
routeTarget	IP or AS prefix for the route
nextHopAddress	IP prefix for the route
peerAddress	The peer address in IP format.
vplsId	The VPLS ID in IP or AS format.
sourceAii	The 4 byte unsigned number of the Source AII.
targetAii	The 4 byte unsigned number of the Target AII.
groupId	The 4 byte unsigned number of the Group Id.
label	The 4 byte unsigned number of the label.
pwState	The boolean value of the PW State.
localPwSubState	The 4 byte unsigned number of the local PW Sub State.

bgp4LearnedRoute Options

Member	Usage
remotePeSubState	The 4 byte unsigned number of the Remote PE Sub State.
cBit - boolean	The boolean value of the C Bit.
mtu	The 2 byte value for the maximum Transmission Unit (MTU).
neighborAddress	The descriptive identifier for the BGP neighbor.
nextHopAddress	A 4-octet IP address which indicates the next hop.
remotePeAddress	The descriptive identifier for the remote PE.
remoteVplsId	The remote VPLS ID indicated by an IP or AS.
remoteVsiId	The remote VSI Id indicated by 4 bytes unsigned number.
routeDistinguisher	The route distinguisher indicated by the IP or AS number.
routeTarget	The route target indicated by the IP or AS number.
supportedLocally	The boolean value indicating whether it is supported locally.

bgp4LearnedRoute Subcommands

Member	Usage
getFirst getNext	Iterate through the list for a particular route type.

bgp4IncludePrefixFilter

The *bpg4IncludePrefixFilter* command is used to filter the learned routes associated with a BGP neighbor. The options in this command are added to the prefix filter list in the bgp4Neighbor command using the *addPrefixFilter* subcommand. Refer to bgp4LearnedRoute for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

See bpg4IncludePrefixFilter for full details. The important options of this command are:

bgp4IncludePrefixFilter Options

Member	Usage
enableExactPrefix	Controls whether a loose or exact prefix match is performed for the entry.
addressType	IPv4 or IPv6.
firstPrefix maskWidth numPrefixes	The prefixes to be included.
asSegmentType	The type of AS list in the item.

bgp4VpnL3Site

The *bgp4VpnL3Site* command holds information about a VPN Layer 3 site. A site is set of networks connected to an internal neighbor interface. Sites specified with this command are added to internal neighbors using the bgp4Neighbor *addL3Site* command.A VPN Layer 3 site includes three lists:

- A list of VPN route ranges. These route ranges are advertised to other routers.
- A list of VPN targets, which will receive routing tables.
- A list of VPN import targets, which are used to filter incoming route tables.

See bgp4VpnL3Site for full details. The important options and subcommands of this command are:

bgp4VpnL3Site Options

Member	Usage
enable	Enables or disables use of the VPN site.
enableVpnMulticast groupAddress enableCluster clusterList distinguisherType distinguisherAsNumber distinguisherIpAddress distinguisherAssignedNumber	Enables and controls the use of Multicast VRFs (MVRFs).
tunnelType includePmsiTunnelAttribute rsvpP2mpId rsvpTunnelId useUpstreamAssignedLabel	Enables and controls the use of Multicast VPNs.
mplsAssignedUpstreamLabel	S-PMSI A-D route is sent with this Upstream Label. This is applicable only if UseUpstreamAssignedLabel is true.
vrfCount	Number of VRFs within the VRF Range.
multicastGroupAddressStep	The increment step to be added to each additional Multicast Group Address.

bgp4VpnL3Site Subcommands

Class	Member	Usage
VPN Route Ranges	clearAllVpnRoute Ranges	Removes all route ranges from the table.
	addVpnRouteRange	Adds a range to the site. The route range must have been previously configured through the use of the bgp4VpnRouteRange command.
	getFirstVpnRouteRange getNextVpnRouteRange getVpnRouteRange	Allows iterative or direct access to all the route ranges. Each accessed item is available via the bgp4VpnRouteRange command.
	delVpnRouteRange	Deletes a route range either directly or iteratively accessed.
	setVpnRouteRange	Allows a VPN route range's configuration to be overwritten on the fly.
VPN L3 Sites	clearAllVpnTargets	Clears the VPN targets associated with the site.
	addVpnTarget	Adds a VPN target to the site. The target must have been previously configured through the use of the bgp4VpnTarget command.
	getVpnTarget getFirstVpnTarget getNextVpnTarget	Allows direct or iterative access to all the VPN targets. Each accessed item is available via the bgp4VpnTarget command.

bgp4VpnL3Site Subcommands

Class	Member	Usage
	delVpnTarget	Deletes a VPN target.
Import Targets	clearAllImportTargets	Clears the import targets associated with the site.
	addImportTarget	Adds a import target to the site. The target must have been previously configured through the use of the bgp4VpnTarget command.
	getImportTarget getFirstImportTarget getNextImportTarget	Allows direct or iterative access to all the import targets. Each accessed item is available via the bgp4VpnTarget command.
	delImportTarget	Deletes an import target.
Learned Routes	requestLearnedRoutes getLearnedRouteList	Requests the list of routes and then retrieves them.

bgp4VpnTarget

The *bgp4VpnTarget* command holds information about a target attribute to be associated with VPN route ranges advertised by a L3 site. VPN targets are added to a bgp4VpnL3Site using the bgp4VpnL3Site *addVpnTarget* subcommand.

It also holds information on import targets used to filter received routes by an L3 site. Import targets are added to a bgp4VpnL3Site using the bgp4VpnL3Site *addImportTarget* subcommand. See bgp4VpnTarget for full details. The important options of this command are:

bgp4VpnTarget Options

Member	Usage
type	The type of target specified: AS or IP.
asNumber	For AS type targets, the associated AS number.
ipAddress	For IP type targets, the associated IP address.
assignedNumber	The assigned number for the target.

bgp4VpnRouteRange

The *bgp4VpnRouteRange* command holds a route range that is associated with a bgp4VpnL3Site. This command includes all of the options and commands of the bgp4RouteItem command and defines a set of routes and associated attributes.

NOTE

Only the additional label-related options are described for this command. Refer to bgp4RouteItem for the remainder of the options. Note that the following options in this command should be used in lieu of those in the bgp4RouteItem command: enableversus enableRouteRange, networkIpAddressversus networkAddressand toPrefixversus thruPrefix.

See bgp4VpnRouteRange for full details. The important options and subcommands of this command are:

bgp4VpnRouteRange Options

Category	Member	Usage
Enable	enable	Enables or disables the route range.
Route Ranges	networkIpAddress toPrefix	Controls the range and number of network prefix addresses generated.
Label	labelMode labelStart labelEnd labelStep labelSpaceId	The MPLS labels associated with advertised VPN routes.
Route Distinguisher	distinguisherMode distinguisherAsNumber distinguisherIpAddress	Specifies the route distinguisher to be

bgp4VpnRouteRange Options

Category	Member	Usage
	distinguisherAssigned Number distinguisherCount distinguisherStep distinguisherIpAddressStep distinguisherAsNumberStep distinguisherAssignedNumberStep distinguisherIpAddressStepAcrossVrfs distinguisherAsNumberStepAcrossVrfs distinguisherAssignedNumberStepAcrossVrfs distinguisherCountPerVrf	used when the route is advertised.
	routeStepAcrossVRFs	The route step across VRFs.

bgp4VpnRouteRange Subcommands

Member	Usage
clearASPathList	Clears the AS Path list associated with the route range.
addASPathItem	Adds an AS path item to the route range. The AS path item must have been previously configured through the use of the bgp4AsPathItem command.
getASPathItem getFirstASPathItem getNextASPathItem	Allows direct or iterative access to all the AS Path items. Each accessed item is available via the bgp4AsPathItem command.

bgp4VpnL2Site

The *bgp4VpnL2Site* command holds information about a VPN Layer 2 site. An L2 CE site is set of L2 switched networks connected to an internal neighbor interface. Sites specified with this command are added to internal neighbors using the bgp4Neighbor *addL2Site* command. A VPN Layer 2 site includes the following list:

- A list of Label Blocks, which contain labels used for demultiplexing VPLS traffic

See bgp4VpnL2Site for full details. The important options and subcommands of this command are:

bgp4VpnL2Site Options

Member	Usage
enable	Enables or disables use of the L2 VPN site.
siteId	The identifier for the L2 (CE) site. The default is 0.
enableCluster clusterList enableControlWord enableSequencedDelivery mtu	Enables and controls the use of L2 VPN VPLS.

bgp4VpnL2Site Options

Member	Usage
routeTargetType routeTargetAS routeTargetAssigned routeTargetIP routeDistinguisherType routeDistinguisherAsNumber routeDistinguisherAssigned routeDistinguisherIP	

bgp4VpnL2Site Subcommands

Class	Member	Usage
VPN Label Blocks	clearAllVpnLabelBlocks	Clears the VPN label blocks associated with the L2 site.
	addVpnLabelBlock	Adds a VPN label block to the site. The label block must have been previously configured through the use of the bgp4VpnLabelBlock command.
	getVpnLabelBlock getFirstVpnLabelBlock getNextVpnLabelBlock	Allows direct or iterative access to all of the label blocks. Each accessed item is available via the bgp4VpnLabelBlock command.
	delVpnLabelBlock	Deletes a label block.
	setVpnLabelBlock	Allows a VPN route range's configuration to be overwritten on the fly.
Learned Routes	requestLearnedRoutes getLearnedRouteList	Requests the list of routes and then retrieves them.
Misc	setDefault	Sets default values for all configuration options.

bgp4VpnLabelBlock

The *bgp4VpnLabelBlock* command holds information about labels related to an L2 VPN site. Label blocks specified with this command are added to BGP L2 VPN sites using the bgp4VpnL2Site *addVpnLabelBlock* command. See bgp4VpnLabelBlock for full details. The important options and subcommands of this command are:

bgp4VpnLabelBlock Options

Member	Usage
enable	Enables or disables use of the L2 VPN label block.
offset	The VPLS Block Offset value used to create a unique subset of the label values. *(default = 0)*
startBlock	The first label in the label block. *(default = 0)*
numberOfVpnLabels	The number of labels contained in the label block. *(default = 0)*

bgp4VpnLabelBlock Subcommands

Member	Usage
setDefault	Sets default values for all configuration options.

bgp4StatsQuery

The bgp4StatsQuery command is used to fetch statistics about a BGP4 session. The BGP4 server must be running in order to obtain statistics. Because of the large number of statistics and the large amount of data that might be generated, this command provides the means to limit the statistics to a set of statistics for a particular pairs of neighbor routers. See bgp4StatsQuery for full details. Also known by this name *bgpStatsQuery*, but this usage is deprecated. The important options and subcommands of this command are:

bgp4StatsQuery Options

Member	Usage
statName	The name of the statistics retrieved.
statValue	The value of the statistic.

bgp4StatsQuery Subcommands

Member	Usage
addNeighbor delNeighbor clearAllNeighbors	Sets up the neighbor pairs to monitor.
addStat delStat clearAllStats	Sets up the statistics to monitor
getStat	Fetches a particular statistics into the options.

bgp4OpaqueRouteRange

Large amount of information is imported from a text file. The route information in these files are generally real life information collected from the internet by the vendors. The imported route information is treated as opaque data and managed separately from the manually configured route ranges. The important options are:

bgpOpaqueRouteRange Options

Member	Usage
enabled	Enables the particular opaque route range.
importedFile	Location of the route import file.
interpretAsPath	The AsPath as present in the file is sent as AS-SET in the Update message.
sendMultiExitDiscovery	
noOfRoutes	Total number of opaque routes.
status	Indicates the status of the imported file.

The subcommands are:

bgpOpaqueRouteRange Subcommands

Member	Usage
applyOpaqueRouteRange	Applies the concerned opaque route.

bgp4routeImportOption

This object holds the different options for route import. The bgpimportRouteRange command is executed considering the arguments of the routeImportOptions object of the relevant importedRoutes. The important options are:

bgpRouteImportOption Options

Member	Usage
advertiseBestRoutes	If checked, only the best routes are imbibed and advertised. The sub-optimal routes are ignored.
configureAsPath	If checked, the AS Path as present in the file is sent in the Update message as AS-SET.
numberOfRoutesPerBlock	Represents the maximum number of routes that can be forwared in a block.
sendMultiExitDiscValue	If enabled, the BGP router sends the MED value of the attribute.
routeFileType	The file format of the import file.

The subcommands are:

bgpRouteImportOptions Subcommands

Member	Usage
import	Imports the route ranges from the file specified as argument.
getSupportedBGPRouteFileTypes	Displays the list of available route files formats.

bgp4VpnBgpAdVplsRange

This object holds the different options for BGP AD VPLS Range. See bgp4VpnBgpAdVplsRange for full details. The important options are:

bgp4VpnBgpAdVplsRange Options

Member	Usage
enable	Enables or disables simulation of the router.
vplsCount	Adds the integer value that indicates the number of VPLS instance emulated using this VPLS range.
routeTargetType	Sets the RT format to AS or IP.
routeTargetIpAddress	An IP value, available for use only if the IPv4 Input is set to IP.
routeTargetAsNumber	An integer value, available for use only if Distinguish Type is set to AS.
routeTargetAssigned Number	This is an integer value that is dependent on the routeTargetType.
routeTargetIpAddressStep	Available for use only if the IPv4 address is set to IP.
routeTargetAsNumberStep	This is an integer value available for use only if routeTargetType is set to AS.
routeTargetAssigned NumberStep	The target assigned number. this is an integer value that is dependent on the on the routeTargetType.
vplsIdType	The VPLS Id. The format is AS or IP.
vplsIdIpAddress	Available for use only if the route VPLS Id Type is set to IP.
vplsIdAsNumber	Available for use only if VPLS Id Type is set to AS.
vplsIdAssigned	The indicated number for thevplsIdAssignedNumber

bgp4VpnBgpAdVplsRange Options

Member	Usage
Number	attribute.
vplsIdIpAddressStep	Available for use only if the route vplsIdType is set to IP.
vplsIdAsNumberStep	Available for use only if vplsIdType is set to AS.
vplsIdAssigned NumberStep	The indicated value for the vplsIdType attribute.
useVplsIdAsRoute Distinguisher	Enables the VPLS Id as the route distinguisher.
routeDistinguisher Type	Sets the RD format to AS or IP.
routeDistinguisherIp Address	Available for use only if the rIPv4 Input is set to IP.
routeDistinguisherAsNumber	Available for use only if Distinguish Type is set to AS.
routeDistinguisher AssignedNumber	The distinguisher assigned number. this is an integer value.
routeDistinguisherIp AddressStep	Available for use only if the rIPv4 Input is set to IP.
routeDistinguisherAsNumberStep	Available for use only if Distinguish Type is set to AS.
routeDistinguisher AssignedNumberStep	The distinguisher assigned number.
vsiId	The VSI Id. This value is Concatenate Number or Concatenate PE Loopback Address.
vsiIdAssignedNumber	The indicated number for the vsiIdAssignedNumber attribute

bgp4McastSenderSite

This object holds the different options for BGP Multicast Sender Site. See bgp4McastSender-Site for full details. The important options are:

bgp4McastSenderSite Options

Member	Usage
enabled	Enables or disables the multicast sender site.
addressFamily	Indicates the IPv4/IPv6 interface id of the router.
startGroupAddress	The first IPv4 or IPv6 Multicast group address in the range of group addresses included in this Register message.
groupMaskWidth	The number of bits in the network mask used with the Group Address.
groupAddressCount	The number of group addresses to be included in the Register message.
sourceGroupMapping	Indicates the source group mapping. One of: • fullyMeshed • oneToOne

bgp4McastSenderSite Options

Member	Usage
startSourceAddress	The first IPv4 or IPv6 source address to be included in this Register message. (IPv4 Multicast addresses are not valid for sources.)
sourceMaskWidth	The number of bits in the mask applied to the Source Address. (The masked bits in the Source Address form the address prefix.) The default value is 32. The valid range is 1 to 128, depending on address family type. Used for (S,G) Type and (S,G, rpt) only
sourceAddressCount	The number of multicast source addresses to be included. The maximum number of valid possible addresses depends on the values of the Source Address and the Source Mask Width. The default value is 0.
sPmsiTrafficGroupId	Creates traffic using MPLS Labels of S-PMSI Tunnel and S-PMSI Upstream Assigned Label.
sPmsiRsvpP2mpId	The P2MP Id represented in IP address format.
sPmsiRsvpP2mpIdAsNumber	The P2MP Id represented in integer format.
sPmsiRsvpP2mpIdStep	Indicates the P2MP ID. This accepts only integer values.
sPmsiRsvpTunnelId	The first Tunnel ID value in the range of Tunnel IDs.
sPmsiRsvpTunnelIdStep	Indicates the P2MP ID. This accepts only integer values.
sPmsiTunnelCount	The total count of the S-PMSI RSVP Tunnel Count.
useUpstreamAssignedLabel	Indicates whether upstream label as configured be used or not. If this field is false, then MPLS Assigned Upstream Label and MPLS Assigned Upstream Label Step fields are disabled.
mplsAssignedUpstreamLabel	S-PMSI A-D route is sent with this Upstream Label. This is applicable only if Use Upstream Assigned Label is true.
mplsAssignedUpstreamLabelStep	This helps to assign unique upstream assigned label for each flow. This is applicable only if Use Upstream Assigned Label is true.
sendTriggeredSourceActiveAdRoute	If true, allows to send the Source Active A-D Route after receiving Source Tree Join C-Multicast route.
setLeafInformationRequiredBit	This is used to send S-PMSI A-D Route with Leaf Information Required bit Set.

bgp4McastReceiverSite

This object holds the different options for BGP Multicast Sender Site. See bgp4McastReceiverSite for full details. The important options are:

bgp4McastSenderSite Options

Member	Usage
enabled	Enables or disables use of the multicast Sender site.
cMastRouteType	The C-Multicast Route Type. One of: • sourceTreeJoin • sharedTreeJoin
addressFamily	Indicates the IPv4/IPv6 interface id of the router.
startGroupAddress	The first IPv4 or IPv6 Multicast group address in the range of group addresses included in this Register message.
groupMaskWidth	The number of bits in the network mask used with the Group Address.
groupAddressCount	The number of group addresses to be included in the Register message.
sourceGroupMapping	Indicates the source group mapping. One of: • fullyMeshed • oneToOne
startSourceAddress	The first IPv4 or IPv6 source address to be included in this Register message. (IPv4 Multicast addresses are not valid for sources.)
sourceMaskWidth	The number of bits in the mask applied to the Source Address. (The masked bits in the Source Address form the address prefix.) The default value is 32. The valid range is 1 to 128, depending on address family type. Used for (S,G) Type and (S,G, rpt) only.
sourceAddressCount	The number of multicast source addresses to be included. The maximum number of valid possible addresses depends on the values for the Source Address and the Source Mask Width. The default value is 0.
supportLeafAdRoutesSending	If true, helps IXIA to send Leaf A-D Route on receiving a S-PMSI A-D Route with the Leaf Information Required flag set. If false, IXIA shall not send the Leaf A-D Route even if such Update message is received.
sendTriggeredCmulticastRoute	This helps to send Source Tree Join C-Multicast route after receiving Source Active A-D route. This is also required by Shared Tree Join C-Multicast route to send Source Tree Join after receiving Source Active A-D Route.

bgp4UserDefinedAfiSafi

This object holds the number of user-defined AFI/SAFI. See bgp4UserDefinedAfiSafi for more information. The attributes of this object are:

routeImportOptions

Attributes	Description
userDefinedAfiSafiRoute	The Afi/Safi routes are being added.

bgp4userDefinedAfiSafiRoute

This object holds the different options for BGP Multicast Sender Site. See bgp4UserDefinedAfiSafiRoute for full details. The important options are:

bgp4McastSenderSite Options

Member	Usage
enabled	Enables or disables use of the afi/safi route options.
length	The data is padded up to length with left alignment otherwise chopped till length.
data	Data to be transmitted for AFI/SAFI, and regular enable-disable.

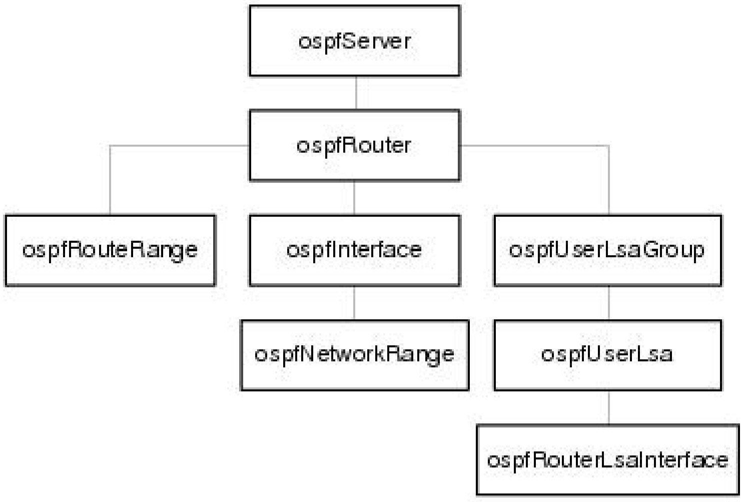
OSPF

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to OSPF. The OSPF related commands are:

- ospfServer — provides access to the OSPF part of a port's protocol server.
- ospfRouter — a container used to hold three lists associate with the router: route ranges, interfaces, and link state advertisements (LSA).
- ospfRouteRange — a set of routes to be included in *ospfRouter.*
- ospfInterface — a network interface to be included in *ospfRouter.*
- ospfUserLSAGroup — a list of LSAs to be included in *ospfRouter*.
- ospfUserLSA — a single LSA description to be included in ospfUserLSAGroup. For router LSA entries, this command holds a list of such entries.
- ospfRouterLSAInterface — a single router LSA interface entry to be included in *ospfUserLSA.*

These commands and the data that they maintain are arranged in an hierarchy as shown in the following figure.

OSPF Command Hierarchy



The MD5 authentication feature is not available on older, non-Linux based load modules.

ospfServer

Refer to ospfServer for a complete description of this command. The *ospfServer* command is necessary in order to access the OSPF component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
ospfServer select 1 5 2
```

will access the OSPF server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routes occurs using the *ospfRouter* command and subsidiary commands. See ospfServer for full details. The important subcommands of this command are:

ospfServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouters	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the *ospfRouter* command.
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. The data appears in the *ospfRouter* command.
delRouter	Deletes a particular router from the list.
setRouter	It is possible to change OSPF router configuration `on the fly'. In order to do this, the following steps are necessary: 1. Modify the router's configuration with *ospfRouter.* (*ospfRouter* has capabilities for modifying elements of underlying objects as well). 2. Use the *ospfServer setRouter* command to set the values from *ospfRouter* into IxHal. 3. Use the *ospfServer* command to write the changes to the hardware.
generateStreams	Once an OSPF simulation has been set up with the other commands in this section, this subcommand automatically generates traffic to all IP addresses in all defined route ranges.
RateControlInterval	The wait time for rate control interval data.
floodLinkStateUpdatesPerInterval	Maximum number of LS update packets that will be sent within the specified rate control interval.

ospfRouter

See ospfRouter for full details. The *ospfRouter* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *ospfRouteRange* command.
- Interfaces — router interface, constructed in the *ospfInterface* command.
- LSA Groups — Link state advertising groups which will be associated with advertised routes, constructed in the *ospfUserLSAGroup* command and subsidiary commands.

ospfRouter Options

Member	Usage
routerId	The ID of the simulated router, expressed as an IPv4 address. (default = 0.0.0.0)

ospfRouter Options

Member	Usage
enable	Enables or disables the use of this emulated OSPF router in the emulated OSPF network. *(default = disabled)*
autoGenerateRouterLsa	If disabled, prevents the emulated OSPF router from automatically creating router LSAs. *(default = enabled)*
enableDiscardLearnedLsas	If enabled, the emulated OSPF router will not learn any LSAs from the neighbor, even when full adjacency has been established. *(default = disabled)*
enableGracefulRestart	Enables the use of graceful restart Helper Mode, per the IETF drafts, on this emulated OSPF router. *(default = disabled)*
enableRebuildAdjacency	Controls the synchronization of OSFP adjacencies during graceful restart. *(default = disabled)*
enableSupportForRFC3623	If enabled, Helper Mode for Graceful Restart per RFC 3623 will be enabled on this emulated OSPF router. *(default = disabled)*
enableStrictLSAChecking	(Available only for use with OSPF Graceful Restart RFC 3623 support.) If enabled, the OSPF Restart Helper will terminate Graceful Restart when there are changes to an LSA that would be flooded to, or retransmitted by, the restarting router. *(default = enabled)*
enableReasonSoftRestart	(Available only for use with OSPF Graceful Restart RFC 3623 support.) If enabled, Graceful Restart Helper Mode will be supported on this emulated OSPF router when the restart reason is an OSPF software restart on the restarting router (for a planned or unplanned outage). *(default = enabled)*
enableReasonSoftReloadUpgrade	(Available only for use with OSPF Graceful Restart RFC 3623 support.) If enabled, Graceful Restart Helper Mode will be supported on this emulated OSPF router when the restart reason is a software reload or upgrade on the restarting router (a planned outage). *(default = enabled)*
enableReasonRedundnatProcessor	(Available only for use with OSPF Graceful Restart RFC 3623 support.) If enabled, Graceful Restart Helper Mode will be supported on this emulated OSPF router when the restart reason is an unplanned switch over to a redundant control processor on the restarting router (an unplanned outage). *(default = enabled)*
enableReasonUnknown	(Available only for use with OSPF Graceful Restart RFC 3623 support.)

ospfRouter Options

Member	Usage
	If enabled, Graceful Restart Helper Mode will be supported on this emulated OSPF router when the restart reason is unknown and unplanned (an unplanned outage). *(default = enabled)*
InterFloodLsUpdateBurstGap	Accesses the gap between each FloodLSUpdate Burst.
MaxFloodLsUpdatesPerBurst	Accesses the particular maximum number of FloodLsUpdate of each Burst data.
LsaRetransmitTime	Defines a value the particular retransmit data.
LsaRefreshTime	Defines a value for the LSA refresh time.
GetInterFloodLsUpdateBurstGap	Accesses the gap between each FloodLSUpdate Burst.
SetInterFloodLsUpdateBurstGap	Defines a value for the gap between each FloodLSUpdate Burst.
GetMaxFloodLsUpdatesPerBurst	Accesses the particular maximum number of FloodLsUpdate of each Burst data.
SetMaxFloodLsUpdatesPerBurst	Defines a value for the maximum number of FloodLSUpdate of each Burst.
GetLsaRetransmitTime	Accesses the particular retransmit data.
SetLsaRetransmitTime	Defines a value for the retransmit time.

ospfRouter Subcommands

Category	Member	Usage
Category	**Member**	**Usage**
Route Ranges	clearAllRouteRanges	Clears all route ranges.
	addRouteRange	Adds a new route range. The route range must have been previously configured through the use of the *ospfRouteRange* command.
	getRouteRange getFirstRouteRange getNextRouteRange	Accesses a particular route range either by ID, or by iterating through all of the route ranges. The data appears in the *ospfRouteRange* command.
	delRouteRange	Deletes a particular route range.
	setRouteRange	It is possible to change route range configuration `on the fly'. In order to do this, the following steps are necessary: 1. Modify the route range's configuration with *ospfRouteRange.* 2. Use the *ospfRouter setRouteRange* command to set the values from *ospfRouteRange* into IxHal. 3. Use the *ospfServer write* command to write the changes to the hardware.
Interfaces	clearAllInterface	Clears all interfaces.
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the *ospfInterface* command.

ospfRouter Subcommands

Category	Member	Usage
	delInterface	Deletes a particular interface.
	getInterface getFirstInterface getNextInterface	Accesses a particular interface either by ID or by iterating through all of the interfaces. The data appears in the *ospfInterface* command.
	setInterface	It is possible to change interface configuration `on the fly'. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with *ospfInterface.* 2. Use the *ospfRouter setInterface* command to set the values from *ospfInterface* into IxHal. 3. Use the *ospfServer write* command to write the changes to the hardware.
User LSA Group	clearAllUserLsaGroup	Deletes a particular user LSA group.
	addUserLsaGroup	Clears all user LSA group.
	delUserLsaGroup	Adds a new user LSA group. The interface must have been previously configured through the use of the *ospfUserLsaGroup* command.
	getUserLsaGroup getFirstUserLsaGroup getNextUserLsaGroup	Accesses a particular user LSA group either by ID or by iterating through all of the LSA groups. The data appears in the *ospfUserLsaGroup* command.
	setUserLsaGroup	It is possible to change user LSA group configuration `on the fly'. In order to do this, the following steps are necessary: 1. Modify the LSA's configuration with *ospfUserLsaGroup.* 2. Use the *ospfRouter setUserLsaGroup* command to set the values from *ospfInterface* into IxHal. 3. Use the *ospfServer write* command to write the changes to the hardware.

ospfRouteRange

The *ospfRouteRange* command describes an individual set of routes. Route ranges are added into *ospfRouter* lists using the *ospfRouter addRouteRange* command. See ospfRouteRange for full details. The important options of this command are:

ospfRouteRange Options

Member	Usage
enable	Enables the use of this route range for the simulated router.
networkIpAddress	The IP address of the routes to be advertised.
prefix	The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0.

ospfRouteRange Options

Member	Usage
numberOfNetworks	The number of prefixes to be advertised.
metric	The cost metric associated with the route.
routeOrigin	Whether the route originated within the area or externally.

ospfInterface

Refer to the ospfInterface for a full description of this command. The *ospfInterface* holds the information related to a single interface on the simulated router. Interfaces are added into the *ospfRouter* interface list using the *ospfRouter addInterface* command. In addition, learned LSAs from the DUT are made available through this command. The important options and subcommands of this command are:

ospfInterface Options

Class	Member	Usage
Basic	enable	Enables the use of the simulated interface.
	connectToDut	Indicates that this interface is directly connected to the DUT.
	protocolInterface Description	The name of the defined *interfaceEntry* from which IP address and mask are extracted for this interface.
	ipAddress ipMask	The IP address associated with the interface, if *protocolInterfaceDescription* is empty.
	adminGroup	Assignment of administrative group numbers to the interface.
OSPF Basic	areaId	The OSPF area ID associated with the interface.
	priority	The priority of the interface, for use in election of the designated or backup master.
	metric	The metric associated with the interface.
	options	Options related to the interface. Multiple options may be or'd together.
	enableValidate Mtu	Enables the validation of maximum transfer units with peers.
	mtuSize	The advertised MTU size.
	networkType	The type of network attached to the link: • Point to point network • Broadcast network
	linkType	The type of link advertised in router LSAs, one of: • Point to point network • Transit network • Stub network
	neighborRouterId	If the *linkType* is a point to point network, this is the address of the other end of the link
Hello Interval	helloInterval	The time between Hello messages sent over the interface.

ospfInterface Options

Class	Member	Usage
	deadInterval	The time after which the DUT router is considered dead if it does not send Hello messages.
Traffic Engineering	enableTraffic Engineering	Enables the generation of the Traffic Engineering opaque LSA with the remainder of the options in this class.
	linkMetric	The traffic engineering metric of the interface.
	maxBandwidth	The maximum bandwidth that can be used on the link.
	maxReservable Bandwidth	The maximum bandwidth that can be reserved on the link.
	unreservedBand widthPriority0-7	The amount of bandwidth not yet reserved at each of the eight priority levels.
Learned LSAs	numberLearned Lsas	The number of obtained LSAs. *ospfInterface getLearnedLsaList* must be called first.
Authentication	authentication Method	The type of authentication to be performed.
	password	If simple password authentication is used, this is the password.
	md5KeyId md5Key	If MD5 authentication is used, this is the key ID and key.
BFD	enableBfdRegistration	Indicates if a BFD session is to be created to the OSPF peer IP address once the OSPF session is established. This allows OSPF to use BFD to maintain IPv4 connectivity the OSPF peer.

ospfInterface Subcommands

Member	Usage
requestLearnedLsa	The first step in obtaining the learned LSA list. It sends a request to the hardware.
getLearnedLsaList	The second step in obtaining the learned LSA list. It allows the Tcl program to wait until the values are available.
getFirstLearnedLsa	The third step in obtaining the learned LSA list. It reads back the first learned LSA, whose values may be read through the *ospfUserLsa* command.
getNextLearnedLsa	The fourth step in obtaining the learned LSA list. It reads back the next learned LSA, whose values may be read through the *ospfUserLsa* command. It should be used multiple times to obtain the whole list.

ospfNetworkRange

Refer to the ospfNetworkRange for a full description of this command. The *ospfNetworkRange* holds the information related to a matrix of simulated routers, organized as a set of rows and columns. Each simulated router is connected to its row and column neighbors and is assigned a unique router ID and IP subnet address. An entry point into the matrix is described as a row and column location. A network range is first described with this command and then associated with an interface with the *ospfRouter addInterface* command; the interface must **not** be connected to the DUT and its *enableAdvertiseNetworkRange*options must be set to *true*.

The important options of this command are:

ospfNetworkRange Options

Member	Usage
numRows	The number of rows of routers in the matrix.
numCols	The number of columns of routers in the matrix.
entryPointRow	The row number for the entry point into the matrix.
entryPointColumn	The column number for the matrix entry point.
firstRouterId	The ID assigned to the first router.
routerIdIncrementBy	The router ID increment.
firstSubnetIpAddress	The IP subnet associated with the first router.
maskWidth	The 32-bit address mask used to indicate the bits of an IP address that are being used for the subnet address.
enableIncrementIpFromMask	If set, the *maskWidth* determines the network part of the address, incremented as new routers are made. Otherwise, use *subnetIpIncrementBy*.
subnetIpIncrementBy	If *enableIncrementIpFromMask* is false, then successive subnets are incremented by this value.
linkType	Router to router links: broadcast or point-to-point.
enableBBit enableEBit	Advertise routers as ABRs or ASBRs; border routers or edge routers.

ospfUserLSAGroup

Refer to ospfUserLsaGroup for a full description. The *ospfUserLSAGroup* describes a list of LSAs to be associated with advertised routes. The list consists of elements from *ospfUserLsa* command.

ospfUserLSAGroup Options

Member	Usage
enable	Enables the use of the LSA group.
areaId	The area ID for the LSA group.

ospfUserLSAGroup Subcommands

Member	Usage
clearAllUserLsas	Clears all LSAs in the list.
addUserLsa	Adds a new user LSA. It must have been previously configured through the use of the *ospfUserLSA* command.
getUserLsa getFirstUserLsa getNextUserLsa	Accesses a particular user LSA by ID or by iterating through all user LSAs. The data appears in the *ospfUserLSA* command.
delUserLsa	Deletes a particular user LSA.

ospfUserLSA

Refer to ospfUserLsa for a full description of this command. The *ospfUserLSA* describes an individual LSA. The types supported are:

- Route LSA — describes all of the router's interfaces along with state and cost. This consists of a list of *ospfRouterLSAInterface* elements added via the *ospfUserLSA addInterfaceDescriptionToRouterLsa* command.
- Network LSA — generated by a designated router and lists all attached routers.
- Summary IP LSA — describes destinations outside of an area.
- Summary ASBR LSA — generated by AS Border Routers and list the ASBR itself.
- Opaque LSAs — holds information used by other protocols. Three types of LSAs are defined to indicate the scope of flooding to be performed: *Local* indicates that the LSA is to be flooded only within the local (sub)network, *Area* indicates that the LSA is to be flooded only within the associated area, and *Domain* indicates that the LSA is to be flooded throughout the AS.

The options available cover all of the possible LSAs that are held by this command. See the command description in ospfUserLsa for the details of which options go with each LSA. The important subcommands of this command relate to Router LSAs only; they are:

ospfUserLSA Subcommands

Member	Usage
clearInterfaceDescription	Clears all router LSA interface descriptions in the list.
addInterfaceDescription ToRouterLsa	Adds a new router LSA interface. The Router LSA must have been previously configured through the use of the *ospfRouterLSAInterface* command.
addInterfaceDescription ToRouterLsaIdentifier	Adds to a router LSA, a particular interface designated by an interface name (for example, "interface 1").
getFirstInterfaceDescription getNextInterfaceDescription	Accesses a particular router LSA by ID or by iterating through the descriptions. The data appears in the *ospfRouterLSAInterface* command.
delInterfaceDescription	Deletes router LSA interface descriptions in the list.
delInterfaceDescription ToRouterLsa	Deletes from a router LSA, a particular interface designated by an interface name (for example, "interface1") that was added using the *addInterfaceDescriptionToRouteLsaIdentifier* command.

ospfRouterLSAInterface

Refer to ospfRouterLsaInterface for a full description of this command.

The *ospfRouterLSAInterface* command describes a single router LSA interface entry.

The data from this entry is added to an *ospfUserLsa* list for a router LSA entry using the *ospfUserLsa addInterfaceDescriptionToRouterLsaIdentifier or addInterfaceDescriptionToRouterLsa* commands. The options available cover the attributes of a router LSA entry. See the command description in Appendix A for the details of which options go with each LSA. No special subcommands are associated with this command.

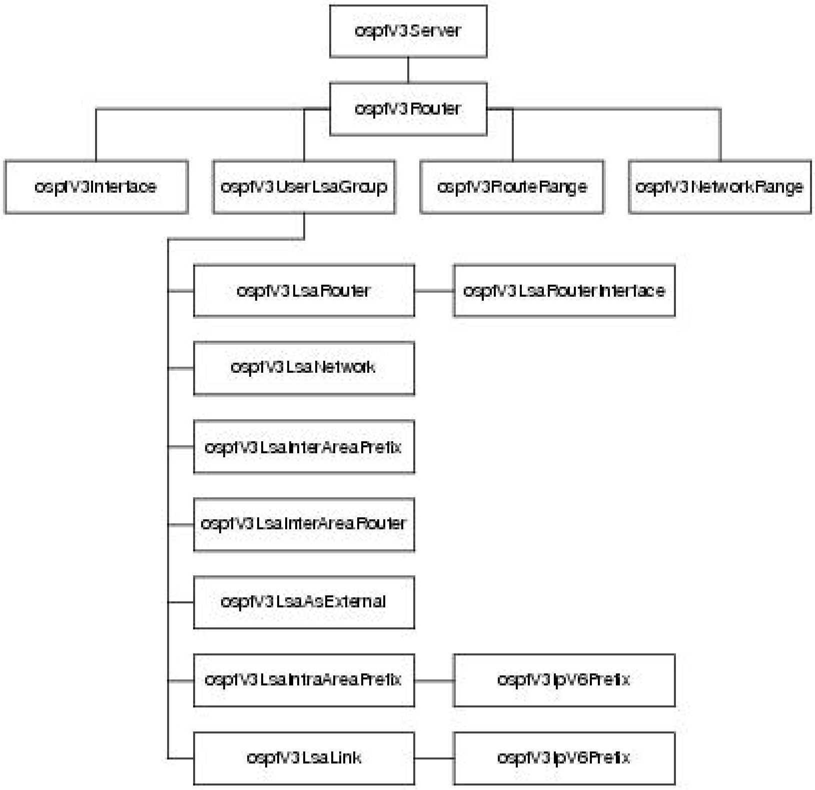
OSPFv3

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to OSPF and OSPFv3. The OSPFv3 related commands are:

- ospfV3Server — provides access to the OSPFV3 part of a port's protocol server.
- ospfV3Router — a container used to hold three lists associated with the router: route ranges, interfaces, and link state advertisements (LSAs).
- ospfV3Interface — a network interface to be included in ospfV3Router.
- ospfV3RouteRange — a set of routes to be included in ospfV3Router.
- ospfV3NetworkRange — a set of network ranges to be included in ospfV3Router.
- ospfV3UserLSAGroup — a list of LSAs to be included in ospfV3Router.
- ospfV3LsaAsExternal — builds an AS external LSA to be included in ospfV3UserLSAGroup.
- ospfV3LsaInterAreaPrefix — builds an inter-area prefix LSA to be included in ospfV3UserLSAGroup
- ospfV3LsaInterAreaRouter — builds an inter-area router LSA to be included in ospfV3UserLSAGroup.
- ospfV3LsaIntraAreaPrefix — builds an intra-area prefix LSA to be included in ospfV3UserLSAGroup.
  - ospfV3IPv6Prefix — sets up address prefixes.
- ospfV3LsaLink — builds a link LSA to be included in ospfV3UserLSAGroup.
  - ospfV3IPv6Prefix— sets up address prefixes.
- ospfV3LsaNetwork — builds a network LSA to be included in ospfV3UserLSAGroup.
- ospfV3LsaRouter — builds a router LSA to be included in ospfV3UserLSAGroup.
  - ospfV3LsaRouterInterface — a single router LSA interface entry to be included in ospfV3LsaRouter.

These commands and the data that they maintain are arranged in an hierarchy as shown in the following figure.

OSPFv3 Command Hierarchy



ospfV3Server

Refer to ospfV3Server for a complete description of this command. The *ospfV3Server* command is necessary in order to access the OSPFv3 component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
ospfV3Server select 1 5 2
```

will access the OSPFv3 server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routes occurs using the *ospfRouter* command and subsidiary commands. The important subcommands of this command are:

ospfV3Server Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouters	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the *ospfRouter* command.
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. The data appears in the *ospfRouter* command.
delRouter	Deletes a particular router from the list.
setRouter	It is possible to change OSPF router configuration `on the fly.' In order to do this, the following steps are necessary: 1. Modify the router's configuration with *ospfRouter.* (*ospfRouter*

ospfV3Server Subcommands

Member	Usage
	has capabilities for modifying elements of underlying objects as well). 2. Use the *ospfServer setRouter* command to set the values from *ospfRouter* into IxHal. 3. Use the *ospfServer write* command to write the changes to the hardware.
generateStreams	Once an OSPFv3 simulation has been set up with the other commands in this section, this subcommand automatically generates traffic to all IP addresses in all defined route ranges.

ospfV3Router

Refer to ospfV3Router for a complete description of this command. The *ospfV3Router* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the ospfV3RouteRange command.
- Network ranges — network ranges to be advertised by the simulated router, constructed in the ospfV3NetworkRange command.
- Interfaces — router interface, constructed in the ospfV3Interface command.
- LSA Groups — link state advertising groups which will be associated with advertised routes, constructed in the ospfV3UserLSAGroup command and subsidiary commands.The important options and subcommands are:

ospfV3Router Options

Member	Usage
enable	Enables or disables the simulated router.
learnedLsaCmd	(*Read-only*.) The ospfV3Router *getFirstLearnedLsa* and *getNextLearnedLsa* commands set the *learnedLsaCmd* (learned LSA command) to one of the learned LSA types if there is any learned LSA; otherwise, *learnedLsaCmd* is set to NULL.
routerId	The ID of the simulated router, expressed as an IP address.
disableAutoGenerate RouterLsa disableAutoGenerate LinkLsa	Used to disable automatic generation of router and link LSAs. These should be turned **on** if you are building OSPF topologies with *ospfV3UserLsa* commands.
enableDiscardLearned Lsas	When this option is set, this simulated OSPF router (RID) will not learn any LSAs from the neighbor.
maxNumLsaPerSecond	Limits the number of LSAs that will be sent per second.

ospfV3Router Subcommands

Category	Member	Usage
Route Ranges	clearAllRouteRanges	Clears all route ranges.
	addRouteRange	Adds a new route range. The route range must have been previously configured through the use of the osp-

ospfV3Router Subcommands

Category	Member	Usage
		fV3RouteRange command.
	getRouteRange getFirstRouteRange getNextRouteRange	Accesses a particular route range either by ID, or by iterating through all of the route ranges. The data appears in the ospfV3RouteRange command.
	delRouteRange	Deletes a particular route range.
	setRouteRange	It is possible to change route range configuration `on the fly'. In order to do this, the following steps are necessary: 1. Modify the route range's configuration with ospfV3RouteRange. 2. Use the ospfV3Router *setRouteRange* command to set the values from ospfV3RouteRange into IxHal. 3. Use the ospfV3Server *write* command to write the changes to the hardware.
Network Ranges	clearAllNetworkRange	Clears all network ranges.
	addNetworkRange	Adds a new network range. The network range must have been previously configured through the use of the ospfV3NetworkRange.
	delNetworkRange	Deletes a particular network range.
	getFirstNetworkRange getNextNetworkRange getNetworkRange	Accesses a particular network range either by ID or by iterating through all of the network ranges. The data appears in the ospfV3NetworkRange command.
	setNetworkRange	It is possible to change network range configuration `on the fly.' In order to do this, the following steps are necessary. 1. Modify the network range's configuration with ospfV3NetworkRange. 2. Use the ospfV3Router *setNetworkRange* command to set the values from ospfV3NetworkRange into IxHal. 3. Use the ospfV3Server *write* command to write the changes to the hardware.
Interfaces	clearAllInterface	Clears all interfaces.
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the ospfV3Interface command.
	delInterface	Deletes a particular interface.
	getInterface getFirstInterface getNextInterface	Accesses a particular interface either by ID or by iterating through all of the interfaces. The data appears in the ospfV3Interface command.
	setInterface	It is possible to change interface configuration `on the fly.' In order to do this, the following steps are neces-

ospfV3Router Subcommands

Category	Member	Usage
		sary: 1. Modify the interface's configuration with ospfV3Interface. 2. Use the *ospfRouter setInterface* command to set the values from ospfV3Interface into IxHal. 3. Use the ospfV3Server *write* command to write the changes to the hardware.
User LSA Group	clearAllUserLsaGroup	Clears all user LSA groups.
	addUserLsaGroup	Adds a new user LSA group. The interface must have been previously configured through the use of the ospfV3UserLSAGroup command.
	delUserLsaGroup	Deletes a particular user LSA group.
	getUserLsaGroup getFirstUserLsaGroup getNextUserLsaGroup	Accesses a particular user LSA group either by ID or by iterating through all of the LSA groups. The data appears in the ospfV3UserLSAGroup command.
	setUserLsaGroup	It is possible to change user LSA group configuration `on the fly.' In order to do this, the following steps are necessary: 1. Modify the LSA's configuration with ospfV3UserLSAGroup. 2. Use the *ospfRouter setUserLsaGroup* command to set the values from ospfV3Interface into IxHal. 3. Use the ospfV3Server *write* command to write the changes to the hardware.
Learned LSA	getLearnedLsaList getFirstLearnedLsa getNextLearnedLsa	1. Gets the list of learned information records populated by protocolLearnedConfig upon the completion of receiving the learned information messages. 2. Gets the first learned information record. 3. Gets the succeeding learned information record.
	requestLearnedLsa	Sends a request to the protocol server for the learned information and registers a callback function for receiving the learned information from the protocol server. Upon receiving the learned information, the callback checks whether the complete learned information is received and sets a flag accordingly.

ospfV3RouteRange

Refer to ospfV3RouteRange for a complete description of this command. The *ospfV3RouteRange* command describes an individual set of routes. Route ranges are added into *ospfV3Router* lists using the *ospfV3Router addRouteRange* command. The important options of this command are:

ospfV3RouteRange Options

Member	Usage
enable	Enables the use of this route range for the simulated router.
networkIpAddress	The IPv6 address of the routes to be advertised.
maskWidth	The number of bits in the prefixes to be advertised.
iterationStep	The increment used between generated addresses.
numberRoutes	The number of routes to be advertised.
metric	The cost metric associated with the route.
routeOrigin	Whether the route originated within the area or externally.

ospfV3NetworkRange

Refer to ospfV3NetworkRange for a complete description of this command. The *ospfV3NetworkRange* command describes an individual set of routes. Network ranges are added into *ospfV3Router* lists using the *ospfV3Router addNetworkRange* command. The important options of this command are:

ospfV3NetworkRange Options

Member	Usage
enable	If true, enables the OSPFv3 network range grid.
entryPointRow	(integer) The row where the entry point router is located in the OSPFv3 network range grid.
entryPointColumn	(integer) The column where the entry point router is located in the OSPFv3 network range grid.
numRows	(integer) The number or rows in a grid.
numColumns	(integer) The number of columns in a grid.
prefix	(integer, range = 1 to 128) The length of the mask used with the IPv6 addresses of the first subnet in the grid. The default is 64.
entryAddressMaskLength	(integer, range = 1 to 128) The length of the mask used with the IPv6 address of the entry point emulated OSPFv3 router in the grid. The default is 64.
linkType	The OSPFv3 link type. One of: • ospfV3NetworkRangeLinkBroadcast • ospfV3NetworkRangeLinkPointToPoint
entryLinkMetric	(integer) The metric of the link connecting the grid with the emulated OSPFv3 router.
BBit	Boolean, to determine whether the router is on the border.
EBit	Boolean, to determine whether the router is on the AS boundary.
routerIdIncrementBy	(Four-byte dotted decimal number, in IPv4 address format.) The identifier for the first emulated OSPFv3 router in the grid.
firstRouterId	(Four-byte dotted decimal number, in IPv4 address format.) The identifier for the first emulated OSPFv3 router in the grid.
firstSubnetIpAddress	(IPv6 address) The IPv6 prefix address of the first subnet in the grid.
entryAddress	(IPv6 address) The IPv6 address of the entry point emulated OSPFv3 router in the grid.

ospfV3Interface

Refer to ospfV3Interface for a full description of this command. The *ospfV3Interface* holds the information related to a single interface on the simulated router. Interfaces are added into the ospfV3Router interface list using the ospfV3Router *addInterface* command. In addition, learned LSAs from the DUT are made available through this command. The important options of this command are:

ospfV3Interface Options

Class	Member	Usage
Basic	enable	Enables the use of the simulated interface.
	protocolInterface Description	The name of the defined *interfaceEntry* from which IP address and mask are extracted for this interface.
OSPF Basic	areaId	The OSPF area ID associated with the interface.
	options	Options related to the interface. Multiple options may be or'd together.
	type	The type of network attached to the link: • Point to point network • Broadcast network
	enableBfdRegistration	Indicates if a BFD session is to be created to the OSPFv3 peer IP address once the OSPFv3 session is established. This allows OSPFv3 to use BFD to maintain IPv4 connectivity the OSPFv3 peer.
	enableIgnoreDBDescMTU	If true, the database set Maximum Transmission Unit (MTU) is ignored.
	instanceId	The instance ID of the interface.
Interval	helloInterval	The time between Hello messages sent over the interface.
	deadInterval	The time after which the DUT router is considered dead if it does not send Hello messages.

ospfV3UserLSAGroup

Refer to ospfV3UserLsaGroup for a full description of this command. The *ospfV3UserLSAGroup* command describes a list of LSAs to be associated with advertised routes. The list consists of elements constructed through the use of the seven commands named:

- ospfV3LsaAsExternal — builds an AS external LSA to be included in ospfV3UserLSAGroup.
- ospfV3LsaInterAreaPrefix — builds an inter-area prefix LSA.
- ospfV3LsaInterAreaRouter — builds an inter-area router LSA.
- ospfV3LsaIntraAreaPrefix — builds an intra-area prefix LSA.
- ospfV3LsaLink — builds a link LSA.
- ospfV3LsaNetwork — builds a network LSA.
- ospfV3LsaRouter — builds a router LSA.

The important options and subcommands of this command are:

ospfV3UserLSAGroup Options

Member	Usage
enable	Enables the use of the LSA group.
areaId	The area ID for the LSA group.
description	A user-specified identifier for this OSPFv3 user LSA group.

ospfV3UserLSAGroup Subcommands

Member	Usage
setDefault	Sets default values for all configuration options.
clearAllUserLsas	Clears all user LSAs in the list.
addUserLsa	Adds a new user LSA.
getUserLsa getFirstUserLsa getNextUserLsa	Accesses a particular user LSA by ID or by iterating through all of the user LSAs.
delUserLsa	Deletes a particular user LSA.
setUserLsa	Allows the configuration values for a user LSA to be overwritten on the fly.

Due to the large number of subsidiary commands that may be used to create the user LSA group, *ospfV3UserLsaGroup* uses a unique means of building and reading a group. An LSA is built using the appropriate command. For example, ospfV3LsaLink is used to build a link LSA. This LSA is then added to the user LSA group by using the command:

```
ospfV3UserLsaGroup addUserLsa lsp3 $::ospfV3LsaLink
```

where *lsp3* is a name associated with the LSP and *$::ospfV3LsaLink* is the means by which *ospfV3UserLsaGroup* knows which LSA to add to the group.

Similarly, a special mechanism is used when reading back the LSAs is a user LSA group. Any of the three get commands (*getUserLsa*, *getFirstUserLsa,* or *getNextUserLsa*) may be used to address the particular LSA. Each of these commands returns the *name* of the command that corresponds to the type of the LSA. For example, the following code will disable the LSA which we added above:

```
 set lsaCmd [ospfV3LsaGroup getUserLsa lsa3]
  $lsaCmd config -enable 0
```

The value of *lsaCmd* is "ospfV3LsaLink" and thus the second command is equivalent to:

```
 ospfV3LsaLink config -enable 0
```

A number of the options in the following seven commands are common. The common options are:

Common OspfV3Lsa Options

Member	Usage
enable	Enables or disables use of the LSA.
advertisingRouterId	The router ID of the router that is originating the LSA.
numLsaToGenerate	The number of LSAs to generate, each with potentially different Link State IDs determined by the value of the *incrementLinkStateIdBy* value.
linkStateId	A unique value to be associated with the LSA. Each of the gen-

Common OspfV3Lsa Options

Member	Usage
incrementLinkStateBy	erated LSAs may have a unique value, as determined by the *incrementLinkStateIdBy* and *numLsaToGenerate* options.
type	The read-only type of the LSA, unique for each LSA type. For example, *$::ospfV3LsaIntraAreaPrefix*.

ospfV3LsaAsExternal

Refer to ospfV3LsaAsExternal for a full description of this command. This command is used to construct an AS external LSA. It uses the common options described in Common OspfV3Lsa Options . The unique options associated with this command are:

ospfV3LsaAsExternal Options

Member	Usage
enableEBit enableFBit enableTBit	Determine the type of external metric and the value of the F-bit and T-bit.
externalRouteTag	If the *enableTBit* is true, an additional value to be used for external routes between AS boundary routers. This field is not used within OSPF.
forwardingAddress	If the *enableFBit*is true, data traffic for the advertised destination will be forwarded to this fully qualified IPv6 address.
prefixAddress prefixLength incrementPrefixBy	The prefix address to be advertised in the LSA and how it is to increment, if *numLsaToGenerate*is more than 1.
prefixOptions	The options associated with the *prefixAddress*.
metric	The cost of the route.
referencedType referencedLinkStateId	If non-zero, indicates the type of a different LSA is referenced and the particular ID of that referenced LSA.

ospfV3LsaInterAreaPrefix

Refer to ospfV3LsaInterAreaPrefix for a full description of this command. This command is used to construct an inter-area prefix LSA. It uses the common options described in Common OspfV3Lsa Options . The unique options associated with this command are:

ospfV3InterAreaPrefix Options

Member	Usage
prefixAddress prefixLength	The prefix address to be advertised in the LSA.
prefixOptions	The options associated with the *prefixAddress*.

ospfV3LsaInterAreaRouter

Refer to ospfV3LsaInterAreaRouter for a full description of this command. This command is used to construct an inter-area router LSA. It uses the common options described in Common OspfV3Lsa Options . The unique options associated with this command are:

ospfV3LsaAsExternal Options

Member	Usage
destinationRouterId incrementDestRouterIdBy	The router ID of the destination router and how it is to be incremented if *numberLsaToGenerate* is greater than 1.
metric	The metric cost of the route.
options	The 24-bit options associated with the destination router.

ospfV3LsaIntraAreaPrefix

Refer to ospfV3LsaIntraAreaPrefix for a full description of this command. This command is used to construct an intra-area prefix LSA. It uses the common options described in Common OspfV3Lsa Options . The unique options associated with this command are:

ospfV3LsaAsExternal Options

Member	Usage
referencedType referencedRouterId referencedLinkStateId	If non-zero, indicates the type of a different LSA is referenced, the router ID and link state ID of that referenced LSA.

ospfV3LsaLink

Refer to ospfV3LsaLink for a full description of this command. This command is used to construct a link LSA. It uses the common options described in Common OspfV3Lsa Options . The unique options associated with this command are:

ospfV3LsaAsExternal Options

Member	Usage
linkLocalAddress	The IPv6 link local address for the interface.
options	The 24-bit options associated with the destination router.
priority	The router's priority for the interface to be used in designated router election.

ospfV3LsaNetwork

Refer to ospfV3LsaNetwork for a full description of this command. This command is used to construct a network LSA. It uses the common options described in Common OspfV3Lsa Options . The unique options associated with this command are:

ospfV3LsaAsExternal Options

Member	Usage
neighborRouterIdList	A space separated list of router IDs for all the routers on the link. Each router is in IPv4 format. For example, {10.1.0.1 192.168.36.2}.
options	The 24-bit options associated with the destination router.

ospfV3LsaRouter

Refer to ospfV3LsaRouter for details. This command constructs a router LSA and contains interfaces constructed with ospfV3LsaRouterInterface. It uses the common options described in Common OspfV3Lsa Options . The unique options and subcommands associated with this command are:

ospfV3LsaRouter Options

Member	Usage
enableBBit enableEBit enableVBit enableWBit	Indicates that the area is an area border router, AS boundary router, an endpoint of one or more fully adjacent virtual links and/or a wild-card multicast receiver.
enableAutoPopulate NeighborInfo	If set, and the value of an interface ID in an ospfV3LsaRouterInterface object added via addInterface is 1, then the *neighborId* and *neighborRouterId* of that object will be ignored and replaced with the DUT's actual values as run time.
options	The 24-bit options associated with the destination router.

ospfV3LsaRouter Subcommands

Member	Usage
clearAllInterfaces	Clears all router interfaces in the list.
addInterface	Adds a new interface configured in the ospfV3LsaRouterInterface command.
getFirstInterface getNextInterface	Accesses a particular interface by iterating through the descriptions. Accessed via ospfV3LsaRouterInterface.
delInterface	Deletes a particular interface.

ospfV3LsaRouterInterface

Refer to ospfV3LsaRouterInterface for a full description of this command. The *ospfV3LsaRouterInterface* command is used to construct an interface descriptor for use in the ospfV3LsaRouter command. The unique options associated with this command are:

ospfV3LsaAsExternal Options

Member	Usage
type	Interface type: point to point, broadcast, or a virtual link.
interfaceId	The router defined interface ID for the interface.
metric	The cost of using this router interface for outbound traffic.
neighborInterfaceId	The interface ID that the neighbor router (or the attached link's designated router when *type* is *broadcast*) has been advertising in Hello packets on the attached link.
neighborRouterId	The router ID of the neighbor router, or the attached link's designated router when *type* is *broadcast*.

ospfV3IPv6Prefix

This command is used to set up and read address prefixes for use in the ospfV3LsaIntraAreaPrefix and ospfV3LsaLink commands. The important options of this command are:

ospfV3RouteRange Options

Member	Usage
address	The prefix address to be advertised in the LSA. Although only *length* bits of the IPv6 address are meaningful, a full IPv6 address should be specified. The *ipV6Address* command can be used to construct the address.

ospfV3RouteRange Options

Member	Usage
incrementBy	If *numLsaToGenerate* in the ospfV3LsaIntraAreaPrefix / ospfV3LsaLink command is greater than 1, this is the value that will be added to the most significant *length* bits of *address* between generated LSAs.
length	The number of high-order bits of *address* that are significant.
options	Specifies options to be included.

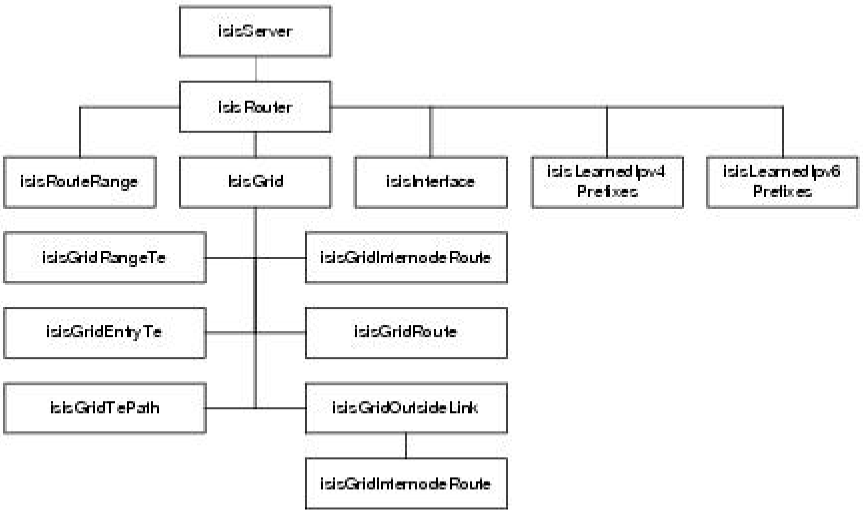
ISIS

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to ISIS. The ISIS related commands are:

- isisServer — provides access to the ISIS part of a port's protocol server.
- isisRouter — a container used to hold two lists associate with the router: route ranges and interfaces.
- isisRouteRange — a set of routes to be included in *isisRouter.*
- isisLearnedIpv4 Prefixes — a grid of simulated ISIS routers behind a router to be included in *isisRouter.*
- isisGridInternodeRoute — the set of networks to and within the grid to be included in *isisGrid.*
- isisGridRoute — the set of advertised networks within the grid to be included in *isisGrid.*
- isisGridOutsideLink — link to another grid to be included in *isisGrid.*
- isisGridRangeTe — Traffic Engineering defaults to be used by all nodes in the grid.
- isisGridEntryTe — Traffic Engineering override values to be used by the entry point to the grid.
- isisGridTePath — path specific Traffic Engineering override values.
- isisInterface — a network interface to be included in *isisRouter.*
- isisLearnedIpv4 Prefixes — a set of commands to fetch learned information of IPv4.
- isisLearnedIpv6 Prefixes — a set of commands to fetch learned information of IPv6.

These commands and the data that they maintain are arranged in a hierarchy as shown in the following figure.

ISIS Command Hierarchy



The following features are not available on older, non-Linux based load modules:

- Hitless Restart
- Clear text authentication
- Point to point support on ISIS interfaces

isisServer

Refer to isisServer for a full description of this command. The *isisServer* command is necessary in order to access the ISIS component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
isisServer select 1 5 2
```

will access the ISIS server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routes occurs using the *isisRouter* command and subsidiary commands. The important subcommands of this command are shown in the following table.

isisServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouters	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the *isisRouter* command.
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. The data appears in the *isisRouter* command.
delRouter	Deletes a particular router from the list.
setRouter	It is possible to change ISIS router configuration `on the fly.' In order to do this, the following steps are necessary: 1. Modify the router's configuration with *isisRouter.* (*isisRouter* has capabilities for modifying elements of underlying objects as well). 2. Use the *isisServer setRouter* command to set the values from *isisRouter* into IxHal. 3. Use the *isisServer write* command to write the changes to the hardware.
generateStreams	Once an OSPF simulation has been set up with the other commands in this section, this subcommand automatically generates traffic to all IP addresses in all defined route ranges.
GetLspMgroupPdusPerInterval	Accesses the particular LSP MGROUP-PDUs per interval.
SetLspMgroupPdusPerInterval	Defines the LSP MGROUP-PDUs for each interval.
GetRateControlInterval	Accesses the particular rate control interval data.
SetRateControlInterval	Defines the wait time for rate control.

isisRouter

Refer to isisRouter for a full description of this command. The *isisRouter* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *isisRouteRange* command.
- Interfaces — router interface, constructed in the *isisInterface* command.
  - Grids — simulated grids of routers behind the router. A virtual interface is generated for each grid.
- learnedInformation IPv4/IPv6 — a set of commands to fetch learned information of IPv4/IPv6.

The important options and subcommands of this command are shown in the following table.

isisRouter Options

Member	Usage
enable	Enables or disables the simulated router.
enableHostName	If true, the given dynamic host name is transmitted in all the packets sent from this router.
hostName	Allows to add a host name to this router.
enableMtIpv6	If checked in L3, emulation type traffic group ID at router level is grayed out and unassigned. NOTE This is unchecked by default for L3 and grayed out for DCE.
routerId	The ID of the simulated router, expressed as an IP address.
maxNumberOfAddresses	The number of area addresses permitted for this IS area.
areaAddressList	The list of area addresses to use.
lspLifetime lspMaxsize lspRefreshRate	Controls the form and lifetime of generated LSPs from this router.
enableDiscardLearned LSPs	Enables or disables retention of learned LSPs.
enableAttached enableOverload enablePartitionRepair enableTrafficEngineering enableWideMetrics	Controls several router simulation options.
enableHitlessRestart hitlessRestartMode hitlessRestartVersion hitlessRestartTime	Controls the operation of hitless restart.

isisRouter Options

Member	Usage
ignoreRecvAuthentication	If *true*, the ISIS router will not authenticate received packets.
areaAuthType areaTxPassword areaRxPasswordList	Sets up authentication for Level 1 LSPs.
domainAuthType domainTxPassword domainRxPasswordList	Sets up authentication for Level 2 LSPs.
GetLspMgroupPdusMinTransmissionInterval	Accesses the particular LSP MGROUP-PDUs minimum transmission intervall data.
SetLspMgroupPdusMinTransmissionInterval	Defines the minimum value for the LSP MGROUP-PDUs transmission interval.
GetPsnpInterval	Gets the Psnp Interval.
SetPsnpInterval	Sets the Psnp Interval.
GetMaxLspMgroupPdusPerBurst	Accesses the particular maximum number of LSP MGROUP-PDUs of each Burst data.
SetMaxLspMgroupPdusPerBurst	Defines a value for the maximum number of LSP MGROUP-PDUs for each Burst.
GetInterLspMgroupPdusBurstGap	Accesses the gap between each LSP MGROUP-PDUs Burst data.
SetInterLspMgroupPdusBurstGap	Defines a value for the gap between each LSP MGROUP-PDUs Burst.

isisRouter Subcommands

Category	Member	Usage
Route Ranges	clearAllRouteRanges	Clears all route ranges.
	addRouteRange	Adds a new route range. The route range must have been previously configured through the use of the *isisRouteRange* command.
	getRouteRange getFirstRouteRange getNextRouteRange	Accesses a particular route range either by ID, or by iterating through all of the route ranges. The data appears in the *isisRouteRange* command.
	delRouteRange	Deletes a particular route range.
	setRouteRange	It is possible to change route range configuration `on the fly.' In order to do this, the following steps are necessary: 1. Modify the interface's configuration with *isisRouteRange.* 2. Use the *isisRouter setRouteRange* command to set the values from *isisRouteRange* into IxHal. 3. Use the *isisServer write* command to write the changes to the hardware.
Interfaces	clearAllInterface	Clears all interfaces.

isisRouter Subcommands

Category	Member	Usage
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the *isisInterface* command.
	delInterface	Deletes a particular interface.
	getInterface getFirstInterface getNextInterface	Accesses a particular interface either by ID or by iterating through all of the interfaces. The data appears in the *isisInterface* command.
	setInterface	It is possible to change interface configuration `on the fly'. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with *isisInterface.* 2. Use the *isisRouter setInterface* command to set the values from *isisInterface* into IxHal. 3. Use the *isisServer write* command to write the changes to the hardware.
Grids	clearAllGrids	Clears all grids.
	addGrid	Adds a new grid. The grid must have been previously configured through the use of the *isisGrid* command.
	delGrid	Deletes a particular grid.
Learned Information	refreshLearnedInformation	This command refreshes the IPv4/IPv6 learned information.
	getLearnedInformation	This command fetches the IPv4/IPv6 learned information.
	getFirstLearnedIpv4PrefixesInfo getFirstLearnedIpv6PrefixesInfo	This command gets the first IPv4/IPv6 prefixes learned information.
	getNextLearnedIpv4PrefixesInfo getNextLearnedIpv6PrefixesInfo	This command gets the next IPv4/IPv6 prefixes learned information.
	getGrid getFirstGrid getNextGrid	Accesses a particular grid either by ID or by iterating through all of the grids. The data appears in the *isisGrid* command.
	setGrid	It is possible to change grid configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the grid's configuration with *isisGrid.* 2. Use the *isisRouter setGrid* command to set the values from *isisGrid* into IxHal. 3. Use the *isisServer write* command to write the changes to the hardware.

isisRouteRange

Refer to isisRouteRange for a full description of this command. The *isisRouteRange* command describes an individual set of routes. Route ranges are added into *isisRouter* lists using the *isisRouter addRouteRange* command. The important options of this command are:

isisRouteRange Options

Member	Usage
enable	Enables the use of this route range for the simulated router.
networkIpAddress	The IP address of the routes to be advertised.
prefix	The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0.
numberOfNetworks	The number of prefixes to be advertised.
metric	The cost metric associated with the route.
routeOrigin	The internal/external route origin indicator.

isisGrid

Refer to isisGrid for a full description of this command. This command allows a grid of ISIS routers to be defined. The grid is added to a particular simulated router via the isisRouter *addGrid*command. The features of the grid configured in this command are:

- Size — the number of columns and rows of nodes.
- Entry point — the location in the grid to which the simulated router is connected.
- Router IDs — the router IDs associated with each element of the grid.
- Link type — whether the links between the simulated router.

The features of a grid which are held in this command are:

- Internode routes — the networks between the simulated router and the grid and the nodes within the grid. These are configured using the isisGridInternodeRoute object and added to this command with the *addInternodeRoute* subcommand.
- Node routes — the routes advertised by each of the nodes within the grid. These are configured using the isisGridRoute object and added to this command with the *addRoute* subcommand.
- Outside links — attachments to the grid to other outside points, which may be placed in other grids. These are configured using the isisGridOutsideLink command and added to this command with the *addOutsideLink* subcommand.
- Traffic Engineering data — a default set of TE data may be associated with all nodes using the isisGridRangeTe command. This data may be overridden for the node connecting the grid to the simulated router using the isisGridEntryTe command. Individual paths through the grid may further override these values using the isisGridTePath command and added to this command with the *addTePath* subcommand.

The important options and subcommands of this command are:

isisGrid Options

Category	Member	Usage
General	enable	Enables the use of the grid.

isisGrid Options

Category	Member	Usage
	numRows numColumns	Specifies the number and organization of the grid.
	entryPointRow entryPointColumn	Indicates the connection point from the simulated router into the simulated grid. Coordinates are 0-based.
	firstRouterId routerIncrementBy	Specifies how router IDs are associated with routers in the grid.
	linkType	Indicates whether the link from the router to the grid and within the grid are broadcast or point to point.
Traffic Engineering	enableTe	Enables generation of traffic engineering (TE) data.
	teRouterId teRouterIncrementBy	Specifies how TE router IDs are associated with routers in the grid.
	overrideEntryTe	Indicates that the TE values for the entry point grid router are to be overridden.
enableHost Name		If true, the given dynamic host name is transmitted in all the packets sent from this router.
hostNamePrefix		Allows to add a host name to this network range. The name prefix is appended by row ID and column ID in ".<rowid.<colid" combination.

isisGrid Subcommands

Category	Member	Usage
Internode Routes	addInternodeRoute delInternodeRoute clearAllInternodeRoutes	Maintains a list of internode routes built with isisGridInternodeRoute.
	getFirstInternodeRoute getNextInternodeRoute	Iterates through the internode routes in the list.
Node Routes	addRoute delRoute clearAllRoutes	Maintains a list of node routes built with isisGridRoute.
	getFirstRoute getNextRoute	Iterates through the node routes in the list.
Outside Links	addOutsideLink delOutsideLink clearAllOutsideLinks	Maintains a list of outside links built with isisGridOutsideLink.
	getFirstOutsideLink getNextOutsideLink	Iterates through the outside links in the list.
TE Paths	addTePath delTePath clearAllTePaths	Maintains a list of TE paths built with isisGridTePath.
	getFirstTePath getNextTePath	Iterates through the TE paths in the list.

isisGridInternodeRoute

Refer to isisGridInternodeRoute for a full description of this command. This command is used in conjunction with isisLearnedIpv4 Prefixes and isisGridOutsideLink.

In conjunction with isisLearnedIpv4 Prefixes:

Internode routes are the networks between the simulated router and the grid and the nodes within the grid. These are configured using this object and added to the grid with the isisLearnedIpv4 Prefixes *addInternodeRoute* subcommand.

The network specified in the options of this command is applied iteratively to:

- The connection between the simulated router and the entry point node in the grid. A new interface is created on the simulated router.
- The connections between all the nodes in each row. That is, the connection between the first and second nodes in the first row, then the second and third nodes in the first row...then between the first and second nodes in the second row, and so forth.
- The connection between each node in a row and the node below it, in a row first manner. That is, the connection between the first node in the first and second rows, then the second node in the first and second rows...then between the first node in the second and third rows and so forth.

The network part of the *ipAddress* option is incremented by 1 between uses. The interface to the left or above in the connection receives the ".1" address on the network and the other receives the ".2" address. The simulated router is considered to be "above" all others.

In conjunction with isisGridOutsideLink:

This command is used to describe a network associated with an outside link to the grid.

The important options of this command are:

isisGridInternodeRoute Options

Member	Usage
ipType	Indicates whether the network is for IPv4 or IPv6.
ipAddress ipMask ipStep	The network being described.

isisGridRoute

Refer to isisGridRoute for a full description of this command. These are the routes advertised by each of the nodes within the grid. These are added to the grid using the isisLearnedIpv4 Prefixes *addRoute* command.

The route range specified in the options of this command is applied iteratively to:

- The interface connecting each node in a row with its right neighbor. That is, the interface from the first node to the second node in the first row, then the second and third nodes in the first row...then between the first and second nodes in the second row and so forth.
- The interface connecting each node in a row with its neighbor below it, in a row first manner. That is, the interface from the first node in the first and second rows, then

the second node in the first and second rows...then between the first node in the second and third rows and so forth.

The number of networks indicated by *numberOfNetworks* is advertised for a node and then the network part of the *networkIpAddress* option is incremented by *nodeStep* between uses. For example, if the following settings were used:

isisGridRoute Example

Option	Value
ipType	addressTypeIpV4
networkIpAddress	60.0.0.0
nodeStep	256
numberOfNetworks	50
prefix	24

Then the interface from the node at row 0, column 0 to the node at row 0, column 1 would advertise the networks 60.0.0.0/24 through 60.0.49.0/24. The interface from row 0, column 1 to row 0, column 2 would advertise the networks 60.1.0.0/24 through 60.1.49.0/24.

The important options of this command are:

isisGridRoute Options

Member	Usage
enable	Enables the use of the node route.
ipType	Indicates whether the network is for IPv4 or IPv6.
networkIpAddress prefix	The first network being described.
numberOfNetworks	The number of networks per node.
nodeStep	The increment to be applied to the network part of networkIpAddress between per node uses.
routeOrigin	Whether the route origin should be indicated as internal or external.
metric	The cost metric associated with the route.

isisGridOutsideLink

Refer to isisGridOutsideLink for a full description of this command. This command describes an attachment from the grid to other outside points, which may be places in other grids. These are added to the grid with the isisLearnedIpv4 Prefixes *addOutsideLink* command.

Multiple IPv4 and IPv6 networks may be associated with the outside link. These are set up with isisGridInternodeRoute and added to this command with the *addRoute* subcommand.

isisGridOutsideLink Options

Member	Usage
connectionRow connectionColumn	The location in the grid from which an outside link is to be defined.
maximumBandwidth maximumReservableBandwidth unreservedBandwidthPriority0-	The bandwidth to be advertised on the link.

isisGridOutsideLink Options

Member	Usage
7	
administrativeGroup	The administrative group membership to be advertised on the link.
linkedRouterId	The router ID of the outside router.

isisGrid Subcommands

Member	Usage
addRoute delRoute clearAllRoutes	Maintains a list of routes built with isisGridInternodeRoute.
getFirstRoute getNextRoute	Iterates through the routes in the list.

isisGridRangeTe

Refer to isisGridRangeTe for a full description of this command. This command sets the default TE data values for all nodes in the ISIS grid. The *enableTe* option in the isisLearnedIpv4 Prefixes command must be set to *true* for this data to be used. This data may be overridden for the node connecting the grid to the simulated router using the isisGridEntryTe command. Individual paths through the grid may further override these values using the isisGridTePath command and added to this command with the *addTePath* subcommand.

The important options of this command are:

isisGridRangeTe Options

Member	Usage
linkMetric	The metric associated with the interface that the TE data is advertised on.
maximumBandwidth maximumReservableBandwidth unreservedBandwidthPriority0-7	The bandwidth to be advertised on the link.
administrativeGroup	The administrative group membership to be advertised on the link.

isisGridEntryTe

Refer to isisGridEntryTe for a full description of this command. This command overrides the default TE data values for all nodes in the ISIS grid set with the isisGridRangeTe command. The *enableTe* and *overrideEntryTe* options in the isisLearnedIpv4 Prefixes command must be set to *true* for this data to be used.

The important options of this command are:

isisGridEntryTe Options

Member	Usage
linkMetric	The metric associated with the interface that the TE data is advertised on.
maximumBandwidth	The bandwidth to be advertised on the link.

Member	Usage
maximumReservableBandwidth unreservedBandwidthPriority0-7	
administrativeGroup	The administrative group membership to be advertised on the link.

isisGridTePath

Refer to isisGridTePath for a full description of this command. This command overrides the default TE data values for all nodes in the ISIS grid set with the isisGridRangeTe command. The *enableTe* option in the isisLearnedIpv4 Prefixes command must be set to *true* for this data to be used.

The path starts and ends with particular nodes in the grid. The row and column of the end point must be greater than or equal to those of the starting point. The path through the grid is described in row and column step sizes. For example, if the grid is 8 x 8, then a path starting at (row = 0, column = 0) and ending at (3, 6) with a row step of 1 and a column step of 2 will go through the following grid nodes:

(0,0), (1, 2), (2, 4) and (3, 6). Any excess row or column step values which would take the path past the endpoint are truncated.

The important options of this command are:

isisGridTePath Options

Member	Usage
startRow startColumn	The starting location in the grid for the path.

endRow
endColumn
	The ending location in the grid for the path.

rowStep columnStep	How many rows and columns to move down and right, respectively, per TE path step.
enableBidirectional	Indicates whether both directions of the path should be advertised.
metric	The metric associated with the interface that the TE data is advertised on.
maximumBandwidth maximumReservableBandwidth unreservedBandwidthPriority0-7	The bandwidth to be advertised on the link.
administrativeGroup	The administrative group membership to be advertised on the link.

isisInterface

Refer to isisInterface for a full description of this command. The *isisInterface* holds the information related to a single interface on the simulated router. Interfaces are added into the *isisRouter* interface list using the *isisRouter addInterface* command. This information is pulled from existing interface during *isisRouter addInterface* command, and is read-only. The important options of this command are:

isisInterface Options

Member	Usage
enable	Enables the use of this interface for the simulated router.
connectToDut	If set, this IS-IS interface is directly connected to the DUT.
enableAutoConfigure Area	If set, the area for the interface is determined during the Hello interchange with the DUT.
enableAutoAdjustMTU	If set, and a padded Hello message is received on the interface, then the interfaces MTU will be adjusted to match the packet length of the received Hello message.
enableAutoAdjust Protocols	If set, and a Hello message is received which contains a protocols TLV, then the interfaces protocols will be adjusted to match the received TLV.
networkType	Indicates the type of network attached to the interface: broadcast or point-to-point.
protocolInterface Description	The name of the defined *interfaceEntry* from which IP address and mask are extracted for this interface.
metric	The cost metric associated with the route.
level	The IS-IS level associated with the interface: Level 1, 2, or both.
interfaceId	The OSI interface ID for this interface. This is a read only value.
IPv6 MT Metric	This metric is same as the Interface Metric. If enabled, it allows you to enter data.
interfaceMetric enableUserInterface Metric	Sets the metric for the interface connected to the grid.
priorityLevel1 priorityLevel2	The priority level associated with the Level 1 or Level 2 aspect of the interface. This is used in master election.
helloIntervalLevel1 helloIntervalLevel2	The Hello interval used with the Level 1 or Level 2 aspect of the interface. Used to send regular messages to neighbor IS-IS routers.
deadIntervalLevel1 dealIntervalLevel2	The dead interval used with the Level 1 or Level 2 aspect of the interface. Used to determine if neighbor routers are non-operational.
circuitAuthType circuitTxPassword circuitRxPasswordList	Sets up authentication for IIHs.
enableBfdRegistration	Indicates if a BFD session is to be created to the ISIS peer IP address once the ISIS session is established. This allows ISIS to use BFD to maintain IPv4 connectivity the ISIS peer.
enable3Way Handshaking	If true, Ixia emulated point-to-point circuit will include 3-way TLV in its P2P IIH and attempt to establish the adjacency as specified in RFC 5303.
extendedLocalCircuitId	The integer value of the local circuit ID. The default is 1.
enableHelloPadding	If true, hellopadding is enabled.

isisLearnedIpv4 Prefixes

Refer to isisLearnedIpv4Prefixes for a full description of this command. This command is used to fetch learned information of L3 ISIS IPv4 prefixes.

The important options of this command are:

isisLearnedIpv4Prefixes Options

Member	Usage
lspId	The LSP number of the IPv4 prefix.
sequenceNumber	Sequence number of the LSP containing the route.
age	The age in time since last refreshed.
hostName	The host name as retrieved from the related packets.
metric	The route metric.
ipv4Prefix	The mask width of the IPv4 Prefix.

isisLearnedIpv6 Prefixes

Refer to isisLearnedIpv6Prefixes for a full description of this command. This command is used to fetch learned information of L3 ISIS IPv6 prefixes.

The important options of this command are:

isisLearnedIpv6Prefixes Options

Member	Usage
lspId	The LSP number of the IPv6 prefix.
sequenceNumber	Sequence number of the LSP containing the route.
age	The age in time since last refreshed.
hostName	The host name as retrieved from the related packets.
metric	The route metric.
ipv4Prefix	The mask width of the IPv6 Prefix.

RSVP-TE

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to RSVP. The RSVP-related commands are:

- rsvpServer — provides access to the RSVP part of a port's protocol server.
- rsvpNeighborPair — describes a pair of routers in which one is the DUT and the other is simulated by the Ixia port.
- rsvpDestinationRange — describes a set of routers that are the destination of MPLS tunnels. Destination ranges correspond to ingress or egress routers.

  NOTE Destination Range as used in this document is synonymous with Tail Range of IxNetwork application.

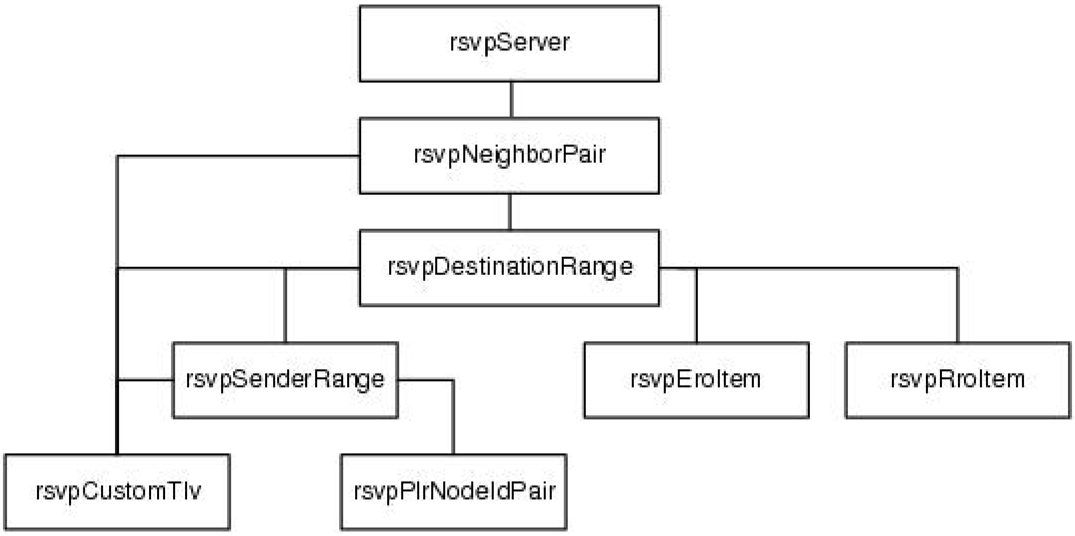
- rsvpTunnelTail TrafficEndPoint — configures the IP addresses to be used in the Destination IP field in traffic to be sent over the LSPs terminating on this Tail Range.
- rsvpTunnelLeafRange — describes a range of tunnel leaf endpoints when the emulation type in Tunnel Tail Ranges is set to RSVP-TE P2MP.
- rsvpSenderRange — describes a set of routers that are the source of MPLS tunnels in an ingress simulation.

  NOTE Sender Range as used in this document is synonymous with Head Range of IxNetwork application.

- rsvpTunnelHead TrafficEndPoint — configures the IP addresses to be used in the Source IP field in traffic to be sent over the LSPs originating from this Head Range.
- rsvpEroItem — describes an explicit route item entry for use in ingress router simulations.
- rsvpRroItem — describes a return route item entry for use in egress router simulations.
- rsvpPlrNodeIdPair — a point of local repair (PLR) and node to avoid pair used in fast reroute.
- rsvpCustomTLV — a custom type-length-value (TLV) item used by several other commands to represent optional message objects.

These commands and the data that they maintain are arranged in a hierarchy, as shown in the following figure.

RSVP Command Hierarchy



The refresh reduction feature is not available on older, non-Linux based load modules.

rsvpServer

Refer to rsvpServer for a full description of this command. The *rsvpServer* command is necessary in order to access the RSVP component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
rsvpServer select 1 5 2
```

will access the RSVP server for chassis 1, card 5, port 2.

This command holds a list of adjacent routers, called neighbor pairs. The definition of these routers occurs using the *rsvpNeighborPair* command and subsidiary commands. The important options and subcommands of this command are shown in the following table.

rsvpServer Options

Member	Usage
enableBgpOverLsp	Setting this option to true allows non-RSVP control packets (such as BGP control packets destined to the far-end PE) to be encapsulated with the MPLS label learned by RSVP. If it is set to false, no control packets are encapsulated with the MPLS label.

rsvpServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllNeighborPair	Removes all neighbor pairs from the list of neighbor pairs.
addNeighborPair	Adds a neighbor pair to the list of neighbor pairs. The neighbor pair must have been previously configured through the use of the *rsvpNeighborPair* command.
getNeighborPair getFirstNeighborPair getNextNeighborPair	Accesses a particular neighbor pair from the list, either directly by ID or by iterating through all of the neighbor pairs. The data appears in the *rsvpNeigh-*

rsvpServer Subcommands

Member	Usage
	borPair command.
delNeighborPair	Deletes a particular neighbor pair from the list.
setNeighborPair	It is possible to change RSVP neighbor pair configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the neighbor pair's configuration with *rsvpNeighborPair.* (*rsvpNeighborPair* has capabilities for modifying elements of underlying objects as well.) 2. Use the *rsvpServer setNeighborPair* command to set the values from *rsvpNeighborPair* into IxHal. 3. Use the *rsvpServer write* command to write the changes to the hardware.
generateStreams	Once an RSVP simulation has been set up with the other commands in this section, has been started, and has run long enough to receive labels from the DUT, this subcommand automatically generates traffic. Traffic is generated for each label received from the DUT. Traffic is sent from the first address in the sender range to the first address in the destination range for the neighbor pair associated with the label. NOTE This command is specific to generate traffic for RSVP-TE P2P lsps.
restartNeighbor	Restarts the RSVP neighbor pair with an identifier of *NeighborPairId*. The results may be accessed using the *rsvpNeighborPair* command.
GetMaxLspInitiationsPerSec	Accesses the maximum number of LSP initiations per second data.
SetMaxLspInitiationsPerSec	Defines a value for the maximum number of LSP initiations per second.
GetEnableControlLspInitiationRate	Accesses the Control LSP Initiation Rate checkbox.
SetEnableControlLspInitiationRate	Enables the Control LSP Initiation Rate checkbox.

rsvpNeighborPair

Refer to NAME - rsvpNeighborPair for a full description of this command. The *rsvpNeighborPair* command represents a pair of routers — one is the DUT and the other is simulated by the Ixia protocol server. In addition to some identifying options, it holds two lists for the router:

- Destination Ranges — a list of routers which represent the termination point of MPLS tunnels being constructed. Destination ranges are constructed in the *rsvpRouteRange* command.
- Hello TLVs — generalized TLV messages that are included with all Hello messages and built with the rsvpCustomTLV command.

The important options and subcommands of this command are shown in the following table.

rsvpNeighborPair Options

Category	Member	Usage
Neighbor Pair	enableNeighborPair	Enables or disables the simulated neighbor pair.
	ipAddress	The IP address of the simulated router.
	dutAddress	The IP address of the device under test. This is the RSVP router that the simulated router is directly connected to.
	enableBFDRegistration	If true, enables BFD registration with RSVP-TE.
	enableBundleMessage Sending	If true, enables the sending of RSVP Bundle Message.
Hello Messages	enableHello	Enables the transmission of Hello messages between the simulated router and the DUT.
	helloInterval	The interval, in seconds, between Hello messages.
	helloTimeoutMultplier	The number of Hellos sent without confirmation before the DUT is considered dead.
Label Space	labelSpaceStart	The first label to be used for RSVP tunnels.
	labelSpaceEnd	The last label to be used for RSVP tunnels.
Returned Labels	lsp_tunnel	This is a string identifier which contains the information to map the returned label to a particular P2P lsp or P2Mp lsp.
	rxLabel	This is the MPLS label associated with the tunnel ID in *lsp_tunnel.*
	numRxLabels	The number of received labels.
	assignedLabel	Label value assigned to the LSP/Tunnel (by the Ixia-emulated router) in response to a Label Request from the DUT.
	reservationState	The reservation state, once there is a graceful restart. The values are: 1. None=0 (Default) 2. Stale=1 (Recovery State but Recovery Label not yet received) 3. Recovered=2 (Recovery Label received) 4. Restarting=3 (RSVP emulated Router is restarting).
	type	This signifies the type of the lsp for which the current label was returned. The values are: 1. RSVP-TE 2. RSVP-TE P2MP
	leafIp	It contains the value of the leafIp which identifies one particular P2MP RSVP-TE sub-lsp for which the label was returned.

rsvpNeighborPair Options

Category	Member	Usage
		NOTE: This does not have any significance for P2P lsps.
	numAssignedLabels	The total number of assigned labels.
Refresh Reduction	enableRefreshReduction	Enables or disables the feature.
	summaryRefreshInterval	The interval between summary refresh messages.
Graceful Restart	enableGracefulRestartHelperMode	If true, enables the graceful restart helper mode.
	enableGracefulRestartingMode	If true, enables the graceful restart - restarting mode.
	actualRestartTime	The actual restart time is the interval after which a hello packet is sent with a new Src Instance Id. The default value is 15000 ms
	restartTimeInterval	This value along with the Recovery Time is advertised in the Hello-packets as part of a Restart-capability object. The default value is 30000 ms.
	recoveryTimeInterval	Ixia waits for a configured interval for the DUT to help it recover the egress LSPs. If no recovery label is received from the DUT within this time , those Egress LSPs are treated as having time-outed and the labels are removed. The default value is 30000 ms.
	gracefulRestartStartTime	The time interval after this restart timer is fired, and the neighboring nodes are restarted. During this interval the hello message are not being sent. The default value is 30000 ms.
	gracefulRestartUpTime	After the Restarting time is over, Ixia waits for this configured interval before trying to repeat the Restart cycle. This is effective only when the number of restarts is not equal to the user-configured number of Graceful Restart cycles . After that Ixia will not take any action to being down a Neighborship on its own. The default value is 30000 ms.
	numberOfGracefulRestarts	The number of times the Ixia emulated RSVP neighbor will move to Restarting / Recovering and Up states before stopping the cycle. The default value is 0.

rsvpNeighborPair Subcommands

Category	Member	Usage
Destination	clearAllDestinationRange	Clears all destination ranges associated with the

rsvpNeighborPair Subcommands

Category	Member	Usage
Ranges		neighbor pair.
	addDestinationRange	Adds a new destination range. The destination range must have been previously configured through the use of the *rsvpDestinationRange* command.
	getDestinationRange getFirstDestinationRange getNextDestinationRange	Accesses a particular destination range either by ID, or by iterating through all of the destination ranges. The data appears in the *rsvpDestinationRange* command.
	delDestinationRange	Deletes a particular destination range.
Hello TLVs	clearHelloTlvList	Clears all Hello TLVs associated with the neighbor pair.
	addHelloTlv	Adds a new Hello TLV. The Hello TLV must have been previously configured through the use of the *rsvpHelloTlv* command.
	getFirstHelloTlv getNextHelloTlv	Accesses a particular Hello TLV by iterating through all of the Hello TLVs. The data appears in the rsvpCustomTLV command.
	delHelloTlv	Deletes a particular Hello TLV.
Received Labels	requestRxLabels	This is the first step in retrieving labels from the DUT. This subcommand requests that all of the received MPLS tunnel-label pairs be retrieved.
	getLabels	This is the second step in retrieving labels from the DUT. This subcommand allows the Tcl program to wait until the labels have been retrieved.
	getFirstLabel getNextLabel	This is the last step in retrieving labels from the DUT. These subcommands get the first and subsequent received tunnel-label pair from the list retrieved with *requestRxLabels*. The values are available in the *numRxLabels*, *lsp_tunnel* and *rxLabel* options.
Assigned Labels	requestAssignedLabels	Requests that the assigned RSVP labels associated with this neighbor pair be retrieved from the protocol server. This command must be followed by call to rsvpNeighborPair getLabels.
	getLabels	This is the second step in retrieving assigned labels from the DUT. This subcommand allows the Tcl program to wait until the labels have been retrieved.
	getFirstLabel getNextLabel	This is the last step in retrieving labels from the DUT. These subcommands get the first and subsequent received tunnel-label pair from the list retrieved with *requestAssignedLabels*. The retrieved label and lsp/sub-lsp identifiers are available by looking at *numAssignedLabels, type, lsp_tunnel, leafIp, and assignedLabels* values.

rsvpDestinationRange

Refer to rsvpDestinationRange for a full description of this command. The *rsvpDestinationRange* command describes a set of MPLS routers which are the tunnel endpoints for a number of MPLS tunnels established with RSVP. A number of other lists are associated with this command:

- rsvpSenderRange — a set of MPLS routers which are the tunnel start-points.
- rsvpEroItem — a set of addresses associated with the explicit route option (ERO) of RSVP messages. This option indicates the path through a set of MPLS routers that the tunnel is to take. This is used when the destination range is associated with an ingress router.
- rsvpRroItem — a set of addresses associated with the returned route option (RRO). This option indicates the set of MPLS routers that were used in a tunnel's creation. This is used when the destination range is associated with an egress router.
- RESV TLV — a set of custom TLVs to be included in RESV messages. These may only be used for egress routers.
- RESV TEAR TLV — a set of custom TLVs to be included in RESV TEAR messages. These may only be used for egress routers.
- RESV ERR TLV — a set of custom TLVs to be included in RESV ERR messages. These may only be used for ingress routers.
- PATH TLV — a set of custom TLVs to be included in PATH messages. These may only be used for egress routers.

NOTE Destination Range as used in this document is synonymous with Tail Range of IxNetwork application.

The important options and subcommands of this command are shown in the following table.

rsvpDestinationRange Options

Category	Member	Usage
DestinationRange	enableDestinationRange	Enables or disables the use of the destination range.
	emulationType	Selects the type of emulation for the RSVP Destination Ranges.
	behavior	Indicates whether the destination range corresponds to an Ingress or egress router.
	fromIpAddress	The IP address of the first destination router.
	rangeCount	The number of destination routers. Each router's address is one greater than the previous one.
	tunnelIdStart	Sets the start of the range of tunnel IDs to be used in simulations.
	tunnelIdEnd	The end of the range of tunnel IDs.
ERO	enableEro	Enables the use of the ERO option in an ingress mode.
	eroMode	Indicates whether the DUT's address is to be prepended to the ERO list and whether it is a LOOSE or STRICT entry.

rsvpDestinationRange Options

Category	Member	Usage
	prefixLength	If the DUT's address is to be prepended to the ERO list, this indicates what prefix length is to be used for the entry.
Ingress Options	enableSendRro	When the destination range is used in ingress mode, this indicates that a SEND RRO option is to be included in RSVP messages sent downstream.
Egress Options	refreshInterval	When the destination range is used in egress mode, this indicates the time, in seconds, between the simulated router's message to the DUT.
	bandwidth	The requested bandwidth for the tunnel, expressed in kbits per second.
	timeoutMultiplier	The number of Hellos before a router is declared dead.
	enableResvConf	Indicates whether RESV confirmation messages are to generated in response to RESV messages with the RESV confirmation class object.
P2MP	p2mpId	The P2MP identifier represented in IP address format.
	isHeadIp Prepended	If true, prepend the tunnel head IP as a RRO/SRRO subobject at the beginning of the RRO / SRRO list in the packet. Note that all flags will be set to 0 if this automatic inclusion option is used.
	isLeafIp Prepended	If true, prepend the tunnel leaf IP as a RRO/SRRO subobject at the beginning of the RRO / SRRO list in the packet. Note that no label will be automatically inserted and all flags will be set to 0 if this automatic inclusion option is used.
	isConnectedIp Appended	If true, append the connected IP as a RRO/SRRO subobject at the end of the RRO / SRRO list in the packet. Note that all flags will be set to 0 if this automatic inclusion option is used.
	isSendingAsSrro	If true, send this as a RRO.
	isSendingAsRro	If true, send this as a SRRO. Note that both Send as RRO and Send as SRRO can be selected at the same time if so required by the user.

rsvpDestinationRange Subcommands

Category	Member	Usage
Sender Ranges	clearAllSender	Clears all sender ranges.
	addSenderRange	Adds a new sender range. The sender range must have been previously configured through the use of the *rsvpSenderRange* command.
	getSenderRange getFirstSenderRange getNextSenderRange	Accesses a particular sender range either by ID, or by iterating through all of the sender ranges. The data appears in the *rsvpSenderRange* command.

rsvpDestinationRange Subcommands

Category	Member	Usage
	delSenderRange	Deletes a particular sender range.
	setSenderRange	It is possible to change sender range configuration on the fly. In order to do this, the following steps are necessary:
		1. Modify the sender range's configuration with *rsvpSenderRange.* 2. Use the *rsvpDestinationRange setSender-Range* command to set the values from *rsvpSenderRange* into IxHal 3. Use the *rsvpServer write* command to write the changes to the hardware.
ERO Items	clearAllEro	Clears all ERO list items.
	addEroItem	Adds a new ERO item. The ERO item must have been previously configured through the use of the *rsvpEroItem* command.
	getFirstEroItem getNextEroItem	Accesses a particular ERO item by iterating through all of the ERO items. The data appears in the *rsvpEroItem* command.
RRO Items	clearAllRro	Clears all RRO list items.
	addRroItem	Adds a new RRO item. The RRO item must have been previously configured through the use of the *rsvpRroItem* command.
	getFirstRroItem getNextRroItem	Accesses a particular RRO item by iterating through all of the RRO items. The data appears in the *rsvpRroItem* command.
PATH TLV Items	clearAllPathTlv	Clears all PATH TLV list items.
	addPathTlvItem	Adds a new PATH TLV item. The PATH TLV item must have been previously configured through the use of the rsvpCustomTLV command.
	getFirstPathTlvItem getNextPathTlvItem	Accesses a particular PATH TLV item by iterating through the PATH TLV items. The data appears in the rsvpCustomTLV command.
RESV TLV Items	clearAllResvTlv	Clears all RESV TLV list items.
	addResvTlvItem	Adds a new RESV TLV item. The RESV TLV item must have been previously configured through the use of the rsvpCustomTLV command.
	getFirstResvTlvItem getNextResvTlvItem	Accesses a particular RESV TLV item by iterating through the RESV TLV items. The data appears in the rsvpCustomTLV command.
RESV ERR TLV Items	clearAllResvErrTlv	Clears all RESV ERR TLV list items.

rsvpDestinationRange Subcommands

Category	Member	Usage
	addResvErrTlvItem	Adds a new RESV ERR TLV item. The RESV ERR TLV item must have been previously configured through the use of the rsvpCustomTLV command.
	getFirstResvErrTlvItem getNextResvErrTlvItem	Accesses a particular RESV ERR TLV item by iterating through the RESV ERR TLV items. The data appears in the rsvpCustomTLV command.
RESV TEAR TLV Items	clearAllResvTearTlv	Clears all RESV TEAR TLV list items.
	addResvTearTlvItem	Adds a new RESV TEAR TLV item. The RESV TEAR TLV item must have been previously configured through the use of the rsvpCustomTLV command.
	getFirstResvTearTlvItem getNextResvTearTlvItem	Accesses a particular RESV TEAR TLV item by iterating through the RESV TEAR TLV items. The data appears in the rsvpCustomTLV command.
P2MP	addTunnelLeafRange delTunnelLeafRange getTunnelLeafRange setTunnelLeafRange getFirstTunnelLeafRange getNextTunnelRange clearAllTunnelLeafRange	Controls adding, deleting, and obtaining information about the tunnel leaf ranges.
	addTailTrafficEndPoint delTailTrafficEndPoint getTailTrafficEndPoint setTailTrafficEndPoint getFirstTailTrafficEndPoint getNextTailTrafficEndPoint clearAllTailTrafficEndPoint	Controls adding, deleting, and obtaining information about the tail traffic endpoint.

rsvpSenderRange

Refer to NAME - rsvpSenderRange for a full description of this command. The *rsvpSenderRange* command holds the information related to the originating routers for the MPLS tunnels being simulated in ingress cases. Sender ranges are added into the *rsvpDestinationRange* list using the *rsvpDestinationRange addSenderRange* command. Three lists are maintained by this command:

- PLR — the fast reroute point of local repair (PLR), constructed with the rsvpPlrNodeIdPair command.
- PATH TLV — a set of custom TLVs to be included in PATH messages, constructed with the rsvpCustomTLV command.
- TEAR TLV — a set of custom TLVs to be included in TEAR messages, constructed with the rsvpCustomTLV command.

NOTE Sender Range as used in this document is synonymous with Head Range of IxNetwork application.

The important options and subcommands of this command are:

rsvpSenderRange Options

Category	Member	Usage
Sender Range	enableSenderRange	Enables the sender range entry.
	fromIpAddress	The IP address of the first sender router.
	rangeCount	The number of routers in the sender range. Each sender router has an IP address one higher than its predecessor.
	lspIdStart	The start of the range of LSP IDs to be generated.
	lspIdEnd	The end of the range of LSP IDs.
General Options	refreshInterval	The value of the refresh interval, in milliseconds.
	bandwidth	The bandwidth requested for the connection, expressed in kbits/sec.
	timeoutMultiplier	The number of Hellos before a neighbor is declared dead.
Session Attributes	enableAuto SessionName	Enables the session name to be generated automatically.
	sessionName	If *enableAutoSessionName* is not set, this is the name assigned to this session.
	setupPriority	This is the session priority with respect to *taking* resources, such as preempting another session. The valid range is from 0 to 7. The highest priority is indicated by 0.

rsvpSenderRange Options

Category	Member	Usage
	holdingPriority	This is the session priority with respect to *holding* resources, such as keeping a session during preemption. The valid range is from 0 to 7. The highest priority is indicated by 0.
	enableLocalProtection Desired	This permits transit routers to use a local traffic rerouting repair mechanism, in the event of a fault on an adjacent downstream link or node. This may result in a violation of the explicit route object.
	enableLabelRecording Desired	This indicates that label information is to be included when doing a record route operation.
	enableSeStyleDesired	This indicates that the tunnel ingress node may reroute this tunnel without tearing it down. A tunnel egress node should use the SE Style when responding with an RESV message.
	enableRaSession Attribute	Enables the use of resource affinities as set by *excludeAny*, *includeAny*, and *includeAll*.
	excludeAny	Represents a set of attribute filters associated with a tunnel, any of which renders a link unacceptable.
	includeAny	Represents a set of attribute filters associated with a tunnel, any of which makes a link acceptable (with respect to this test). When all bits are set to 0 (null set), it automatically passes.
	includeAll	Represents a set of attribute filters associated with a tunnel, all of which must be present for a link to be acceptable (with respect to this test). When all bits are set to 0 (null set), it automatically passes.
Fast Reroute	enableFastReroute	Enables the use of the fast reroute feature.
	enableBandwidthProtectionDesired	Indicates that PLRs should offer bandwidth protection for the protection path.
	enableNodeProtection Desired	Indicates that PLRs should skip at least the next node for a backup path.
	enableFacilityBackup enableOneToOne BackupDesired	Indicates the basic type(s) of fast reroute techniques are requested.

rsvpSenderRange Options

Category	Member	Usage
	fastRerouteSetupPriority fastRerouteHoldingPriority	Indicate the priority for taking and holding resources along the backup path.
	fastRerouteHopLimit	Indicates the number of extra hops that may be added by a protection path.
	fastRerouteBandwidth	An estimate of the bandwidth needed for the protection path.
	fastRerouteExcludeAny fastRerouteIncludeAny fastRerouteIncludeAll	Capability filters used to dictate which backup paths are acceptable or unacceptable.
	enableSendDetour	Enables the generation of a DETOUR object for one to one operation.
Path Re-optimization	enablePathReoptimization	If true, enables the Path Re-optimization option.
	enablePeriodicReEvaluationRequest	Enables the head LSR to send periodic path re-evaluation request in every Re-Optimization Interval.
	backupLspIdPoolStart	This option helps to set the LSP Id for the re-optimized LSP.

rsvpSenderRange Subcommands

Category	Member	Usage
PATH TLV Items	clearAllPathTlv	Clears all PATH TLV list items.
	addPathTlvItem	Adds a new PATH TLV item. The PATH TLV item must have been previously configured through the use of the rsvpCustomTLV command.
	getFirstPathTlvItem getNextPathTlvItem	Accesses a particular PATH TLV item by iterating through the PATH TLV items. The data appears in the rsvpCustomTLV command.
	reEvaluationRequestInterval	Represents the time period (in milliseconds) at which the path re-evaluation request is sent by the head LSR. The default value is: 180000 ms (3 mins).
TEAR TLV Items	clearAllTearTlv	Clears all TEAR TLV list items.
	addTearTlvItem	Adds a new TEAR TLV item. The TEAR TLV item must have been previously configured through the use of the rsvpCustomTLV command.
	getFirstTearTlvItem getNextTearTlvItem	Accesses a particular TEAR TLV item by iterating through the TEAR TLV items. The data appears in the rsvpCustomTLV command.
PLR List	clearPlrList	Clears all PLR list items.
	addPlr	Adds a new PLR item. The PLR item must have been previously configured through the use of

rsvpSenderRange Subcommands

Category	Member	Usage
		the rsvpPlrNodeIdPair command.
	getFirstPlr getNextPlr	Accesses a particular PLR item by iterating through the PLR items. The data appears in the rsvpPlrNodeIdPair command.
	addTunnelHeadToLeaf delTunnelHeadToLeaf getTunnelHeadToLeaf setTunnelHeadToLeaf getFirstTunnelHeadToLeaf getNextTunnelHeadToLeaf clearAllTunnelHeadToLeaf	Adds, deletes, and accesses the RSVP head to leaf tunnels.
	addHeadTrafficEndPoint delHeadTrafficEndPoint getHeadTrafficEndPoint setHeadTrafficEndPoint getFirstHeadTrafficEndPoint getNextHeadTrafficEndPoint clearAllHeadTrafficEndPoint	Adds, deletes, and accesses the RSVP traffic end-points.

rsvpEroItem

Refer to NAME - rsvpEroItem for a full description of this command. The *rsvpEroItem* holds the information related to an ERO item used in an ingress mode. ERO items are added into the *rsvpDestinationRange* list using the *rsvpDestinationRange addEroItem* command. The important options of this command are:

rsvpEroItem Options

Member	Usage
type	The type of contents in the ERO entry. Either IP (IPv4 address) or autonomous system (AS).
ipAddress	If the *type* field is *IP*, then this is the ERO value as an IP address prefix.
asNumber	If the *type* field is *AS*, then this is the ERO value as an autonomous system N=number.
prefixLength	If the *type* field is *IP*, then this defines the prefix length of the DUT IP address.
enableLooseFlag	Indicates whether the ERO item is to be considered a LOOSE item or a STRICT item.

rsvpRroItem

Refer to NAME - rsvpRroItem for a full description of this command. The *rsvpRroItem* holds the information related to an RRO item used for egress mode. RRO items are added into the *rsvpDestinationRange* list using the *rsvpDestinationRange addRroItem* command. The important options of this command are:

rsvpRroItem Options

Member	Usage
type	The type of contents in the RRO entry. Either IP (IPv4 address) or

rsvpRroItem Options

Member	Usage
	label.
ipAddress	If the *type* field is *IP*, then this is the RRO value as an IPv4 address.
label	If the *type* field is *Label*, then this is the RRO value as an assigned label.
enableProtection Available	If the *type* field is *IP*, then this indicates that local protection is made available for the downstream link.
enableProtectionInUse	If the *type* field is *IP*, then this indicates that the local protection is being used currently to maintain this tunnel.
cType	If the *type* field is *IP*, then this is the C_Type of the included Label Object.
enableGlobalLabel	If the *type* field is *IP*, then this indicates that the label will be understood if received on any interface.
enableBandwidth Protection	Indicates that bandwidth protection is available for a protection path.
enableNodeProtection	Indicates that node protection is available for the protection path.

rsvpPlrNodeIdPair

Refer to NAME - rsvpPlrNodeIdPair for a full description of this command. The *rsvpPlrNodeIdPair* command holds a single pair of items related to the DETOUR object used for RSVP fast reroute. The PLR is added to a sender range using the rsvpSenderRange *addPlr*command. The important options of this command are:

rsvpPlrNodeIdPair Options

Member	Usage
plrId	The IPv4 address identifying the beginning point of detour which is a PLR. Any local address on the PLR can be used.
avoidNodeId	The IPv4 address identifying the immediate downstream node that the PLR is trying to avoid. The router ID of the downstream node is the preferred value.

rsvpCustomTLV

Refer to NAME - rsvpCustomTlv for a full description of this command. The *rsvpCustomTlv* command holds a generalized type-length-value object used in many RSVP protocol messages as per RFC 3473. The important options of this command are:

rsvpCustomTlv Options

Member	Usage
tlvClass	The class distinguisher of the TLV.
cType	The C-Type distinguisher of the TLV.
data	The data associated with the TLV. The length of the TLV will be calculated from the length of this data.

rsvpTunnelHeadToLeaf

Refer to NAME - rsvpTunnelHeadToLeaf for a full description of this command. This command is used for the enhanced functionality of ERO and SERO configuration for the head range.

rsvpTunnelHeadToLeaf Options

Member	Usage
enabled	It enables or disables the ERO/SERO specific configuration.
tunnelLeafIpStart tunnelLeafCount	These together represents a range of IP addresses identifying a set of leaves for which the rsvpTunnelHeadToLeaf configuration is applicable.
isPrependDut	Enables prepend DUT to the ERO / SERO list.
dutHopType	Based on the input, the corresponding L bit in the packet is set. [RFC 3209]
dutPrefixLength	Prefix length of DUT.
isAppendTunnelLeaf	If enabled, this appends the tunnel leaf at the end of the ERO / SERO list in the packet.
tunnelLeafHopType	This is enabled if Append Leaf is enabled. Based on the input, corresponding L bit in the packet is set. [RFC 3209]
tunnelLeafPrefixLength	Prefix length of tunnel leaf.
isSendingAsEro	If enabled, the entire configuration would go as ERO.
isSendingAsSero	If enabled, the entire configuration would go as SERO. NOTE: If ERO and SERO are both enabled, then the configuration would go both as ERO and SERO for that <head, leaf tuple.

subObjectList	The sub-object list for this ERO/SERO can be configured by typing it as a string.
	• Input String:
	`NULL| [<Subobject ;< Subobject list]`
	• Subobject list:
	`NULL| [<Subobject ;< Subobject list]`
	• Subobject:
	`<AS :< 1-65535 :< S|L>| <IP :< IP Addr/<1-32 :< S|L>`
	• IP Addr:
	`<0-255.<0-255.<0-255.<0-255`
	NULL: ="
	Example `IP:2.2.2.2/ 24:S;AS:100:L;IP:33.33.33.33/32:S "`

rsvpTunnelHead TrafficEndPoint

Refer to NAME - rsvpTunnelHeadTrafficEndPoint for a full description of this command. This command configures the IP addresses to be used in the Source IP field in traffic to be sent over the LSPs originating from this Head Range.

rsvpTunnelHeadTrafficEndPointOptions

Member	Usage
endPointType	Sets the endpoint type for this head traffic endpoint, either IPv4 or IPv6.
insertExplicitTrafficItem	This inserts an IPv6 Explicit NULL as the innermost label in addition to learned label when trying to generate IPv6 traffic over the IPv4 lsp. The purpose is to route the traffic to the IPv6 Protocol Stack at the egress for routing towards the IPv6 destination.
ipCount	This is used to simulate traffic from multiple source endpoints to be sent over the LSPs originated from the Head Range. NOTE NOTE: Allows value greater than or equal to Tunnel Head IP Count. Default is 1.
ipStart	The start source IP address, one of IPv4 or IPv6, to be used for traffic to be sent over LSPs from the Head End Point.

rsvpTunnelLeafRange

Refer to NAME - rsvpTunnelLeafRange for a full description of this command. It describes a range of tunnel leaf endpoints when the emulation type in Tunnel Tail Ranges is set to RSVP-TE P2MP.

rsvpTunnelLeafRange

Member	Usage
enable	If true the tunnel leaf range is enabled.
ipCount	The number of IPv4 addresses in the range of Tunnel Tail addresses.
ipStart	The first IPv4 address in the range of Tunnel Tail addresses to be associated with the parent Tail Range. The P2Mp RSVP-TE LSPs will terminate the sub-lsp for each P2MP lsp in the Tail Range to the set of endpoints identified by these IPv4 addresses.
subLspDown	This is a run-time configuration option which has immediate effect on RSVP state machine, unlike most other configuration options in IxNetwork, which require config object disable/enable for change to take effect. NOTE NOTE: If this field is enabled and an lsp to the leaf or set of leaves identified is already in up state, the corresponding sub-lsps are torn down immediately by sending ResvTears to the ingress. If the field is enabled, Path messages destined to these leaves are discarded, thus simulating the sub-lsps to have gone down. This field is applicable only for Tail Ranges configured under the egress Tail Ranges.

rsvpTunnelTail TrafficEndPoint

Refer to NAME - rsvpTunnelTailTrafficEndPoint for a full description of this command. This command configures the IP addresses to be used in the Destination IP field in traffic to be sent over the LSPs terminating on this Tail Range.

rsvpTunnelTailTrafficEndPoint

Member	Usage
endPointType	Sets the endpoint type for this head traffic endpoint, either IPv4 or IPv6.
ipCount	This indicates that the number of destination IPs to which the traffic sent over the P2MP RSVP-TE tunnel is destined. NOTE NOTE: The minimum and default value is 1.
ipStart	The Start Destination IP address for traffic that is sent over the P2MP RSVP-TE tunnel. Normally this is an IPv4 or IPv6 Multicast address.

LACP

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to LACP. The LACP-related commands are:

- lacpLearnedInfo — provides access to the learned actor and partner information.
- lacpLink — a container used to hold configuration of the LACP links.
- lacpServer — a container used to hold the configuration of the LACP server.

lacpLearnedInfo

Refer to NAME - lacpLearnedInfo for a full description of this command. The *lacpLearnedInfo* command fetches and describes the learned data actor and partner nformation. (This information will be visible only for a GRE interface.) The important subcommands and options of this command are:

lacpLearnedInfo Options

Member	Usage
actorSystemId	(read only) The learned Actor system identifier, in 6 byte format.
actorSystemPriority	(read only) The learned Actor system priority, in hexadecimal format.
actorPortNumber	(read only) The learned Actor port number in hexadecimal format.
actorPortPriority	(read only) The learned Actor port priority, in hexadecimal format.
actorOperationalKey	(read only) The learned Actor operation key, in hexadecimal format.
administrativeKey	(read only) This field controls the aggregation of ports of the same system with similar Actor Key.
partnerSystemId	(read only) The learned Partner system identifier, in 6 byte format.
otherLagMemberCount	(read only) The total number of ports,excluding the individual port that are a part of the LAG.
otherLagMemberDetails	(read only) The detailed information of the other member ports of the same LAG, visible in card:port format.
partnerPortNumber	(read only) The learned Partner port number in hexadecimal format.
partnerOperationalKey	(read only) The learned Partner operation key, in hexadecimal format.
partnerPortPriority	(read only) The learned Partner port priority, in hexadecimal format.
partnerSystemPriority	(read only) The learned Partner system priority, in hexadecimal format.
partnerCollector MaxDelay	(read only) The learned maximum Collection Delay for the port, in microseconds.

lacpLearnedInfo Options

Member	Usage
enabledAggregation	(read only) Learned Link Aggregation status of the link, whether Aggregated or Not Aggregated.
partnerLacpActivity	(read only) The learned Partner LACP activity mode, either Passive or Active.
partnerLinkAggregationStatus	(read only) The learned aggregatability status of the partner, whether Aggregatable or Individual.
partnerDistributingFlag	(read only) The learned Partner Distributing Flag status, either True of False. If True, the Distributing Flag is enabled.
partnerLacpTimeout	(read only) The learned Partner LACPDU timeout mode, either Long or Short.
partnerSyncFlag	(read only) The learned Partner synchronized status, either OUT_OF_SYNC/IN_SYNC.
partnerCollectingFlag	(read only) The learned Partner Collecting Flag status, either True of False. If True, the Collecting Flag is enabled.
partnerDefaultedFlag	(read only) The learned Partner Defaulted Flag status, either True of False. If True, the Defaulted Flag is enabled.
partnerExpiredFlag	(read only) The learned Partner Expired Flag status, either True of False. If True, the Expired Flag is enabled.
actorLacpActivity	(read only) The learned Actor LACP activity mode, either Passive or Active.
actorLinkAggregation Status	(read only) The aggregatability status of the actor, whether Aggregatable or Individual.
actorDistributingFlag	(read only) The learned Actor Distributing Flag status, either True of False. If True, the Distributing Flag is enabled.
actorLacpTimeout	(read only) The learned Actor LACPDU timeout mode, either Long or Short.
actorSyncFlag	(read only) The learned Actor synchronized status, either OUT_OF_SYNC/IN_SYNC.
actorCollectingFlag	(read only) The learned Actor Collecting Flag status, either True of False. If True, the Collecting Flag is enabled.
actorDefaultedFlag	(read only) The learned Actor Defaulted Flag status, either True of False. If True, the Defaulted Flag is enabled.
actorExpiredFlag	(read only) The learned Actor Expired Flag status, either True of False. If True, the Expired Flag is enabled.

lacpLink

Refer to NAME - lacpLink for a full description of this command. The *lacLink* command configures the link options for the port. The important subcommands and options of this command are:

lacpLink Options

Member	Usage
enabled	If true, this particular LACP link entry is enabled.
actorSystemId	Specifies the system identifier for the link Actor. It is a 6 byte field, with a default of 00-00-00-00-00-01. Min: 00-00-00-00-00-00 Max: FF-FF-FF-FF-FF-FF
actorSystemPriority	Specifies the system priority of the link Actor. It is a 2 byte field, with a default or 1. Min: 0 Max: 65535
actorPortNumber	The port number assigned to the port by the Actor (the System sending the PDU). It is a 2 byte field with a default of 1. Min: 0 Max: 65535
autoPickPortMac	If true, the source MAC is the interface MAC address. It is enabled by default.
portMac	Specifies the port MAC address. This option is grayed out if Auto Pick Port MAC is selected. This is a 6 byte field.
actorPortPriority	Specifies the port priority of the link Actor. It is a 2 byte field, with a default or 1. Min: 0 Max: 65535
administrativeKey	Controls the aggregation of ports of the same system with similar Actor Key.
actorKey	The operational Key value assigned to the port by the Actor. This is a 2 byte field with a default of 1. Min: 0 Max: 65535
enablePreservePartnerInfo	If true, and a new link of the same port is enabled, transmitted LACPDUs carry the information for learned partner. If false, the LACPDUs carry partner information as 0.
collectorMaxDelay	The maximum time in microseconds that the Frame Collector may delay the delivery of a frame received from an Aggregator to its MAC client. This is a 2 byte field with a default 0.
lacpduPeriodic TimeInterval	Defines how frequently LACPDUs are sent to the link partner.
lacpTimeout	This timer is used to detect whether received protocol information has expired.

lacpLink Options

Member	Usage
lacpActivity	Sets the value of LACPs Actor activity, either passive or active.
supportRespondingToMarker	When true, LACP doesn't respond to MARKER request PDUs from the partner.
sendPeriodicMarker Request	When true, periodic Marker Request PDUs are sent after both actor and partner are IN SYNC and our state is aggregated. The moment we come out of this state, the periodic sending of Marker will be stopped.
markerRequestMode	Sets the marker request mode for the Actor link.
interMarkerPDUDelay	Sets the value for the Marker Request Mode, in seconds. If the Marker Request Mode is Fixed, then enter a single number from 0 to 255 (default 6). If Marker Request Mode is Random, then enter a number range (each endpoint from 0 to 255, with a default of 0 - 15).
sendMarkerRequestOnLagChange	If true, causes LACP to send a Marker PDU on the following situations: • System Priority has been modified • System Id has been modified • Actor Key has been modified Port Number/Port Priority has been modified while we are in Individual mode.
aggregationFlagState	If true, sets the port status to allow aggregation. Default: true.
syncFlag	If true, the actor port state is set to True based on Tx and Rx state machines. Otherwise, the flag in LACPDU remains reset for all packets sent.
markerResponseWait Time	The number of seconds to wait for Marker Response after sending a Marker Request. After this time, the Marker Response Timeout Count is incremented. If a marker response does arrive for the request after this timeout, it is not considered as a legitimate response.
distributingFlag	If true, the actor port state Distributing is set to true based on Tx and Rx state machines. Otherwise, the flag in LACPDU remains reset for all packets sent.
collectingFlag	If true, the actor port state Collecting is set to true based on Tx and Rx state machines. Otherwise, the flag in LACPDU remains reset for all packets sent.
name	Sets a text string as the name of the link.

lacpServer

Refer to NAME - lacpServer for a full description of this command. The *lacpServer* command configures the server options for the port.

lacpServer Subcommands

Member	Usage
select	Accesses the LACP component of the protocol server for the indicated port.
addLink	Adds a link to the list of LACP links at the current position.
delLink	Removes a link from the list of LACP links at the current position.
getLink	Finds the link indicated by the linkName, sets it to the `current' link and retrieves the data so that it can be viewed and modified.
setLink	Replaces the data associated with a link, either the link with the indicated linkName or the currently selected neighbor, if the linkName argument is omitted.
getFirstLink	Makes the first link in the list the `current' link and retrieves the data
getNextLink	Makes the next link in the list the `current' link and retrieves the data.
clearAllLinks	Clears all the links in the list.
write	Writes or commits the changes in IxHAL to hardware for the LACP related parameters on the port selected with the select subcommand.
sendUpdate	This command sends an update to the link partners after changing a link's configuration parameters.
sendMarker Request	Sends Marker Requests at will. The contents of the marker PDU contain the current view of partner (which can be defaulted if no partner is present). The marker will be sent regardless of which state the link is in.
startPDU	Used to start PDUs related to LACP (for example, LACPDU, Marker Request PDU, Marker Response PDU) while the protocol is running on the port.
stopPDU	Used to stop PDUs related to LACP (for example, LACPDU, Marker Request PDU, Marker Response PDU) while the protocol is running on the port.
requestLacp LearnedInfo	Requests learned information from the link partners.
getLacpLearned Info	Retrieves the learned LACP information after a requestLacpLearnedInfo command.
setDefault	Sets default values for all configuration options.

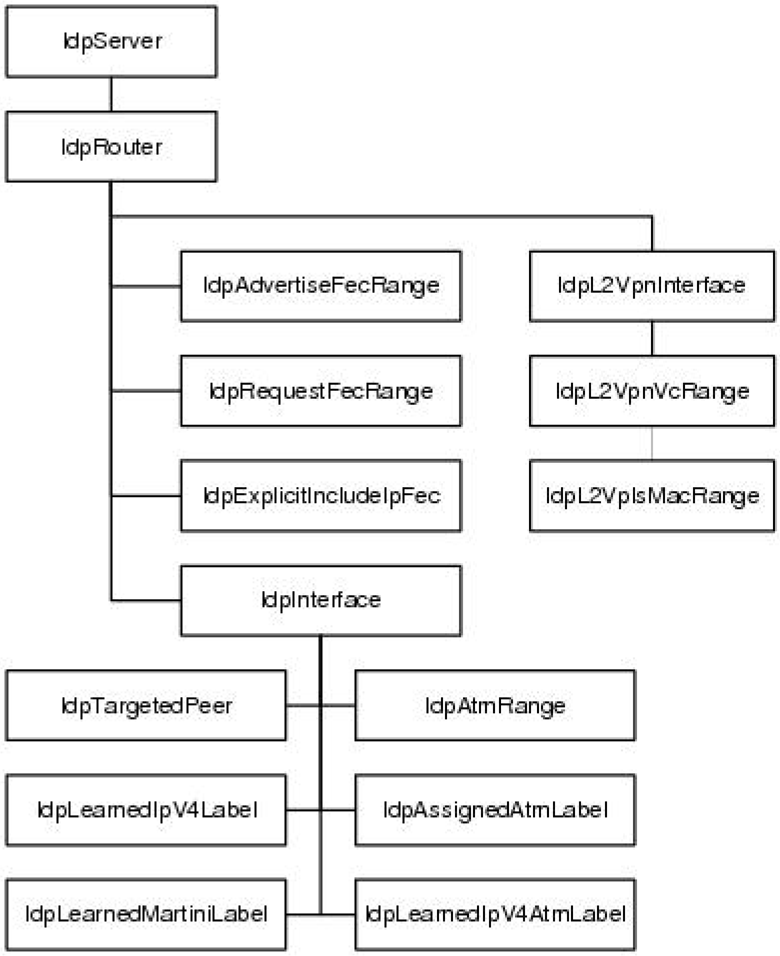
LDP

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to LDP. The LDP-related commands are:

- ldpServer — provides access to the LDP part of a port's protocol server.
- ldpRouter — a container used to hold two lists associate with the router: advertised forwarding equivalence classes (FECs) and interfaces.
- ldpAdvertiseFecRange — a set of FECs to be included in *ldpRouter.*
- ldpRequestFecRange — a request for a set of FEC ranges to be included in *ldpRouter.*
- ldpInterface — a network interface to be included in *ldpRouter.*
- ldpTargetedPeer — a targeted peer to be included in *ldpInterface.*
- ldpLearnedMartini Label — a single Martini label from the list maintained by *ldpInterface.*
- ldpLearnedIpV4 Label — a single IPv4 label from the list maintained by *ldpInterface.*
- ldpAtmRange — a single ATM range of VPIs and VCIs used for ATM sessions.
- ldpAssignedAtmLabel — a single ATM label from the list maintained by *ldpInterface.*
- ldpLearnedIPv4AtmLabel — a single IPv4 ATM label from the list maintained by *ldpInterface.*
- ldpL2VpnInterface — a single interface on a simulated router to be used in establishing Layer 2 VPNs; associated with an *ldpRouter.*
- ldpExplicitIncludeIpFec — an instance of a particular FEC that may be used to filter learned routes for an interface. A list of these is maintained by ldpInterface.
- ldpL2VpnVcRange — a VC range associated with a *ldpL2VpnInterface.*
- ldpL2VplsMac Range — an associated MAC range for a VC range. A list of these is maintained in *ldpL2VpnVcRange*.
- ldpMulticastLeafRange — adds a multicast leaf range.
- ldpOpaqueValueElement — adds an opaque value element.
- ldpLearnedMulticastLabel — a single learned multicast label from the list maintained by *ldpInterface.*
- ldpLearnedOpaqueValueElement — a single learned opaque value element from the list maintained by *ldpInterface.*

These commands and the data that they maintain are arranged in an hierarchy as shown in the following figure.

LDP Command Hierarchy



ldpServer

Refer to NAME - ldpServer for a full description of this command. The *ldpServer* command is necessary in order to access the LDP component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
ldpServer select 1 5 2
```

will access the LDP server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routes occurs using the *ldpRouter* command and subsidiary commands. The important options and subcommands of this command are shown in the following table.

ldpServer Options

Member	Usage
enableDiscardSelf AdvertiseFecs	Discards learned labels from the DUT that match any of the enabled configured IPv4 FEC ranges.
enableLabelExchangeOverLsp	Enables protocol sessions to run over established LSPs. If true, when a protocol packet is transmitted by an Ixia port and the IP details match an established LSP, the packet is MPLS encapsulated. The MPLS label is set to the value learned from the LSP

ldpServer Options

Member	Usage
helloHoldTime helloInterval enableHelloJitter	The timers associated with maintaining adjacencies based on Hello messages. The last option allows staggered transmission of many Hello messages.
keepAliveHoldTime keepAliveInterval	The timers associated with maintaining adjacencies based on PDU and keep-alive messages.
targetedHelloHoldTime targetedHelloInterval	The timers associated with maintaining targeted peer adjacencies based on Hello messages.

ldpServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouters	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the *ldpRouter* command.
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. The data appears in the *ldpRouter* command.
delRouter	Deletes a particular router from the list.
setRouter	It is possible to change LDP router configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the router's configuration with *ldpRouter.* (*ldpRouter* has capabilities for modifying elements of underlying objects as well.) 2. Use the *ldpServer setRouter* command to set the values from *ldpRouter* into IxHal. 3. Use the *ldpServer write* command to write the changes to the hardware.
generateIpV4 Streams	Once an LDP simulation has been set up with the other commands in this section, this subcommand automatically generates traffic to all IPv4 addresses in all defined route ranges.

ldpRouter

Refer to NAME - ldpRouter for a full description of this command. The *ldpRouter* command represents a simulated router. In addition to some identifying options, it holds four lists for the router:

- Advertise FEC Range — FECs to be advertised by the simulated router, constructed in the *ldpAdvertiseFecRange* command.
- Interfaces — router interface, constructed in the *ldpInterface* command.
- L2 VPN Interfaces — Layer 2 VPN interfaces used in establishing VPNs.
- Explicit Include List — an optional list of IP FECs used to filter received FECs. This allows the simulated router to ignore all other FECs.

The important options and subcommands of this command are shown in the following table.

ldpRouter Options

Member	Usage
enable	Enables or disables the simulated router.
routerId	The ID of the simulated router, expressed as an IP address.
enableRemoteConnect	Allows other LDP routers not on the local multicast network to connect to the simulated router.
enableL2VpnVcFecs	Enables the use of Layer 2 Virtual Circuit FECs.
enableExplicitIncludeIpFec	Enables the use of explicit include IP FEC list to filter learned LSAs.
areaAddressList	The list of area addresses to use.
enablePduRateControl interPduGap	Controls the use of PDU gaps for rate control.
enableGracefulRestart reconnectTime recoveryTime	Parameters for graceful restart.
useTransportAddress	The transport address to use.
transportAddress	The transport address.

ldpRouter Subcommands

Category	Member	Usage
FEC Ranges	clearAllAdvertiseFecRange	Clears all FEC ranges.
	addAdvertiseFecRange	Adds a new FEC range. The FEC range must have been previously configured through the use of the *ldpAdvertiseFecRange* command.
	getAdvertiseFecRange getFirstAdvertiseFecRange getNextAdvertiseFecRange	Accesses a particular FEC range either by ID, or by iterating through all of the FEC ranges. The data appears in the *ldpAdvertiseFecRange* command.
	delAdvertiseFecRange	Deletes a particular FEC range.
	setAdvertiseFecRange	It is possible to change FEC range configuration on the fly. In order to do this, the following steps are necessary:
		1. Modify the interface's configuration with *ldpAdvertiseFecRange.* 2. Use the *ldpRouter setAdvertiseFecRange* command to set the values from *ldpAdvertiseFecRange* into IxHal.

ldpRouter Subcommands

Category	Member	Usage
		3. Use the *ldpServer write* command to write the changes to the hardware.
Interfaces	clearAllInterface	Clears all interfaces.
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the *ldpInterface* command.
	delInterface	Deletes a particular interface.
	getInterface getFirstInterface getNextInterface	Accesses a particular interface either by ID or by iterating through all of the interfaces. The data appears in the *ldpInterface* command.
	setInterface	It is possible to change interface configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with *ldpInterface.* 2. Use the *ldpRouter setInterface* command to set the values from *ldpInterface* into IxHal. 3. Use the *ldpServer write* command to write the changes to the hardware.
L2VPN Interfaces	clearAllL2VpnInterface	Clears all L2 VPN interfaces.
	addL2VpnInterface	Adds a new L2 VPN interface. The interface must have been previously configured through the use of the *ldpL2VpnInterface* command.
	delL2VpnInterface	Deletes a particular L2 VPN interface.
	getFirstL2VpnInterface getNextL2VpnInterface	Accesses a particular L2 VPN interface by iterating through all of the interfaces. The data appears in the *ldpL2VPnInterface* command.
	setL2VpnInterface	It is possible to change interface configuration on the fly. In order to do this, the following steps are necessary:

ldpRouter Subcommands

Category	Member	Usage
		1. Modify the interface's configuration with *ldpL2VPNInterface.* 2. Use the *ldpRouter setL2VpnInterface* command to set the values from *ldpL2VpnInterface* into IxHal. 3. Use the *ldpServer write* command to write the changes to the hardware.
Explicit Include IP FECs	clearAllExplicitIncludeIpFec	Clears all explicit include IP FECs.
	addExplicitIncludeIpFec	Adds a explicit include IP FEC. The FEC must have been previously configured through the use of the *ldpExplicitIncludeIpFec* command.
	delExplicitIncludeIpFec	Deletes a particular explicit include IP FEC.
	getFirstExplicitIncludeIpFec getNextExplicitIncludeIpFec	Accesses a particular explicit include IP FEC by iterating through all of the interfaces. The data appears in the *ldpExplicitIncludeIpFec* command.
ldpRouter-LearnedIpV4Label	requestLdpBgpAdVplsLearnedInfo	Requests the per router BgpAdVpls learned info. If true, the value returned is 0.
	getLdpBgpAdVplsLearnedInfo	Gets the router level BGP AD Learned Info after requestLdpBgpAdVplsLearnedInfo has been called. If true, the value returned is 0.
	getFirstLdpBgpAdVplsLearnedInfo	Gets the first record of the retrieved learned info. If true, the value returned is 0.
	getNextLdpBgpAdVplsLearnedInfo	Gets the subsequent record of the retrieved learned info. If true, the value returned is 0.
multicastLeafeRange	addMulticastLeafRange	Add a new multicast leaf range to the ldp router.

ldpAdvertiseFecRange

Refer to NAME - ldpAdvertiseFecRange for a full description of this command. The *ldpAdvertiseFecRange* command describes an individual set of FECs. FECs are added into *ldpRouter* lists using the *ldpRouter addAdvertiseFecRange* command. The important options of this command are:

ldpAdvertiseFecRange Options

Member	Usage
enable	Enables the use of this FEC for the simulated router.
networkIpAddress	The IP address of the beginning of the advertised FEC.
numRoutes	The number of routes to be advertised.
maskWidth	The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0.
baseLabel	The first label to be associated with the FEC.
labelIncrementMode	Whether the same label will be associated with all FEC elements or incrementing values used.

ldpRequestFecRange

The *ldpRequestFecRange* holds the information related to a single FEC range request associated with download on demand advertising mode. Requests are added into the *ldpRouter* request FEC range list using the *ldpRouter addRequestFecRange* command. The important options of this command are:

ldpRequestFecRange Options

Member	Usage
enable	Enables the use of this request FEC range for the simulated router.
networkIpAddress	The IP address of the beginning of the requested FEC.
numRoutes	The number of routes to be requested.
maskWidth	The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0.
nextHopPeerIp	The IPv4 address of the LDP Peer that is the next hop router on this path. (0.0.0.0 indicates that requests will be sent to all of this router's peers that are in downstream on demand mode.)
enableHopCountTlv hopCount	The number of hops along the path of the LSP.
enableStaleTimer staleRequestTime	Enables the stale request timer. The stale request time value. Value range is 1 to 65.535 seconds. *(default = 300)*

ldpInterface

Refer to NAME - ldpInterface for a description of this command. The *ldpInterface* holds the information for a single interface on the LDP router. Interfaces are added into the *ldpRouter* interface list using the *ldpRouter addInterface* command. The important options and subcommands of this command are:

ldpInterface Options

Member	Usage
enable	Enables the use of this interface for the simulated router.
protocolInterface Description	The interface table entry to use for this interface.
labelSpaceId	The LDP label space used by this interface.
advertisingMode	The mode by which labels are advertised, either downstream unsolicited or downstream on demand.
requestingMode	The mode by which labels are requested. Currently, only independent is used.
discoveryMode	The discovery mode used for the interface: basic, extended, or extended Martini.
numLearnedLabels	The number of learned IPv4 and Martini labels.
enableAtmSession	Enables the establishment of ATM sessions.
atmVcDirection	Whether a virtual circuit (VC) may be used one way or both ways as an LSP.
authenticationType	The cryptographic authentication type used by the interface; one of: NULL (no authentication) or MD5. When MD5 is used, an md5Key must be configured by the user.
md5Key	Used with MD5 authentication. A user-defined string; maximum = 255 characters.
enableBFD Registration	If true, enables the use of the BFD registration with LDP option.

ldpInterface Subcommands

Category	Member	Usage
Targeted Peers	clearAllTargetedPeers	Clears all targeted peers.
	addTargetedPeers	Adds a new targeted peer. The targeted peer must have been previously configured through the use of the *ldpTargetedPeer* command.
	getTargetedPeers getFirstTargetedPeers getNextTargetedPeers	Accesses a particular targeted peer either by ID, or by iterating through all of the targeted peers. The data appears in the *ldpTargetedPeer* command.
	delTargetedPeer	Deletes a particular targeted peer.
	setTargetedPeers	It is possible to change targeted peer configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with *ldpTargetedPeer.* 2. Use the *ldpInterface setTargetedPeer* command to set the values from *ldpTargetedPeer* into IxHal.

ldpInterface Subcommands

Category	Member	Usage
		3. Use the *ldpServer write* command to write the changes to the hardware.
ATM Ranges	clearAllAtmRanges	Clears all ATM ranges. ATM ranges are used when *atmSessions* is enabled.
	addAtmRanges	Adds a new ATM range. The ATM range must have been previously configured through the use of the *ldpAtmRange* command.
	getAtmRanges getFirstAtmRanges getNextAtmRanges	Accesses a particular ATM range either by ID, or by iterating through all of the ATM ranges. The data appears in the *ldpAtmRange* command.
	delAtmRange	Deletes a particular ATM range.
	setAtmRange	It is possible to change ATM range configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with *ldpAtmRange.* 2. Use the *ldpInterface settAtmRange* command to set the values from *ldpAtmRange* into IxHal. 3. Use the *ldpServer write* command to write the changes to the hardware.
Learned Labels	requestLearnedLabels getLearnedLabelList	Use the first then the second of these commands to retrieve the learned IPv4 and Martini labels.
	getFirstLearnedIpV4Label getNextLearnedIpV4Label	Retrieves the learned IPv4 labels one at a time. The label details are available in the *ldpLearnedIpV4Label* command.
	getFirstLearnedIpV4AtmLabel getNextLearnedIpV4AtmLabel	Retrieves the learned IPv4 ATM labels one at a time.The label details are available in the *ldpLearnedIpV4AtmLabel* command.
	getFirstLearnedMartiniLabel getNextLearnedMartiniLabel	Retrieves the learned Martini labels one at a time. The label details are available in the *ldpLearnedMartiniLabel* command.
ldpInterface	getFirstLearnedMulticastIpV4Label getNextLearnedMulticastIpV4Label	Retrieves the learned multicast IPv4 labels.
ATM Assigned Labels	requestAssignedAtmLabels getAssignedAtmLabelList	Use the first then the second of these commands to retrieve the assigned ATM labels.
	getFirstAssignedAtmLabel getNextAssignedAtmLabel	Retrieves the assigned ATM labels one at a time. The label details are available in

ldpInterface Subcommands

Category	Member	Usage
		the *ldpAssignedAtmLabel* command.

ldpTargetedPeer

Refer to NAME - ldpTargetedPeer for a full description of this command. The *ldpTargetedPeer* holds information about a targeted peer to be associated with an LDP interface. Targeted peers are added to a ldpInterface using the ldpInterface *addTargetedPeer* sub-command. The optional LDP test package must be installed in order for this command to operate. The important options of this command are:

ldpTargetedPeer Options

Member	Usage
enable	Enables the use of this targeted peer.
ipAddress	The IP address of the targeted peer.
authenticationType	The cryptographic authentication type used by the targeted peer; one of: NULL (no authentication) or MD5. When MD5 is used, an md5Key must be configured by the user.
initiateTargetedHello	If true, a Targeted Hello will be sent to the LDP Peer specified by the IP address in this row.
md5Key	Used with MD5 authentication. A user-defined string; maximum = 255 characters.

ldpLearnedMartini Label

Refer to NAME - ldpLearnedMartiniLabel for a full description of this command. The *ldpLearnedMartiniLabel* command holds an element of the LDP learned Martini label list obtained in the ldpInterface command.

ldpLearnedIpV4 Label

Refer to NAME - ldpLearnedIpV4Label for a full description of this command. The *ldpLearnedIpV4Label* command holds an element of the LDP learned IPv4 label list obtained in the ldpInterface command.

ldpAtmRange

Refer to NAME - ldpAtmRange for a full description of this command. The *ldpAtmRange* command holds an range of VPI/VCI values that are used in ATM session negotiation.

ldpAssignedAtmLabel

Refer to NAME - ldpAssignedAtmLabel for a full description of this command. The *ldpAssignedAtmLabel* command holds an ATM label that was assigned as a result of an LDP session.

ldpLearnedIPv4AtmLabel

Refer to NAME - ldpLearnedIpV4AtmLabel for a full description of this command. The *ldpLearnedIpV4AtmLabel* command holds an element of the LDP ATM learned IPv4 label list obtained in the ldpInterface command.

ldpL2VpnInterface

Refer to NAME - ldpL2VpnInterface for a full description of this command. The *ldpL2VpnInterface* holds the information related to a single interface on the simulated router to be used in establishing Layer 2 VPNs. L2 VPN interfaces are added into the ldpRouter interface list using the ldpRouter *addL2VpnInterface* command. This command holds a list of virtual circuit (VC) ranges formed with the *ldpL2VpnVcRange* command. The important options and subcommands of this command are:

ldpL2VpnInterface Options

Member	Usage
enable	Enables the use of this interface for the simulated router.
type	The type of virtual circuit.
groupId	The group ID associated with all VC FEC elements of this interface.
count	The number of contiguous values of groupId that will be used in generating FECs.

ldpL2VpnInterface Subcommands

Category	Member	Usage
L2 VPN VC Ranges	clearAllL2VpnVcRanges	Clears all L2 VPN VC ranges.
	addL2VpnVcRanges	Adds a new L2 VPN VC range. The L2 VPN VC range must have been previously configured through the use of the *ldpL2VpnVcRange* command.
	getL2VpnVcRanges getFirstL2VpnVcRanges getNextL2VpnVcRanges	Accesses a particular L2 VPN VC range either by ID, or by iterating through all of the L2 VPN VC ranges. The data appears in the *ldpL2VpnVcRange* command.
	delL2VpnVcRanges	Deletes a particular L2 VPN VC range.
	setL2VpnVcRanges	It is possible to change L2 VPN VC range configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with *ldpL2VpnVcRange.* 2. Use the *ldpRouter setL2VpnVcRanges* command to set the values from *ldpL2VpnVcRange* into IxHal. 3. Use the *ldpServer write* command to write the changes to the hardware.

ldpL2VpnVcRange

Refer to NAME - ldpL2VpnVcRange for a full description of this command. The *ldpL2VpnVcRange* holds information about a VC range to be associated with an LDP L2 VPN interface. VC ranges are added to a ldpL2VpnInterface using the ldpL2VpnInterface *addL2VpnVcRange* subcommand. The important options and subcommands of this command are:

ldpL2VpnVcRange Options

Category	Member	Usage
General	enable	Enables use of this VC range.
	peerAddress	The IPv4 address of the LDP router which is the peer for this VC range.
	vcId/ vcIdStep count	The virtual circuit ID and the number of times that it will be incremented.
	enableDescription/	A description of the interface.
	enableCBit	Controls generation of the control word.
	mtu	(in octets) The 2-octet value for the maximum Transmission Unit (MTU).
	enableMtu/ mtuSize	The maximum transfer unit.
	labelMode/ labelValueStart	Indicates whether the same label or incrementing labels should be used in the VC ranges, plus the initial label.
	ceIpAddress	The IP address of attached CE endpoint. If IP type is set to IPv4, then the default is 0.0.0.0, and if the IP type is set to IPv6, then the default is 0:0:0:0:0:0:0:0.
	celpStep	The increment step to be added to each additional CE endpoints in the range of CE endpoints.
	enablePwStatusTlv	Enables the use of PW status TLV in notification messages to notify the PW status.
	fecType	The FEC type. The options are: • PW Id FEC 0x80 • Generalized Id FEC 0x81 VPLS
	enableSendPwStatus	If true, it enables a notification message with a PW status for the corresponding PW.
	downStartInterval	The duration in time after session becomes up and a notification message being sent to make the session down. Default = 30 sec
	downInterval	Time interval for which the PW status will remain down. Default = 60 sec
	upInterval	Time interval for which the PW status will remain up. Default = 30 sec
	repeatCount	The number of times to repeat Up Interval and Down Interval. Default = 1

ldpL2VpnVcRange Options

Category	Member	Usage
	provisioningModel	Editable dropdown to denote the Provisioning Model. The options are as follows: • PW Id FEC 0x80 • Generalized Id FEC 0x81 VPLS
	pwStatusCode	Editable dropdown to denote the PW status. This field is editable and the range is from 0x00000001 0xFFFFFFFF. The options are as follows: • 0 = Pw not forwarding • 1 = AC Rx Fault • 2 = AC Tx Fault • 3 = Pw Rx Fault • 4 = Pw Tx Fault
	vplsIdType	The VPLS ID Type: The options are: • AS • IP
	sourceAiiAsIp	The IP address.
	sourceAiiAsNumber	The numberical value for AS.
	sourceAiiType	Editable dropdown. The options are: • AS • IP
	targetAiiType	Editable dropdown. The options are: • AS • IP
	targetAiiAsIp	The IP address.
	targetAiiAsNumber	The numberical value for AS.
	vplsIdAsNumber	The 2 byte unsigned integer value.
	vplsIdStepAsNumber	The 2 byte unsigned integer value.
	vplsStepIdAssignedNumber	The 2 or 4 byte unsigned integer value dependent on the vplsIdType.
	vplsIdAssignedNumber	The 2 or 4 byte unsigned integer value dependent on the vplsIdType.
	vplsIdCount	The 4 byte unsigned integer.
	vplsIdStepIpAddress	The IP address.
	vplsIdIpAddress	The IP address.
	vplsIdType	Editable dropdown.

ldpL2VpnVcRange Options

Category	Member	Usage
		The options are: • AS • IP
	capableOfReassembly	If true, the VC range is capable of reassembly.
	cas	This is available only for vc type 0x0017.The available values are: • 01- an E1 trunk • 10 - a T1/ESF trunk • 11 - a T1 SF trunk
	frequency	The frequency of the VC range.
	includeSsrc	If true, the SSRC is enabled.
	ssrc	The positive value for SSRC.
	sp	Editable dropdown. The options are: • ldpL2VpnVcHexVal1= 0x00 • ldpL2VpnVcHexVal2= 0x01 • ldpL2VpnVcHexVal3= 0x02 • ldpL2VpnVcHexVal4= 0x03
	timestampMode	Editable dropdown. The options are: • Absolute • Differential
	includeTdmOption	If true, the TDM option is enabled.
	includeRtpHeader	If true, the RTP header is enabled.
	tdmBitrate	The integer value fro TDM bitrate
	includeTdmBitrate	If true, TDM bitrate option is included.
	tdmDataSize	The integer value for the data size.
	includeTdmPayload	If true, the TDM data size is enabled.
	payloadType	The integer value for Payload type. The acceptable range is 0x00 0x7F.
ATM	enableAtm/ maxNumAtmCells	Enables and generates an interface parameter with the maximum number of concatenated ATM cells.
CEM	enableCemOptions/ cemOptions	Enables and generates an interface parameter with CEM options.
	enableCemPayload/ cemPayloadBytes	Enables and generates an interface parameter with a number of CEM payload bytes.

ldpL2VpnVcRange Subcommands

Member	Usage
clearAllVplsMacRanges	Clears all VPLS MAC ranges.
addVplsMacRange	Adds a new VPLS MAC range. The VPLS MAC range must have been previously configured through the use of the *ldpL2VplsMacRange* command.
getFirstVplsMacRange getNextVplsMacRange	Accesses a particular VPLS MAC range by iterating through all of the VPLS MAC ranges. The data appears in the *ldpL2VplsMacRange* command.
delVplsMacRange	Deletes a particular VPLS MAC range.
purgeVc	Causes the DUT to unlearn all MACs.
addVcIpRange	Adds a new VC IP range.
clearAllVcMacVlanRanges	Clears all VC MAC VLAN ranges.
delVcIpRange	Deletes a particular VC IP range.
getVcIpRange	Gets a particular VC IP range.
getFirstVcIpRange getNextVcIpRange	Accesses a particular VC IP range by iterating through all of the VC IP ranges.
setVcIpRange	Sets up a particular VC IP range.
clearAllVcMacVlanRanges	Clears all VC MAC VLAN ranges.
addVcMacVlanRange	Adds a particular VC MAC VLAN range.
delVcMacVlanRange	Deletes a particular VC MAC VLAN range.
getVcMacVlanRange	Gets a particular VC MAC VLAN range.
getFirstVcMacVlanRange getNextVcMacVlanRange	Accesses a particular VC MAC VLAN range either by ID, or by iterating through all of the L2 VPN VC ranges.
setVcMacVlanRange	Sets up a particular VC MAC VLAN range.

ldpL2VplsMac Range

Refer to NAME - ldpL2VplsMacRange for a full description of this command. The *ldpL2VplsMacRange* command holds a range of MAC addresses to be associated with an LDP L2 VPN VC range. MAC ranges are added to a ldpL2VpnVcRange using the ldpL2VpnVcRange*addVplsMacRange* subcommand. The important options of this command are:

ldpL2VplsMacRange Options

Member	Usage
macAddress count	The first MAC address and the number of addresses to be used.
enableGenerateUnique	If *false*, then the same MAC addresses will be associated with all of the VCIDs in the ldpL2VpnVcRange command. If *true*, each new VCID generated in the *ldpL2VpnVcRange* command will receive unique ascending MAC addresses.

ldpL2VplsMacRange Subcommands

Member	Usage
purgeMac	Causes the DUT to unlearn this MAC.

ldpExplicitIncludeIpFec

Refer to NAME - ldpExplicitIncludeIpFec for a full description of this command. The *ldpExplicitIncludeIpFec* holds information about FEC filter to be associated with an LDP L2 VPN interface. Explicit include IP FECs are added to a ldpL2VpnInterface using the ldpL2VpnInterface *addExplicitIncludeIpFec* subcommand. The important options of this command are:

ldpExplicitIncludeIpFec Options

Member	Usage
enable	Enables use of this FEC.
networkIpAddress maskWidth	Defines the network associated with the FEC.
numNetworks	The number of networks in the FEC.
enableExactPrefix	If *true*, then the mask width (*maskWidth*) of the received FEC must match as well as the *networkIpAddress*. Otherwise, any prefix match less than or equal to *maskWidth* will allow the received FEC to be learned.

ldpMulticastLeafRange

Refer to NAME - ldpMulticastLeafRange for a full description of this command. The *ldpMulticastLeafRange* command creates a rmulticast leaf range to be associated with an LDP Router. The important options of this command are:

ldpMulticastLeafRange Options

Member	Usage
enable	Enable use of this multicast leaf range.
lspType	The type of multicast LSP. Currently only P2MP is supported. Possible values include: • p2mp
rootAddress	The root address of the multicast LSP.
rootAddressCount	The root address count for this Multicast leaf range.
rootAddressStep	The Root Address increment step. This is applicable only if Root Address Count is greater than 1.
lspCountPerRoot	This is to specify how many different LSPs are created per Root.
labelValueStart	The start label value for first leaf.
labelValueStep	The label value increment step for more than 1 range.

ldpMulticastLeafRange Subcommands

Member	Usage
setDefault	Set the range as default.
addOpaqueValueElement	Adds an opaque value element.

ldpOpaqueValueElement

Refer to NAME - ldpOpaqueValueElement for a full description of this command. The *ldpOpaqueValueElement* command creates opaque value tlvs. The important options of this command are:

ldpOpaqueValueElement Options

Member	Usage
type	The type of TLV.
length	The length of the TLV.
value	The value of the TLV.
increment	The increment value.

ldpMulticastLeafRange Sub-commands

Member	Usage
setDefault	Sets the default configurations.

ldpLearnedMulticastLabel

Refer to NAME - ldpLearnedMulticastLabel for a full description of this command. The *ldpLearnedMulticastLabel* command holds a Multicast label that was assigned as a result of an LDP session.

ldpLearnedOpaqueValueElement

Refer to NAME - ldpOpaqueValueElement for a full description of this command. The *ldpLearnedOpaqueValueElement* command holds a learned Opaque Value Element that was assigned as a result of an LDP session.

MPLS OAM

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to MPLS OAM. The MPLS OAM-related commands are:

- mplsOamServer — a container used to hold the configuration of the MPLS OAM server.
- mplsOamRouter — a container used to hold the configuration of the MPLS OAM router.
- mplsOamInterface — holds the information related to a single interface on the simulated router.
- bgp4VpnL2Site — holds information about a VPN Layer 2 site.
- ldpServer — accesses the LDP component of the protocol server for a particular port.
- rsvpServer — accesses the RSVP component of the protocol server for a particular port.
- mplsOamGeneralLearnedInfo — holds the lists of the general learned route information.
- mplsOamTriggeredPingLearnedInfo — holds the lists of the triggered ping learned information.

mplsOamServer

Refer to NAME - mplsOamServer for a full description of this command. The *mplsOamServer* command configures the server options for the port.

mplsOamServer Subcommands

Member	Usage
select	This is used to select the chassis, card, and port to operate on.
getRouter	This is used to get the mplsOamRouter instance from the mplsOamServer. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
setRouter	This is used to save/set a particular mplsOamRouter instance to the mplsOamServer. Only one router is supported at this time. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
getFirstRouter	This is used to get the first instance from the list of mplsOamRouter instances configured under mplsOamServer. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
getNextRouter	This is used to get the next instance from the list of mplsOamRouter instances configured under mplsOamServer. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
delRouter	This is used to delete the mplsOamRouter instance from the mplsOamServer.

mplsOamServer Subcommands

Member	Usage
write	This is used to send any changes made with mplsOamServer setRouter, or mplsOamServer getRouter to the protocol server for immediate application.
showRouteNames	This is used to display the names of all routes, configured under mplsOamServer. NOTE At present, this shows the name of only one router.

mplsOamRouter

Refer to NAME - mplsOamRouter for a full description of this command. The *mplsOamRouter* command configures the server options for the port.

mplsOamRouter Subcommands

Member	Usage
getInterface	This is used to get a particular instance of mplsOamInterface from the mplsOamRouter object. • Arguments: interfaceName (string) • Return value: 0 (success) • ErrorValue: non-zero value (failure)
setInterface	This is used to save/set a particular instance of mplsOamInterface to the mplsOamRouter. • Arguments: interfaceName (string) • Return value: 0 (success) • ErrorValue: non-zero value (failure)
getFirstInterface	This is used to get the first instance from the list of mplsOamInterfaces configured under mplsOamRouter. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
getNextInterface	This is used to get the next instance from the list of mplsOamInterfaces configured under mplsOamRouter. This should be called after a call to the getFirstInterface. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
delInterface	This is used to remove a particular mplsOamInterface instance from the mplsOamRouter object.
set	This is used to set the current configuration of the protocol server on the most recently selected port to its hardware.
get	This is used to get the current mplsOamRouter configuration.
enabled	If true, the mplsOamRouter object is enabled.
routerId	The assigned router ID

mplsOamInterface

Refer to NAME - mplsOamInterface for a full description of this command. The *mplsOamInterface* holds the information related to a single interface on the simulated router. The important options and subcommands of this command are:

mplsOamInterface Options

Member	Usage
set	This is used to save/set a particular instance of mplsOamInterface to the mplsOamServer. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
get	This is used to get a particular instance of mplsOamInterface from the mplsOamServer. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
destination Ipv4Address	The allocated destination IPv4 address for this interface.
echoRequestInterval	The minimum interval, in milliseconds, between received Echo packets that this interface is capable of supporting.
echoResponseTimeout	The minimum tiomeout interval, in milliseconds, between received Echo packets that this interface is capable of supporting.
enableFecValidation	If true, the check box is selected to enable validation.
enablePeriodicPing	If true, the router is pinged at regular intervals.
enabled	If true, it enables or disables the simulated router.
flapTxIntervals	The number of seconds between route flaps for MPLS OAM. A value of zero means no flapping.
enableIncludePadTlv	If true, selects the check box to include Pad TLV.
enableIncludeVendorEnterpriseNumberTlv	If true, selects the checkbox to include the the TLV number of the vendor organization
minRxInterval	The minimum interval, in milliseconds, between received BFD Control packets that this interface is capable of supporting
multiplier	The negotiated transmit interval, multiplied by this value, provides the detection time for the interface
padTlvFirstOctet	Select the first octate of the Pad TLV.

mplsOamInterface Options

Member	Usage
padTlvLength	Specifies the length of the Pad TLV
replyMode	Selects the mode of reply
txInterval	The minimum interval, in milliseconds, that the interface would like to use when transmitting Control packets.
vendorEnterpriseNumber	Specifies the enterprise number of the vendor.

bgp4VpnL2Site

Refer to NAME - bgp4VpnL2Site for a full description of this command. The *bgp4VpnL2Site* holds information about a VPN Layer 2 site. The important options and subcommands of this command are:

bgp4VpnL2Site Options

Member	Usage
enableBfdVccv	If true, enables the use of BFD VCCV. *(default = false)*
enableVccvPing	If true, enables VCCV ping. *(default = false)*

ldpServer

Refer to NAME - ldpServer for a full description of this command. The *ldpServer* accesses the LDP component of the protocol server for a particular port. The important options and subcommands of this command are:

ldpServer Options

Member	Usage
enableUseTransportLabelsForMplsOam	If true, enables label exchange over LSP.

rsvpServer

Refer to NAME - rsvpServer for a full description of this command. The *rsvpServer* accesses the RSVP component of the protocol server for a particular port. The important options and subcommands of this command are:

rsvpServer Options

Member	Usage
enableUseTransportLabelsForMplsOam	If true, enables label exchange over LSP.

mplsOamGeneralLearnedInfo

Refer to NAME - mplsOamGeneralLearnedInfo for a full description of this command. The *mplsOamGeneralLearnedInfo* holds the lists of the general learned route information. The important options and subcommands of this command are:

mplsOamGeneralLearnedInfo Options

Member	Usage
averageRtt	(read only) The learned average MPLS OAM Round-Trip-Time.
fec	(read only) Forwarding equivalence class (FEC) type.
incomingLabelStack	(read only) BGP sends the assigned labels information to this MPLS OAM module which is used for validation of FEC stack received in an echo request.

mplsOamGeneralLearnedInfo Options

Member	Usage
incomingLspLabel	(read only) The incoming LSP label value.
incomingPwLabel	(read only) The incoming PW label value.
lspPingReachability	(read only) Specifies whether the queried LSP Ping could be reached or not.
maxRtt	(read only) Specifies the maximum Round Trip Time.
minRtt	(read only) Specifies the minimum Round Trip Time.
bfdSessionMyState	(read only) This window provides read-only information about the state of BFD interface on the specified emulated router.
myDiscriminator	(read only) The discriminator for the session on this interface.
myIpAddress	(read only) The IP address for this interface.
outgoingLabelStack	(read only) BGP sends the assigned labels information to this MPLS OAM module which is used for validation of FEC outgoing Label stack that is received in an echo request.
outgoingLspLabel	(read only) The outgoing LSP label value.
outgoingPwLabel	(read only) The outgoing PW label value.
bfdSessionPeerState	(read only) The state of the far side of the BFD session, either active or not.
peerDiscriminator	(read only) The discriminator for the far side of the session.
peerIpAddress	(read only) The learnt IP address for the session.
pingAttempts	(read only) Specifies the number of ping attempts.
pingFailures	(read only) Specifies the number of ping failures.
pingReplyTx	(read only) Specifies the number of ping reply transmitted at regular intervals.
pingRequestRx	(read only) Specifies the number of ping request received at regular intervals.
pingSuccess	(read only) Specifies the rate of ping success.
receivedMinRxInterval	(read only) The minimum receive interval, in milliseconds, for the far side of the session.
receivedMultiplier	(read only) The number of received negotiated transmit intervals when multiplied by this value, provides the detection time for the interface.
receivedPeerFlags	(read only) The number of peer generated flags received.
receivedTxInterval	(read only) The minimum transmit interval, in milliseconds, for the far side of the session.
returnCode	(read only) The return code value.
returnSubCode	(read only) The return subcode value.

mplsOamTriggeredPingLearnedInfo

Refer to NAME - mplsOamTriggeredPingLearnedInfo for a full description of this command. The *mplsOamTriggeredPingLearnedInfo* holds the lists of the triggered ping learned information. The important options and subcommands of this command are:

mplsOamTriggeredPingLearnedInfo Options

Member	Usage
fec	(read only) Forwarding equivalence class (FEC) type.
incomingLabelStack	(read only) The incoming label stack value.
outgoingLabelStack	(read only) The outgoing label stack value.
peerIpAddress	(read only) The learnt IP address for the session.
reachability	(read only) Specifies whether the queried MEP could be reached or not, Failure/ Partial/Complete.
returnCode	(read only) The return code value.
returnSubCode	(read only) The return subcode value.
rtt	(read only) Denotes the Round Trip Time.

Link OAM

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to Link OAM. The Link OAM-related commands are:

- linkOamServer — a container used to hold the configuration of the Link OAM server.
- linkOamLink — a container used to hold the configuration of the Link OAM links.
- linkOamInterface — holds the information related to a single interface on the simulated link.
- linkOamServer — configures the error symbol period event tlv in the event notification packet.
- linkOamFrameTlv — configures the errored frame event tlv in the event notification packet.
- linkOamPeriodTlv — configures the errored frame period event tlv in the event notification packet.
- linkOamSSTlv — configures the errored frame seconds summary event tlv in the event notification packet.
- linkOamOrgEventTlv — configures the organization specific event tlv in the event notification packet.
- linkOamOrgInfoTlv — configures the organization specific information tlv in the event notification packet.
- linkOamVar Container — configures the variable response container.
- linkOamOrgTlv — configures the organization specific tlv.
- linkOamDiscLearnedInfo — fetches and describes the discovered learned data.
- linkOamEventNotifnInfo — fetches and describes the learned data for event notification.
- linkOamVarRequest LearnedInfo — fetches and describes the learned data for variable request.

linkOamServer

Refer to NAME - linkOamServer for a full description of this command. The *linkOamServer* command configures the server options for the port.

linkOamServer Subcommands

Member	Usage
addLink	This is used to add a single link to the linkOamServer object. • Arguments: linkName (string) • Return value: 0 (success) • ErrorValue: non-zero value (failure) **Note**: At present, only one link can be added to the linkOamServer.
getLink	This is used for getting the linkOamLink instance from the linkOamServer. • Return value: 0 (success)

linkOamServer Subcommands

Member	Usage
	- ErrorValue: non-zero value (failure)
setLink	This is used for saving/setting a particular linkOamLink instance to the linkOamServer. Only one link is supported at this time. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
showLinkNames	This is used to display the names of all links, configured under linkOamServer. **Note**: At present, this shows the name of only one link.

linkOamLink

Refer to NAME - linkOamLink for a full description of this command. The *linkOamLink* command configures the link options for the port. The important subcommands and options of this command are:

linkOamLink Options

Member	Usage
enabled	If true, this particular Link OAM link entry is enabled.
macAddress	Specifies the MAC address of the local DTE.
operationMode	Specifies the OAM operation mode in Ixia port. One of: - Active (*default*) - Passive
informationPduCount PerSecond	Specifies the timer that is used to ensure that OAM sub layer adheres to maximum number of OAMPDUs per second, and emits at least one OAMPDU per second. Default: 1 Min: 1 Max: 10
localLostLinkTimer	Indicates the timer that is used to reset the Discovery state machine of local DTE when it is not receiving any Information PDU from remote DTE. Default: 5 seconds Min: 2 seconds Max: 90 seconds
supportsVariable Retrieval	If true, enables the variable retrieval support in ixia port. Default is True.
supportsRemote Loopback	If true, enables the remote loopback support in ixia port. Default is True.
supportsInterpreting LinkEvents	If true, enables the link event interpreting support in ixia port. Default is True.

Member	Usage
supportsUnidirectionalMode	If true, enables the unidirectional mode in ixia port. Default is True.
enableCriticalEvent	If true, helps to indicate a critical event to remote peer by setting Critical Event bit in the flags field of Information PDUs transmitted thereafter. Default is False.
enableLinkFault	If true, indicates that a fault has occurred in the receive direction by setting Link Fault bit in the flags field of Information PDUs transmitted thereafter. Also the local DTE will move into FAULT state. Default is False.
enableDyingGasp	If true, helps to indicate an unrecoverable local failure condition to remote peer by setting Dying Gasp bit in the flags field of Information PDUs transmitted thereafter. Default is False.
enableVariable Response	If true, enables Variable response. This is used to determine whether to respond to variable request. Default value is True. **Note**: This is done for negative testing.
enableLoopback Response	If true, enables Loopback response. Default is True. **Note**: This is done for negative testing.
disableInformationPduTx	If true, it controls the transmission of information PDU. Default is False.
disableNonInformationPduTx	If true, it controls the transmission of non- information PDU. Default is False.
overrideLocal Evaluating	If true, the local evaluating bit transmitted within Local Information TLVs is overridden. Default is False. **Note**: This is done for negative testing.
overrideLocalSatisfied	If true, the local_satisfied variable used for discovery is overridden and the state of the local DTE is re-calculated accordingly. Default is False. **Note**: This is done for negative testing.
overrideLocalStable	If true, the local stable flag is overridden. Default is False. **Note**: This is done for negative testing.

Member	Usage
overrideRemote Evaluating	If true, the remote evaluating bit transmitted within Local Information TLVs is overridden. Default is False. **Note**: This is done for negative testing.
overrideRemoteStable	If true, the remote stable bit transmitted within Local Information TLVs is overridden. Default is False. **Note**: This is done for negative testing.
overrideRevision	If true, overrides the revision field. **Note**: If this not true, then the Revision field is disabled.
overrideSequence Number	If true, the current sequence number can be overridden. Default is False.
revision	Specifies the revision description. Default: 0 Max: 65535
sequenceNumber	Indicates the sequence number with which to override the current sequence number. This field remains false except when Override Sequence Number option is true. Default: 0 Max: 65535
maxOamPduSize	The maximum OAMPDU size supported by local DTE. Default: 1500 octets. Min: 64 Max: 1500 in octets
version	Specifies the version supported by this local DTE. It accepts integer values. Default: 0x01 Min: 0x00 Max: 0xFF
oui	Contains the 24 bit Organizationally Unique Identifier. Default: 00 01 00.
vendorSpecific Information	Contains the vendor specific information. This is used to differentiate the product modes/version of a vendor. Default: 00000000.
linkEventTxMode	Indicates the link event tx mode. One of: • Single (*Default*) • Periodic

Member	Usage
eventInterval	Indicates the periodic interval of event pdu when event pdu is to be sent periodically. Default: 1 sec Min: 1 sec Max: 10 secs
updateRequired	If true, sends the updated parameters information.
loopbackTimeout	Indicates the loopback timeout in milliseconds. Loopback timeout is the time period till which the local DTE will wait for the remote DTE's response to the Loopback Control PDU transmitted. Default: 1000 ms
loopbackCommand	This contains the options of Enable OAM Remote Loopback. One of: • enableLoopback (*default*) • disableLoopback
variableResponse Timeout	The maximum time in seconds to wait for the variable response pdu. Default: 1 sec Min: 500 ms Max: 10 sec

linkOamLink Commands

Member	Usage
addInterface	Used to add a *linkOamInterface* object to the *linkOamLink* object. • Arguments: interfaceName (string) • Return value: 0 (success) • ErrorValue: non-zero value failure
delInterface	Used to remove a particular *linkOamInterface* from the *linkOamLink* object. • Arguments: interfaceName (string) • Return value: 0 (success) • ErrorValue: non-zero value failure
getInterface	Used to get a particular instance of *linkOamInterface* from the *linkOamLink* object. • Arguments: interfaceName (string) • Return value: 0 (success) • ErrorValue: non-zero value failure
setInterface	Used for setting a particular instance of the *linkOamInterface* to the *linkOamLink*.

Member	Usage
	- Arguments: interfaceName (string) - Return value: 0 (success) - ErrorValue: non-zero value failure
getFirstInterface	Used to get the first instance from the list of *linkOamInterfaces* configured under *linkOamLink*. - Return value: 0 (success) - ErrorValue: non-zero value failure
getNextInterface	Used to get the next instance from the list of *linkOamInterfaces* configured under *linkOamLink*. This should be called after a call to the *getFirstInterface*. - Return value: 0 (success) - ErrorValue: non-zero value failure
clearAllInterfaces	Deletes all the *linkOamInterfaces* configured under *linkOamLink*. - Return value: 0 (success) - ErrorValue: non-zero value failure
showInterfaceNames	Used to display the names of the all the *linkOamInterfaces* configured under the *linkOamLink* object.
getErroredSymbol PeriodEventTlv	Gets the Errored Symbol Period Event Tlv object instance configured under *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
setErroredSymbol PeriodEventTlv	Sets the instance of the Errored Symbol Period Event Tlv object to the *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
getErroredFrame EventTlv	Gets the Errored Frame Event Tlv object instance configured under *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
setErroredFrame EventTlv	Sets the instance of the Errored Frame Event Tlv object to the *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
getErroredFrame PeriodEventTlv	Gets the Errored Frame Period Event Tlv object instance configured under *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
setErroredFrame PeriodEventTlv	Sets the instance of the Errored Frame Period Event Tlv object to the *linkOamLink* object.

Member	Usage
	- Return value: 0 (success) - ErrorValue: non-zero value failure
getErroredFrame SSEventTlv	Gets the Errored Frame Seconds Summary Event Tlv object instance configured under *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
setErroredFrame SSEventTlv	Sets the instance of the Errored Frame Seconds Summary Event Tlv object to the *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
getOrgSpecEventTlv	Gets the Organization Specific Event Tlv object instance configured under *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
setOrgSpecEventTlv	Sets the instance of the Organization Specific Event Tlv object to the *linkOamLink* object. - Return value: 0 (success) - ErrorValue: non-zero value failure
addOrgSpecInfoTlv	Used to add a *linkOamOrgInfoTlv* object to the *linkOamLink* object. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value failure
delOrgSpecInfoTlv	Used to remove a particular *linkOamOrgInfoTlv* from the *linkOamLink* object. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value failure
getOrgSpecInfoTlv	Used to get a particular instance of *linkOamOrgInfoTlv* from the *linkOamLink* object. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value failure
setOrgSpecInfoTlv	Used for setting a particular instance of the *linkOamOrgInfoTlv* to the *linkOamLink*. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value failure
getFirstOrgSpecInfo Tlv	Used to get the first instance from the list of *linkOamOrgInfoTlv* configured under *linkOamLink*.

Member	Usage
	• Return value: 0 (success) • ErrorValue: non-zero value failure
getNextOrgSpecInfo Tlv	Used to get the next instance from the list of *linkOamOrgInfoTlv* configured under *linkOamLink*. This should be called after a call to the *getFirstOrgSpecInfoTlv*. • Return value: 0 (success) • ErrorValue: non-zero value failure
clearAllOrgSpecInfo Tlvs	Deletes all the *linkOamOrgInfoTlv* configured under *linkOamLink*. • Return value: 0 (success) • ErrorValue: non-zero value failure
showOrgSpecInfoTlvNames	Used to display the names of the all the *linkOamOrgInfoTlvs* configured under the *linkOamLink* object.
addVariableResponseDbContainer	Used to add a *linkOamVarContainer* object to the *linkOamLink* object. • Arguments: containerName (string) • Return value: 0 (success) • ErrorValue: non-zero value failure
delVariableResponseDbContainer	Used to remove a particular *linkOamVarContainer* from the *linkOamLink* object. • Arguments: containerName (string) • Return value: 0 (success) • ErrorValue: non-zero value (failure)
getVariableResponseDbContainer	Used to get a particular instance of *linkOamVarContainer* from the *linkOamLink* object. • Arguments: containerName (string) • Return value: 0 (success) • ErrorValue: non-zero value (failure)
setVariableResponseDbContainer	Used for setting a particular instance of the *linkOamVarContainer* to the *linkOamLink*. • Arguments: containerName (string) • Return value: 0 (success) • ErrorValue: non-zero value (failure)
getFirstVariable ResponseDb Container	Used to get the first instance from the list of *linkOamVarContainer* configured under *linkOamLink*. • Return value: 0 (success) • ErrorValue: non-zero value (failure)
getNextVariable ResponseDb Container	Used to get the next instance from the list of *linkOamVarContainer* configured under *linkOamLink*. This should be called after a call to the *getFirstVari-*

Member	Usage
	ableResponseDbContainer. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
clearAllVariable ResponseDb Container	Deletes all the *linkOamVarContainer* configured under linkOamLink. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
showVariable ResponseDbContainerNames	Used to display the names of the all the *linkOamVarContainer* configured under the *linkOamLink* object.
addOrgSpecTlv	Used to add a *linkOamOrgTlv* object to the *linkOamLink* object. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure) **Note**: Only one instance of this tlv should be configured under the linkOamLink object.
delOrgSpecTlv	Used to remove a particular linkOamOrgTlv from the linkOamLink object. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getOrgSpecTlv	Used to get a particular instance of *linkOamOrgTlv* from the *linkOamLink* object. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure)
setOrgSpecTlv	Used for setting a particular instance of the *linkOamOrgTlv* to the *linkOamLink*. - Arguments: tlvName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure)
clearAllOrgSpecTlvs	Deletes all the *linkOamOrgTlv* configured under *linkOamLink*. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
showOrgSpecTlv Names	Used to display the names of the all the *linkOamOrgTlv* configured under the *linkOamLink* object.
addVarDescriptor	Used to add a *linkOamVarDescriptor* object to the *linkOamLink* object.

Member	Usage
	- Arguments: descriptorName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure)
delVarDescriptor	Used to remove a particular *linkOamVarDescriptor* from the *linkOamLink* object. - Arguments: descriptorName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getVarDescriptor	Used to get a particular instance of *linkOamVarDescriptor* from the *linkOamLink* object. - Arguments: descriptorName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure)
setVarDescriptor	Used for setting a particular instance of the *linkOamVarDescriptor* to the *linkOamLink*. - Arguments: descriptorName (string) - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getFirstVarDescriptor	Used to get the first instance from the list of *linkOamVarDescriptor* configured under *linkOamLink*. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getNextVarDescriptor	Used to get the next instance from the list of *linkOamVarDescriptor* configured under *linkOamLink*. This should be called after a call to the *getFirstVarDescriptor*. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
clearAllVarDescriptors	Deletes all the *linkOamVarDescriptor* configured under linkOamLink. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
showVarDescriptor Names	Displays the names of the all the *linkOamVarDescriptor* configured under the *linkOamLink* object.
requestDiscLearned Info	Requests the Per-Link Discovered Learned Info. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
sendLoopback	Sends a Loopback request on a per-link basis. - Return value: 0 (success)

Member	Usage
	- ErrorValue: non-zero value (failure)
getDiscLearnedInfo	Retrieves the Discovered Learned Info after *requestDiscLearnedInfo* is being called. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
requestEventNotifn LearnedInfo	Requests the Per-Link Event Notification Learned Info. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getEventNotifn LearnedInfo	Retrieve the Event Notification Learned Info after *requestEventNotifnLearnedInfo* is being called. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
requestVariable ResponseLearnedInfo	Requests the Per-Link Variable Response Learned Info. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getVariableResponseLearnedInfoList	Retrieves the list of *linkOamVarRequestLearnedInfo* after *requestVariableResponseLearnedInfo* has been called. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getFirstVariable ResponseLearnedInfo	Retrieves the first *linkOamVarRequestLearnedInfo* from the list of linkOamVarRequestLearnedInfo obtained by the *getVariableResponseLearnedInfoList* call. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
getNextVariable ResponseLearnedInfo	Retrieves the next *linkOamVarRequestLearnedInfo* from the list of *linkOamVarRequestLearnedInfo*. This should be called after *getFirstVariableResponseLearnedInfo*. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
sendUpdated Parameters	Updates any change to the On-the-fly parameters to the daemon. This is an On-the-Fly trigger api. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
startEventPDU Transmission	Starts transmission of the OAM Event PDU. This is one-time or continous, depending whether *linkEventTxMode* is single or periodic. This is an On-the-Fly trigger api.

Member	Usage
	- Return value: 0 (success) - ErrorValue: non-zero value (failure)
stopEventPDU Transmission	Stops transmission of the OAM Event PDU. This is in effect if *linkEventTxMode* is periodic. This is an On-the-Fly trigger api. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
sendOrgSpecificPDU	Sends an Organization Specific PDU. This is an On-the-Fly trigger api. - Return value: 0 (success) - ErrorValue: non-zero value (failure)
restartDiscovery	Restarts discovery mechanism of the OAM protocol. This is an On-the-Fly trigger api. - Return value: 0 (success) - ErrorValue: non-zero value (failure)

linkOamInterface

Refer to NAME - linkOamInterface for a full description of this command. The *linkOamInterface* holds the information related to a single interface on the simulated link. The important options and subcommands of this command are:

linkOamInterface Options

Member	Usage
enabled	Enables the protocol interface.
interfaceId	(read ony) Specifies the interface id.
protocolInterfaceDescription	Specifies the pre-configured protocol interfaces.

linkOamSymTlv

Refer to NAME - linkOamSymTlv for a full description of this command. The *linkOamSymTlv* command configures the error symbol period event tlv in the event notification packet.The important options and subcommands of this command are:

linkOamSymTlv Options

Member	Usage
enabled	If true, includes the error symbol period event tlv in the event notification packet. Default is False.
errSymbWindow	Indicates the number of symbols in the period, encoded as a 16-bit unsigned integer. Default: 10 Min: 1 Max: 65535
errSymb	Indicates the number of errored symbols in the period. This is

Member	Usage
Threshold	required to be equal to or greater than, in order for the event to be generated. It is encoded as a 16-bit unsigned integer. Default: 1 Min: 0 Max: 65535
errSymbols	Indicates the number of symbol errors in the period, encoded as a 32-bit unsigned integer. Default: 0 Min: 0 Max: 4 octets

linkOamFrameTlv

Refer to NAME - linkOamFrameTlv for a full description of this command. The *linkOamFrameTlv* command configures the errored frame event tlv in the event notification packet. The important options and subcommands of this command are:

linkOamFrameTlv Options

Member	Usage
enabled	If true, it determines whether to include this tlv in the event notification packet. Default is False.
errFrame Threshold	Indicates whether the number of detected errored frames in the period is required to be equal to or greater than in order for the event to be generated. It is encoded as a 16-bit unsigned integer. Default: 1 Min: 0 Max: 65535
errFrames	Indicates the number of detected errored frames in the period, encoded as a 16-bit unsigned integer. Default: 1 Min: 0 Max: 4 octets
errFrameWindow	Indicates the duration of the period in terms of 100 ms intervals, encoded as a 16-bit unsigned integer. Default:1 second (10ms) Min: 1 second (10ms) Max: 1 minute (600 ms)

linkOamPeriodTlv

Refer to NAME - linkOamPeriodTlv for a full description of this command. The *linkOamPeriodTlv* command configures the errored frame period event tlv in the event notification packet. The important options and subcommands of this command are:

TABLE 3-164. linkOamPeriodTlv Options

Member	Usage
enabled	If true, it determines whether to include this tlv in the event notification packet. Default is False.
errFrameWindow	Indicates the duration of period in terms of frames, encoded as a 16-bit unsigned integer. Default: 10 second (10ms in terms of 100ms) Min: 1 Max: 65535
errFrameT hreshold	Indicates whether the number of errored frames in the period is required to be equal to or greater than in order for the event to be generated. It is encoded as a 16-bit unsigned integer. Default: 1 Min: 0 Max: 65535
errFrames	Indicates the number of frame errors in the period, encoded as a 16-bit unsigned integer. Default: 1 Min: 0 Max: 65535

linkOamSSTlv

Refer to NAME - linkOamSSTlv for a full description of this command. The *linkOamSSTlv* command configures the errored frame seconds summary event tlv in the event notification packet. The important options and subcommands of this command are:

linkOamSSTlv Options

Member	Usage
enabled	If true, it determines whether to include this tlv in the event notification packet. Default is False.
errFrameSecSumWindow	Indicates the duration of the period in terms of 100 ms intervals, encoded as a 16-bit unsigned integer. Default: 60 seconds (600 ms in terms of 100 ms) Min: 10 seconds (100 ms in terms of 100 ms)

Member	Usage
	Max: 900 seconds (9000 ms in terms of 100 ms)
errFrameSecSumThreshold	Indicates whether the number of errored frame seconds in the period is required to be equal to or greater than, in order for the event to be generated. It is encoded as a 16-bit unsigned integer. Default: one errored second Min: zero errored second Max: two octets errored seconds
errFrameSecSum	Indicates the number of errored frame seconds in the period, encoded as a 16-bit unsigned integer. Default: 1 Min: 0 Max: 65535

linkOamOrgEventTlv

Refer to NAME - linkOamOrgEventTlv for a full description of this command. The *linkOamOrgEventTlv* command configures the organization specific event tlv in the event notification packet. The important options and subcommands of this command are:

linkOamOrgEventTlv Options

Member	Usage
enabled	If true, it determines whether to include this tlv in the event notification packet. Default is False.
oui	This three-octet field contains a 24-bit Organizationally Unique Identifier (OUI). Default: 00 01 00. **Note**: Any three octets hex value may be given.
value	Indicates the value of the Organization Specific Event. **Note**: This has an unspecified field length. Any hex value may be given.

linkOamOrgInfoTlv

Refer to NAME - linkOamOrgInfoTlv for a full description of this command. The *linkOamOrgInfoTlv* command configures the organization specific information tlv in the information OAMPDU. The important options and subcommands of this command are:

linkOamOrgInfoTlv Options

Member	Usage
enabled	If true, determines whether to include this tlv in the information OAMPDU.
oui	This is a three octets field in hex, which contains the 24 bit Organizationally Unique Identifier (OUI).

Member	Usage
	Default: 00 01 00.
value	Indicates the value of Organization Specific Information (OUI) TLV in hex. The length of this field is unspecified.

linkOamVar Container

Refer to NAME - linkOamVarContainer for a full description of this command. The *linkOamVarContainer* command configures the variable response container. The important options and subcommands of this command are:

linkOamVarContainer Options

Member	Usage
enabled	If true, enables the container. Default: false
variableBranch	Indicates the variable branch value in hex. Default: 00
variableWidth	Indicates the length of the variable value field in octets. Default: 00
variableIndication	If true, this indicates that the Leaf has some error. Default: False
variableLeaf	Indicates the variable leaf value in hex. Default: 00
variableValue	The Variable Value may be 1 to 128 octets in length. Its width is determined by the Variable Width field. Default: 00

linkOamOrgTlv

Refer to NAME - linkOamOrgTlv for a full description of this command. The *linkOamOrgTlv* command configures the organization specific OAMPDU. The important options and subcommands of this command are:

linkOamOrgTlv Options

Member	Usage
enabled	If true, determines whether to include this tlv in the organization specific OAMPDU.
oui	This is a three octets field in hex, which contains the 24 bit Organizationally Unique Identifier (OUI). Default: 00-01-00
value	Indicates the value of Organization Specific Information TLV in hex. The length of this field is unspecified.

linkOamDiscLearnedInfo

Refer to NAME - linkOamDiscLearnedInfo for a full description of this command. The *linkOamDiscLearnedInfo* command fetches and describes the discovered learned data. The important subcommands and options of this command are:

linkOamDiscLearnedInfo Options

Member	Usage
remoteMac Address	Indicates the Mac address of the remote DTE for the link.
remoteStable	Indicates the stability status of the remote DTE. It is displayed as either 0 or 1.
remoteEval	Indicate whether remote DTE is in the discovery process or not. It is displayed as either 0 or 1.
remoteCritical Event	Indicates whether any critical event has been received from the remote DTE. It is displayed as either 0 or 1.
remoteDying Gasp	Indicates whether any unrecoverable failure has occurred on the remote DTE. It is displayed as either 0 or 1.
remoteLinkFault	Indicates whether receive path has detected error on remote DTE. It is displayed as either 0 or 1.
remoteOam Version	Indicates the OAM version supported by the remote DTE.
remoteMuxAction	Indicates the state of multiplexer of remote DTE. One of: • fwd (value 0): Remote DTE is forwarding non-OAMPDUs to the lower sub layer • discard (value 1): Remote DTE is discarding non-OAMPDUs
remoteMode	Indicates the configuration mode for the remote DTE. One of: • Active (value 0): Remote DTE is in active mode • Passive (value 1): Remote DTE is in passive mode
remoteParser Action	Indicates the state of parser of remote DTE. One of: • fwd (value 0): Remote DTE is forwarding non-OAMPDUs to higher layer • lb (value 1): Remote DTE is looping back the non-OAMPDUs • discard (value 2): Remote DTE is discarding non-OAMPDUs
remoteRevision	Indicates the current revision of the information tlv of remote DTE
remoteMaxPdu Size	Indicates the maximum pdu size supported by the remote DTE.
remoteVariable Retrieval	Indicates whether remote DTE supports responding to variable request. It is displayed as either 0 or 1.

Member	Usage
remoteLinkEvent	Indicates whether remote DTE supports interpreting link events. It is displayed as either 0 or 1.
remoteLoopbackSupport	Indicates whether remote DTE is capable of remote loopback mode. It is displayed as either 0 or 1.
remote Unidirectional Support	Indicates whether remote DTE is capable of sending OAMPDUs when the receive path is non operational. It is displayed as either 0 or 1.
remoteOui	Specifies the remote OUI value.
remoteVendor SpecInfo	Indicates the remote vendor specific information.
localDiscStatus	Indicates the status of the discovery process. One of: • fault • activeSendLocal • passiveWait • sendLocalRemote • sendLocalRemoteOk • sendAny
localStable	Indicates the stability status of local DTE. It is displayed as either TRUE or FALSE.
localEval	Indicates whether the local DTE is in the discovery process or not. It is displayed as either TRUE or FALSE.
localRevision	Indicates the current revision of the information tlv of local DTE.
localMuxAction	Indicates the state of multiplexer of local DTE. One of: • fwd (value 0): Local DTE is forwarding non-OAMPDUs • discard (value 1): Local DTE is discarding non-OAMPDUs
localParserAction	Indicates the state of parser of the local DTE. One of: • fwd (value 0): Local DTE is forwarding non-OAMPDUs to higher layer • lb (value 1): Local DTE is looping back the non-OAMPDUs • discard (value 2): Local DTE is discarding non-OAMPDUs
remoteHeader Refreshed	Boolean to check whether remote information is available or not.
remoteTlv Refreshed	Boolean to check whether remote tlv is available or not.

linkOamEventNotifnInfo

Refer to NAME - linkOamEventNotifnInfo for a full description of this command. The *linkOamDiscEventNotifnInfo* command fetches and describes the learned data for event notification. The important subcommands and options of this command are:

linkOamSymTlv Options

Member	Usage
remoteSymbol PeriodWindow	The number of symbols in the period configured in the remote DTE.
remoteSymbol PeriodThreshold	The number of errored symbols configured in the remote DTE to generate this event.
remoteSymbol PeriodErrors	The number of errored symbols in the period received in the last received event.
remoteSymbol PeriodError RunningTotal	The total number of Errored Symbol Period Symbols Error received from the emulation start time.
remoteSymbol PeriodEvent RunningTotal	The total number of Errored Symbol Period Event TLVs received from the emulation start time.
remoteFrame Window	The duration of period in terms of 100 ms intervals configured in the remote DTE.
remoteFrame Threshold	The number of errored frames configured in the remote DTE to generate this event.
remoteFrame Error	The number of errored frames in the period received in the last received event.
remoteFrame ErrorRunning Total	The total number of Errored Frame Error received from the emulation start time.
remoteFrame EventRunning Total	The total number of Errored Frame Event TLVs received from the emulation start time.
remoteFrame PeriodWindow	The duration of period in terms of frames configured in the remote DTE.
remoteFrame PeriodThreshold	The number of errored frames configured in the remote DTE, to generate this event.
remoteFrame PeriodError	The number of errored frames in the period received in the last received event.
remoteFrame PeriodError RunningTotal	The total number of Errored Frame Period Frame Error received from the emulation start time.
remoteFrame PeriodEvent RunningTotal	The total number of Errored Frame Period Event TLVs received from the emulation start time.
remoteFrameSecSumWindow	The duration of period in terms of 100 ms configured in the remote DTE.
remoteFrameSecSum Threshold	The number of errored frames seconds configured in the remote DTE, to generate this event.
remoteFrameSecSumError	The number of errored frames second in the period received in the last received event.
remoteFrameSecSumError	The Total number of Errored Frame SS Error received

Member	Usage
Running Total	from the emulation start time.
remoteFrameSecSumEvent Running Total	The total number of Errored Frame SS Event TLVs received from the emulation start time.
localSymbolPeriodErrorRunning Total	The total number of Errored Symbol Period Symbols Error sent from the local DTE, from the emulation start time.
localSymbolPeriodEventRunning Total	The total number of Errored Symbol Period Event TLVs sent from local DTE, from the emulation start time.
localFrameError RunningTotal	The total number of Errored Frame Error sent from local DTE, from the emulation start time.
localFrameEventRunningTotal	The total number of Errored Frame Event TLVs sent from local DTE, from the emulation start time.
localFramePeriodErrorRunning Total	The total number of Errored Frame Period Frame Error sent from local DTE, from the emulation start time.
localFramePeriodEventRunning Total	The total number of Errored Frame Period Event TLVs sent from local DTE, from the emulation start time.
localFrameSec SumError Running Total	The total number of Errored Frame Sec Sum Error sent from local DTE, from the emulation start time.
localFrameSec Sum EventRunning Total	The total number of Errored Frame Sec Sum Event TLVs sent from local DTE, from the emulation start time.

linkOamVarRequest LearnedInfo

Refer to NAME - linkOamVarRequestLearnedInfo for a full description of this command. The *linkOamVarRequestLearnedInfo* command fetches and describes the learned data for variable request. The important subcommands and options of this command are:

linkOamSymTlv Options

Member	Usage
variableBranch	Contains the value of the requesting branch.
variableWidth	Indicates the length of the variable value. An encoding of 0x00 equals to 128 octets.
variableIndication	Indicates the status of the retrieved variable container.
variableLeaf	Contains the value of the requesting leaf value.
variableValue	Indicates the variable value.

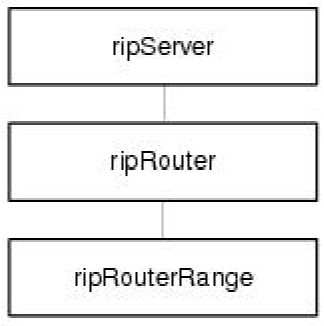
RIP

Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for an overview of the Ixia RIP test model. The RIP-related commands are:

- ripServer — provides access to the RIP part of a port's protocol server.
- ripInterfaceRouter — a container used to hold the list of route ranges associated with the simulated router.
- ripRouteRange — a set of routes to be included in *ripInterfaceRouter.*

These commands and the data that they maintain are arranged in an hierarchy as shown in the following figure.

RIP Command Hierarchy



ripServer

Refer to NAME - ripServer for a full description of this command. The *ripServer* command is necessary in order to access the RIP component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
ripServer select 1 5 2
```

will access the RIP server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routers occurs using the *ripInterfaceRouter* command and subsidiary commands. The important subcommands of this command are shown in the following table.

ripServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouter	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the *ripInterfaceRouter* command.
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. *getFirstRouter* should be called before *getNextRouter.* The data appears in the *ripInterfaceRouter* command.

Member	Usage
delRouter	Deletes a particular router from the list.
setRouter	This command allows you to change a router's configuration on the fly while the RIP protocol is running. Changes to the router's configuration are made with the *ripInterfaceRouter* command. This call must be followed by a call to *ripServer write.*

ripInterfaceRouter

Refer to NAME - ripInterfaceRouter for a full description of this command. The *ripInterfaceRouter* command represents a simulated router. In addition to some identifying options, it holds a list for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *ripRouteRange* command.

The important options and subcommands of this command are shown in the following table.

ripInterfaceRouter Options

Category	Member	Usage
Router	enableRouter	Enables or disables the simulated router.
	protocolInterface Description	The name of the defined *interfaceEntry* from which IP address and mask are extracted for this interface.
Operating mode	responseMode	The current implementation uses split horizon as its update mode. Split horizon is a method for omitting routes learned from a neighbor in update messages to that same neighbor. Additional options will be offered in future releases.
	sendType	The method for sending RIP packets. One of: • Multicast — sends Version 2 packets via multicast. • Broadcast Ver1 — sends V1 packets via broadcast. • Broadcast Ver2 — sends V2 packets via broadcast.
	receiveType	Filters the version of messages this router will receive. One of: • Version1 — RIP Version 1 messages only.
		• Version2 — RIP Version 2 messages only. • Version1+2 — Both RIP version messages.
	updateInterval	The time, in seconds, between transmitted update messages.
	updateIntervalOffset	A random percentage of this time value, expressed in seconds, which will be added or subtracted from the update interval.
Authorization	enableAuthorization	Indicates whether authorization is included in update messages.

Category	Member	Usage
	authorizationPassword	If *enableAuthorization* is set, this is the 16-character password to be used. Only simple password authentication is supported.

ripInterfaceRouter Subcommands

Category	Member	Usage
Route Ranges	clearAllRouteRange	Clears all route ranges.
	addRouteRange	Adds a new route range. The route range must have been previously configured through the use of the *ripRouteRange* command.
	getRouteRange getFirstRouteRange getNextRouteRange	Accesses a particular route range either by ID, or by iterating through all of the route ranges. *getFirstRouteRange* should be called before *getNextRouteRange*. The data appears in the *ripRouteRange* command.
	delRouteRange	Deletes a particular route range.
	setRouteRange	This command may be used to change an individual route range on the fly while the RIP protocol server is running. Changes to the route range are made with the *ripRouteRange* command. This command must be followed with a call to *ripServer write.*

ripRouteRange

Refer to NAME - ripInterfaceRouter for a full description of this command. The *ripRouteRange* command describes an individual set of routes. Route ranges are added into *ripInterfaceRouter* lists using the *ripInterfaceRouter addRouteRange* command. The important options of this command are:

ripRouteRange Options

Member	Usage
enableRouteRange	Enables the use of this route range for the simulated router.
routeTag	An arbitrary value associated with the routes in this range. It is used to provide a means for distinguishing internal versus external RIP routes.
networkIpAddress	The network address to be used in creating this route range.
networkMaskWidth	The 32-bit network mask to be applied to the *networkIpAddress* to yield the non-host part of the address. A value of 0 means there is no subnet address.
numberOfNetworks	The number of networks to be generated for this route range, based on the network address plus the network mask.
nextHop	The immediate next hop IP address on the way to the destination address.
metric	The total metric cost for these routes. The valid range is from 1 to 16 (inclusive). A value of 16 means that the destination is not reachable, and that route will be removed from service.

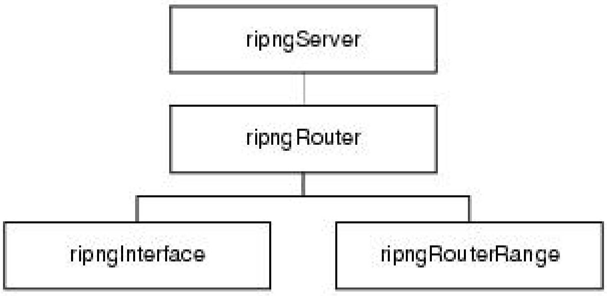
RIPng

Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for an overview of the Ixia RIP test model. RIPng routers are used to simulate IPv6 routers using the RIPng protocol as defined in RFC 2080. The RIPng-related commands are:

- ripngServer — provides access to the RIPng part of a port's protocol server.
- ripngRouter — represents a simulated RIPng router which holds a list of route ranges and interfaces associated with the simulated router
- ripngInterface — a set of interfaces to be included in ripngRouter.
- ripngRouteRange — a set of routes to be included in ripngRouter.

These commands and the data that they maintain are arranged in an hierarchy as shown in the following figure.

RIPng Command Hierarchy



ripngServer

Refer to NAME - ripngServer for a full description of this command. This command is necessary in order to access the RIPng component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
ripngServer select 1 5 2
```

will access the RIPng server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routers occurs using the ripngRouter and subsidiary commands. The important options and subcommands of this command are shown in the following table.

ripngServer Options

Member	Usage
numRoutes timePeriod	Tuning parameters that allow the number of routes generated to be throttled.

ripngServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouter	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the ripngRouter command.

Member	Usage
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. *getFirstRouter* should be called before *getNextRouter.* The data appears in the ripngRouter command.
delRouter	Deletes a particular router from the list.
setRouter	This command allows you to change a router's configuration on the fly while the RIPng protocol is running. Changes to the router's configuration are made with the ripngRouter command. This call must be followed by a call to ripngServer *write.*

ripngRouter

Refer to NAME - ripngServer for a full description of this command. The *ripngRouter* command represents a simulated router. In addition to some identifying options, it holds two lists for the router:

- Interfaces — physical interfaces to the router over which updates are sent, constructed in the ripngInterface command.
- Route ranges — routes to be advertised by the simulated router, constructed in the ripngRouteRange command.

The important options and subcommands of this command are shown in the following tables.

ripngRouter Options

Member	Usage
enables	Enables or disables each router.
routerId	The ID associated with the simulated router.
receiveType	Indicates whether received routes will be stored or ignored.
enableInterfaceMetric	Allows multiple interfaces to transmit distinct updates based on the same routing table.
updateInterval	The time, in seconds, between transmitted update messages.
updateIntervalOffset	A random percentage of this time value, expressed in seconds, which will be added or subtracted from the update interval.

ripInterfaceRouter Subcommands

Category	Member	Usage
Interfaces	clearAllInterfaces	Clears all interfaces.
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the ripngInterface command.
	getInterface getFirstInterface getNextInterface	Accesses a particular interface either by ID, or by iterating through all of the interfaces. *getFirstInterface* should be called before *getNextInterface*. The data appears in the ripngInterface command.
	delInterface	Deletes a particular interface.
	setInterface	This command may be used to change an individual inter-

Category	Member	Usage
		face on the fly while the RIPng protocol server is running. Changes to the interface are made with the ripngInterface command. This command must be followed with a call to ripngServer *write.*
Route Ranges	clearAllRouteRanges	Clears all route ranges.
	addRouteRange	Adds a new route range. The route range must have been previously configured through the use of the ripngRouteRange command.
	getRouteRange getFirstRouteRange getNextRouteRange	Accesses a particular route range either by ID, or by iterating through all of the route ranges. *getFirstRouteRange* should be called before *getNextRouteRange*. The data appears in the ripngRouteRange command.
	delRouteRange	Deletes a particular route range.
	setRouteRange	This command may be used to change an individual route range on the fly while the RIPng protocol server is running. Changes to the route range are made with the ripngRouteRange command. This command must be followed with a call to ripngServer *write.*

ripngInterface

Refer to NAME - ripngInterface for a full description of this command. The *ripngInterface* command represents an interface on a simulated RIPng router. A RIPng interface uses an interface defined with the *interfaceEntry* command. The important options of this command are shown in the following table.

ripngInterface Options

Member	Usage
enable	Enables or disables the use of the interface.
protocolInterface Description	Matches the name of an interface entry defined with *interfaceEntry* which defines the characteristics of the interface.
responseMode	Controls the manner in which received routes are repeated back to their source. The modes are split horizon, no split horizon, and split horizon with poison reverse.
interfaceMetric	Used in conjunction with the *enableInterfaceMetric*in the ripngRouter command to differentiate routing tables generated by multiple interfaces.

ripngRouteRange

Refer to NAME - ripngRouteRange for a full description of this command. The *ripngRouteRange* command describes an individual set of routes. Route ranges are added into ripngRouter lists using the ripngRouter *addRouteRange* command. The important options of this command are:

ripngRouteRange Options

Member	Usage
enable	Enables the use of this route range.

Member	Usage
routeTag	A arbitrary value associated with the routes in this range. It is used to provide a means for distinguishing internal versus external RIPng routes.
networkIpAddress	The network address to be used for this route range.
maskWidth	The network mask to be applied to the *networkIpAddress* to yield the non-host part of the address. A value of 0 means there is no subnet address.
nextHop	The immediate next hop IP address on the way to the destination address.
metric	The total metric cost for these routes. The valid range is from 1 to 16 (inclusive). A value of 16 means that the destination is not reachable, and that route will be removed from service.
numRoutes step	The number of routes to be generated for this range based on the IP address/network mask. For each route generated, the network number is incremented by *step.*
enableInclude Loopback enableInclude Multicast	Indicates whether to include loopback and multicast addresses when generating route ranges.

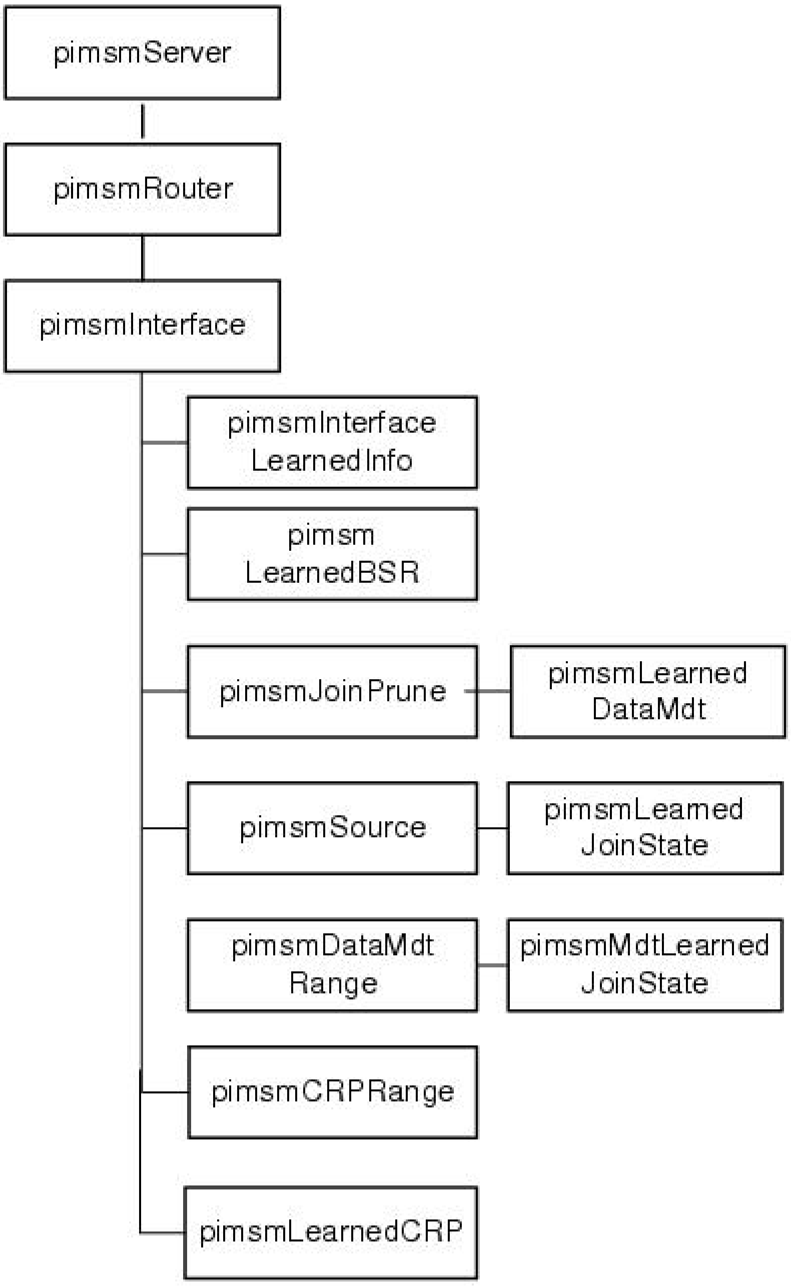
PIM-SM

Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for an overview of the Ixia PIM-SM test model. PIM-SM routers are used to simulate PIM-SM routers using the PIM-SM protocol as defined in *RFC 2362* and *draft-ietf-pim-sm-v2-new-05*. The PIM-SM-related commands are:

- pimsmServer — provides access to the PIM-SM part of a port's protocol server.
- pimsmRouter — represents a simulated PIM-SM router which holds a list of interfaces associated with the simulated router.
- pimsmInterface — a set of interfaces to be included in pimsmRouter.
- pimsmInterfaceLearnedInfo — learned information associated with a pimsmInterface.
- pimsmJoinPrune — a set of multicast addresses to be included in pimsmInterface.
- pimsmSource — a set of register addresses to be included in pimsmInterface.
- pimsmLearnedJoinState — learned join state information associated with a pimsmSource.
- pimsmDataMdtRange — a set of Data MDT Ranges to be included in pimsmInterface.
- pimsmMdtLearnedJoinState — learned join state information associated with a pimsmDataMdtRange.
- pimsmCrpRange — a set of Candidate Rendezvous Points to be included in pimsmInterface.
- pimsmLearnedBSRInfo — learned bootstrap information associated with a pimsmInterface.
- pimsmLearnedCRPInfo — learned crp information associated with a pimsmInterface.

These commands and the data that they maintain are arranged in an hierarchy as shown in the following figure.

PIM-SM Command Hierarchy



pimsmServer

Refer to NAME - pimsmServer for a full description of this command. This command is necessary in order to access the PIM-SM component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
pimsmServer select 1 5 2
```

will access the PIM-SM server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routers occurs using the pimsmRouter and subsidiary commands. The important options and sub-commands of this command are:

pimsmServer Options

Member	Usage
enableRateControl interval bsmFramePerInterval crpFramePerInterval sourceMessagesPerInterval joinPrunMessagesPerInterval registerStopMessagesPerInterval	Allows the rate of transmitted PIM-SM messages to be controlled.

pimsmServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouter	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the pimsmRouter command.
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. *getFirstRouter* should be called before *getNextRouter.* The data appears in the pimsmRouter command.
delRouter	Deletes a particular router from the list.
setRouter	This command allows you to change a router's configuration on the fly while the PIM-SM protocol is running. Changes to the router's configuration are made with the pimsmRouter command. This call must be followed by a call to pimsmServer *write.*
denyGrePimIp Prefix	Ixia will reject all GRE-PIM packets whose outer source IP address falls within this specified network prefix.

pimsmRouter

Refer to NAME - pimsmRouter for a full description of this command. The *pimsmRouter* command represents a simulated PIM-SM router. In addition to some identifying options, it holds a list for the router:

- Interfaces — physical interfaces to the router over which PIM-SM messages are sent, constructed in the pimsmInterface command.

The important options and subcommands of this command are:

pimsmRouter Options

Member	Usage
enable	Enables or disables the router's simulation.
routerId	The ID of the router, in IPv4 format.
drPriority	The designated router (DR) priority, used for DR election.
joinPruneHoldTime	The amount of time that neighbor routers should hold a received join state.
joinPruneInterval	The interval between transmitted join/prune messages.
dataTimeOut	The Data MDT hold time, in seconds. If a PE router connected to a receiver does not receive a data MDT join TLV message within this time period, it will leave the data MDT group. *(default = 180)*

Member	Usage
dataMdtInterval	The time interval, in seconds, between transmissions of data MDT join TLV messages by the source PE router. *(default = 60)*

pimsmRouter Subcommands

Category	Member	Usage
Interfaces	clearAllInterfaces	Clears all interfaces.
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the pimsmInterface command.
	getInterface getFirstInterface getNextInterface	Accesses a particular interface either by ID, or by iterating through all of the interfaces. *getFirstInterface* should be called before *getNextInterface*. The data appears in the pimsmInterface command.
	delInterface	Deletes a particular interface.
	setInterface	This command may be used to change an individual interface on the fly while the PIM-SM protocol server is running. Changes to the interface are made with the pimsmInterface command. This command must be followed with a call to pimsmServer *write.*

pimsmInterface

Refer to NAME - pimsmInterface for a full description of this command. The *pimsmInterface* command represents an interface on a simulated PIM-SM router. A PIM-SM interface uses an interface defined with the *interfaceEntry* command. The important subcommands and options of this command are:

pimsmInterface Subcommands

Category	Member	Usage
	setDefault	Sets the options to default values.
Joins Prunes	clearAllJoins Prunes	Clears all joins/prunes.
	addJoinPrune	Adds a new join/prune. The join/prune must have been previously configured through the use of the pimsmJoinPrune command.
	getJoinPrune getFirstJoinPrune getNextJoinPrune	Accesses a particular join/prune either by ID, or by iterating through the join/prunes. The data appears in the pimsmJoinPrune command.
	delJoinPrune	Deletes a particular join/prune.
	setJoinPrune	This command may be used to change an individual join/prune on the fly while the PIM-SM protocol server is running. Changes to the join/prune are made with the pimsmJoinPrune command. This command must be followed with a call to pimsmServer *write.*
Sources	clearAllSources	Clears all sources.
	addSource	Adds a new source. The source must have been previously configured through the use of the pimsmSource

Category	Member	Usage
		command.
	getSource getFirstSource getNextSource	Accesses a particular source either by ID, or by iterating through all of the source. *getFirstSource* should be called before *getNextSource*. The data appears in the pimsmSource command.
	delSource	Deletes a particular source.
	setSource	This command may be used to change an individual source on the fly while the PIM-SM protocol server is running. Changes to the sources are made with the pimsmSource command. This command must be followed with a call to pimsmServer *write.*
Data MDT Ranges	clearAllDataMdtRanges	Clears all data MDT ranges.
	addDataMdtRange	Adds a new data MDT range. The data MDT range must have been previously configured through the use of the pimsmDataMdtRange command.
	getDataMdtRange getFirstDataMdtRange getNextDataMdtRange	Accesses a particular data MDT range either by ID, or by iterating through all of the interfaces. *getFirstDataMdtRange* should be called before *getNextDataMdtRange*. The data appears in the pimsmDataMdtRange command.
	delDataMdtRange	Deletes a particular data MDT range.
	setDataMdtRange	This command may be used to change an individual data MDT range on the fly while the PIM-SM protocol server is running. Changes to the data MDT ranges are made with the pimsmDataMdtRange command. This command must be followed with a call to pimsmServer *write.*
CRP	addCRPRange delCRPRange getCRPRange setCRPRange getFirstCRPRange getNextCRPRange clearAllCRPRanges	Accesses a particular CRP range either by ID, or by iterating through all of the interfaces.

pimsmInterface Options

Member	Usage
enable	Enables or disables the use of the interface.
protocolInterface Description	Matches the name of an interface entry defined with *interfaceEntry* which defines the characteristics of the interface.
helloHoldTime	The amount of time that neighbor routers should hold the interface as reachable.
helloInterval	The interval between transmitted Hello messages.
generationIdMode	The mode used for creating the 32-bit value for the generation ID. This can either be incrementing, random or constant. *(default = constant)*

Member	Usage
enableSendGenerationId	Enables the send generation ID option, and the generation ID mode field will become available to make a mode selection. *(default = enabled)*
enableSendBidirectional Option	Enables (sets) the header bi-directional PIM-SM flag bit (=1), per IETF DRAFT draft-ietf-pim-bidir-04. **NOTE**: Designated forwarder election is not currently supported. *(default = disabled)*
enablePruneDelay pruneDelay	The delay that an upstream router should apply awaiting joins following a prune message.
enablePruneDelayTBit	Sets (enables) the T flag bit in the LAN Prune Delay option of the Hello message (=1). Setting this bit specifies that the sending PIM-SM router has the ability to disable join message suppression. *(default = disabled)*
overrrideInterval	The delay interval, in milliseconds, for randomizing the transmission time for override messages, which are used when scheduling a delayed join message. This is part of the LAN prune delay option included in Hello messages. The valid range is 100 to 7FFF msec. *(default = 2500)*
ipType neighborIp	The type (IPv4 or IPv6) of the neighbor and the neighbor's address.
enableAutoPick Neighbor	Allows the neighbor IP address to be automatically selected from received Hello messages.
enableDiscardDataMdt	If enabled, interface learned info will be available. If disabled, data MDT learned info will be available. (*default= disabled*)
enableBfdRegistration	Indicates if a BFD session is to be created to the PIMSM peer IP address once the PIMSM session is established. This allows PIMSM to use BFD to maintain IPv4 connectivity the PIMSM peer.
useV4MappedV6Address	Indicates that PIMSM will use an IPv6 type address (which is the v4-mapped-v6 address on the GRE interface) as the source address instead of using the link-local address in the Hello packets.
enableBootstrap	If enabled, the PIM-SM interface participates in Bootstrap Router election procedure.
bootstrapPriority	Priority of the Bootstrap Router (BSR) that is set with the same name in all Bootstrap Messages sent by this BSR.
bootstrapHashMaskLen	Hash Mask Length of the Bootstrap Router (BSR) that is set with the same name in all Bootstrap Messages sent by this BSR.
bootstrapInterval	The time interval (in seconds) between two consecutive bootstrap messages sent by the BSR.
bootstrapTimeout	Amount of time (in seconds) of not receiving any Bootstrap Messages, after which the BSR if candidate at that point of time; will decide that the currently elected BSR has gone down and will restart BSR election procedure.

Member	Usage
forceSemanticFragmentation	If enabled, this forces the BSR to send only one group specific RP list per bootstrap message, even if there is space in the packet to push in more RP list information pertaining to a different group.
supportUnicastBootstrap	If enabled, this supports the sending and processing of Unicast bootstrap messages.
discardLearntRPInfo	If true, disregards group mappings learnt from Bootstrap Message (in case not acting as elected BSR) or from Candidate RP Advertisement (in case of elected BSR).
learnSelectedRPSet	If enabled, it controls whether all RP-to-group mappings are stored or the selected RP set consisting of one best RP per group is stored.

pimsmInterfaceLearnedInfo

Refer to NAME - pimsmInterfaceLearnedInfo for a full description of this command. The *pimsmInterfaceLearnedInfo* command fetches and describes the learned data MDT TLV information for the current interface defined in *pimsmInterface*. (This information will be visible only for a GRE interface.) The important subcommands and options of this command are:

pimsmInterfaceLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.

pimsmInterfaceLearnedInfo Options

Member	Usage
mdtGroupAddress	*(Read-only)* The learned MDT (PE) group address contained in this data MDT TLV.
mdtSourceAddress	*(Read-only)* The learned MDT (PE) source address contained in this data MDT TLV.
ceGroupAddress	*(Read-only)* The learned MDT CE group address contained in this data MDT TLV.
ceSourceAddress	*(Read-only)* The learned MDT CE source address contained in this data MDT TLV.
age	*(Read-only)* The amount of time remaining before this data MDT TLV times out, in seconds.

pimsmJoinPrune

Refer to NAME - pimsmJoinPrune for a full description of this command. The *pimsmJoinPrune* command describes a range of multicast addresses that PIM-SM routers will express an interest in on behalf of their local clients. Joins/prunes are added into pimsmRouter list using the pimsmRouter *addJoinPrune* command. The important subcommands and options of this command are:

pimsmJoinPrune Subcommands

Member	Usage
setDefault	Sets the options to default values.

Member	Usage
requestLearnedDataMdt getLearnedDataMdt	Requests and then fetches the learned data MDT information. The retrieved values are visible in the pimsmLearnedDataMdt command.

pimsmJoinPrune Options

Member	Usage
enable	Enables the use of this join/prune.
rangeType	The type of join/prune messages to generate. One of: • (*,*,RP) wildcard group • (*,G) group specific • (S,G) source and group specific • (*,G)-(S,G) switchover type. The *switchoverInterval* indicates the time to switch between the two states. • (S,G) based on register messages received in *registerStopTriggerCount*.
switchoverInterval	The time interval, in seconds, allowed for the switch from using the RP tree to using a source-specific tree. *(default = 5)*
rpAddress	The IP address of the rendezvous point (RP) router.
groupAddress groupMaskWidth groupAddressCount	Defines the multicast group range(s).
pruneSourceAddress pruneSourceMaskWidth pruneSourceAddressCount	Defines the prune source address(es).
enableFlap	Enables join/prune flapping.
flapInterval	Defines the join/prune flapping interval.
enablePacking	Enables packing of multiple groups into a single packet.
sourceGroupMapping	Controls the mapping from sources to groups during advertisement. One of: • pimsmMappingFullyMeshed — (default) All sources to all groups. • pimsmMappingOneToOne — One source to one group
sourceAddress sourceMaskWidth sourceAddressCount	Defines the source address(es) that generates multicast traffic. A source address must be a unicast address.
registerStopTriggerCount	The number of register messages that can be used to transmit an (S,G) in the last case of the *rangeType* setting.
enableDataMdtFlag	If enabled, *pimsmLearnedDataMdt* will be available. *(default = disabled)*

pimsmLearnedDataMdt

Refer to NAME - pimsmLearnedDataMdt for a full description of this command. The *pimsmInterfaceLearnedInfo* command fetches and describes the learned data MDT information for the current join/prune defined in pimsmJoinPrune. The important subcommands and options of this command are:

pimsmLearnedDataMdt Subcommands

Member	Usage
setDefault	Sets the options to default values.
getAllDataMdts getDataMdt	Gets the learned data MDT information associated with this data MDT join/prune. The retrieved values are visible in the NAME - pimsmLearnedDataMdt command.

pimsmLearnedDataMdt Options

Member	Usage
numSources	*(Read-only)* The number of sources associated with the data MDT join/prune.
numGroupsPerSource	(*Read-only*) The number of groups received per source address.

pimsmSource

Refer to NAME - pimsmSource for a full description of this command. The *pimsmSource* command describes a range of multicast addresses that PIM-SM routers will register with the DR. Sources are added into pimsmRouter list using the pimsmRouter *addSource* command. The important subcommands and options of this command are:

pimsmSource Subcommands

Member	Usage
setDefault	Sets the options to default values.
requestLearnedJoinState getLearnedJoinState	Requests and then fetches the learned join state from the simulated interface. The retrieved values are visible in the pimsmLearnedJoinState command.

pimsmSource Options

Member	Usage
enable	Enables the use of this source.
enableSendNullRegAtBeginning	Enables the transmission of an initial null registration at emulation startup.
txIterationGap	The gap between periodically transmitted register messages.
rpAddress	The IP address of the rendezvous point router.
groupAddress groupAddressCount	Defines the multicast group range.
sourceAddress sourceAddressCount	Defines the source address for the register messages.
enableDiscardJoin	Discards learned join messages.
udpDestinationPort udpSourcePort	The number of UDP destination ports in the receiving multicast group and the number of UDP source ports sending encapsulated UDP packets to multicast groups via register messages to the RP.

Member	Usage
sourceGroupMapping	Controls the mapping from sources to groups during advertisement. One of: • pimsmMappingFullyMeshed — *(default)* All sources to all groups. • pimsmMappingOneToOne — One source to one group
activationInterval	(for data MDTs only) The time period, in seconds, after which the sources will start sending packets to support the switchover from the default MDT to the data MDT. *(default = 60)*

pimsmLearnedJoinState

Refer to NAME - pimsmLearnedJoinState for a full description of this command. The *pimsmLearnedJoinState* command fetches and describes the learned join states for the current source defined in pimsmSource. The important subcommands and options of this command are:

pimsmLearnedJoinState Subcommands

Member	Usage
setDefault	Sets the options to default values.
receivedAllJoins	*(Read-only)* Indicates that the join messages that have been received includes all source and group addresses.
receivedAllJoinsForGroup	*(Read-only)* Indicates that the join messages that have been received for a specific group includes all source addresses.
receivedAllJoinsForSource	*(Read-only)* Indicates that the join messages that have been received for a specific source includes all group addresses.
receivedJoin	*(Read-only)* Indicates that the join messages that have been received includes a specific source and group address.

pimsmLearnedJoinState Options

Member	Usage
numGroupsPerSource	The number of groups received per source address.
numSources	The number of sources in the learned join state.

pimsmDataMdtRange

Refer to NAME - pimsmDataMdtRange for a full description of this command. The *pimsmDataMdtRange*command describes a range of data MDT group multicast addresses. Data MDT ranges are added into the pimsmInterface list using the pimsmInterface *addDataMdtRange* command. The important subcommands and options of this command are:

pimsmDataMdtRange Subcommands

Member	Usage
setDefault	Sets the options to default values.
requestMdtLearnedJoinState getMdtLearnedJoinState	Requests and then fetches the learned data MDT join state from the simulated interface. The retrieved values are visible in the *pimsmMdtLearnedJoinState* command.

pimsmDataMdtRange Options

Member	Usage
enable	Enables the use of this data MDT range. *(default = disable)*
groupAddress	The first IPv4 multicast group address in the range. *(default = 225.0.0.0)* **NOTE**: This must be a valid IPv4 multicast address.
groupAddressCount	Used with the groupAddress to define the range. *(default = 1)*
sourceAddress	The efirst IPv4 source address in the range. *(default = 0.0.0.1)* **NOTE**: This must be a valid IPv4 unicast (non-loopback) address.
sourceAddressCount	Used with the source address to define the range. *(default = 1)*
sourceGroupMapping	Controls the mapping from sources to groups during advertisement. One of: • pimsmMappingFullyMeshed — *(default)* All sources to all groups. • pimsmMappingOneToOne — One source to one group.
dataMdtAddress	The first IPv4 data MDT multicast address in the range. *(default = 230.0.0.0)* **NOTE**: This must be a valid multicast address.
dataMdtAddressCount	Used with the dataMdtAddress to define the data MDT range. *(default = 1)*
enablePacking	Enables the packing of multiple addresses into a single packet. *(default = enabled)*
activationInterval	The time interval for the switchover from the default MDT to the data MDT, in seconds. *(default = 60)*
enableDiscardLearnedStates	If disabled, *pimsmMdtLearnedJoinStates* will be available. If enabled, *pimsmLearnedJoinStates* will be available instead. (default = enabled)

pimsmMdtLearnedJoinState

Refer to *pimsmMdtLearnedJoinState* for a full description of this command. The *pimsmMdtLearnedJoinState* command fetches and describes the learned join states for the current Data MDT Range defined in *pimsmDataMdtRange*. The important subcommands and options of this command are:

pimsmMdtLearnedJoinState Subcommands

Member	Usage
setDefault	Sets the options to default values.
getAllMdtJoinsgetMdtJoin	Gets the learned states for all joins associated

pimsmMdtLearnedJoinState Options

Member	Usage
numSources	*(Read-only)* The number of sources associated with the pimsmMdtLearnedJoinState.
numGroupsPerSource	*(Read-only)* The number of groups received per source address.

pimsmCrpRange

Refer to NAME - pimsmCRPRange for a full description of this command. The *pimsmcrpRange* command describes a range of Candidate Rendezvous Points under a PIM-SM interface. CRP ranges are added into the pimsmInterface list using the pimsmInterface*addCRPRange* command. The important subcommands and options of this command are:

pimsmCRPRange Subcommands

Member	Usage
setDefault	Sets the options to default values.

pimsmCRPRange Options

Member	Usage
enable	Enables the use of this CRP range. *(default = disable).*
cRPAddress	Start address of the set of candidate RPs to be simulated.
routerCount	Total number of candidate RPs to be simulated starting from C-RP Address. A contiguous address range is used for this RP range simulation.
meshingType	This indicates if the mappings for groups and RP addresses are **Fully-Meshed** or **One-To-One.**
groupAddress	Indicates the starting group address of the group range for which the candidate RP will advertise candidacy.
groupCount	Indicates the number of groups in the range.
groupMaskLen	Mask width (prefix length in bits) for the group range.
periodicAdvertisementInterval	Rate controlling variable indicating how many C-RP-Adv messages can be sent in the specified time interval.
advertisementHoldTime	The time interval (in seconds) between two consecutive Candidate RP advertisements.
backOffInterval	The back off time interval for the C-RP-Adv messages.
priorityValue	Value of priority field sent in candidate RP advertisement messages.
priorityType	This indicates the type of priority to be held by the candidate RPs (CRPs). One of: • Same: *(default)* CRPs send advertisement messages with time invariant fixed priority as specified in CRP Advertisement Message Priority. • Incremental: Priority starts from the configured value and with every Priority Change Interval, the CRP's priority get incremented by 1.

Member	Usage
	- Random: The start value is selected based on a pseudorandom number generator with every Priority Change Interval, when sending the next batch of CRP-Adv messages.
priorityChangeInterval	Time interval after which priority of all the RPs get changed, if priority type is incremental or random.

pimsmLearnedBSRInfo

Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for an overview of the Ixia RIP test model. RIPng routers are used to simulate IPv6 routers using the RIPng protocol as defined in RFC 2080. The RIPng-related commands are:

Refer to NAME - pimsmLearnedBSRInfo for a full description of this command. The *pimsmLearnedBSR* command fetches and describes the learned data BSR information for the current interface defined in pimsmInterface. The important subcommands and options of this command are:

pimsmLearnedBSR Subcommands

Member	Usage
setDefault	Sets the options to default values.
requestLearnedCRPBSRInfo	Requests the learned BSR information for the respective BSR Router.
getLearnedBSRInfo	Retrieves the full list of learned BSR info.

pimsmLearnedBSR Options

Member	Usage
bSRAddress	*(Read-only)* The address of the elected bootstrap Router that is sending periodic bootstrap messages.
bSRPriority	*(Read-only)* Priority of the elected Bootstrap router as received in Bootstrap messages or configured priority.
bSRState	*(Read-only)* The state of the configured bootstrap router. The options are - Not started - Pending - Candidate - Elected
bSRTimerValue	*(Read-only)* The elapsed time (in seconds) since last bootstrap message was received (in case not acting as elected bootstrap router) or since last bootstrap message was sent (in case of elected bootstrap router).

pimsmLearnedCRPInfo

Refer to NAME - pimsmLearnedCRPInfo for a full description of this command. The *pimsmInterfaceLearnedCRPInfo* command fetches and describes the learned CRP information for the current interface defined in pimsmInterface. The important subcommands and options of this command are:

pimsmLearnedCRP Subcommands

Member	Usage
setDefault	Sets the options to default values.
requestLearnedCRPBSRInfo	Requests the learned CRP information for the PIMSM interface.
getFirstLearnedInfo getNextLearnedInfo	Retrieves the first learned info entry, then iterates through the list of additional entries.

pimsmLearnedCRP Options

Member	Usage
cRPAddress	*(Read-only)* The RP address expressing candidacy for this specific group. If the entire set is displayed then, there can be multiple RPs that have expressed candidacy for the same group.
groupAddress	*(Read-only)* Group Address learned through Candidate RP Advertisements or Bootstrap Messages. Configured C-RP-Range values on this PIM interface are not shown here.
cRPPriority	*(Read-only)* Indicates thepriority of this candidate RP.
groupMaskWidth	The number of bits in the mask applied to the group address. (The masked bits in the group address form the address prefix.)The default value is 32. The valid range is 1 to 128, depending on address family type.
mappingExpiryTimerValue	*(Read-only)* The expiry time for this specific record as received in C-RP-Adv Message/Bootstrap Message.

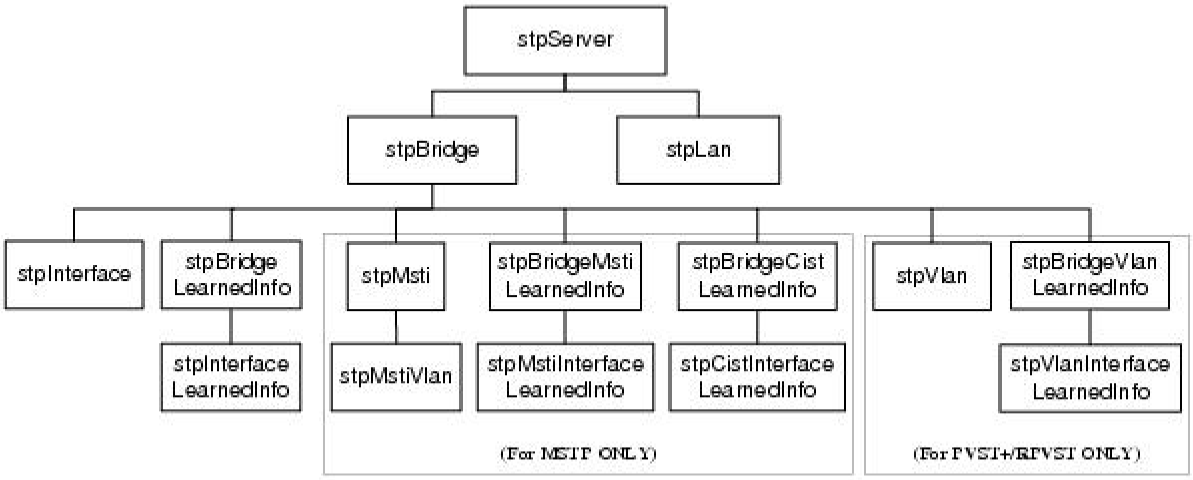
STP

Ixia ports are used to simulate STP/RSTP bridges using the RSTP protocol as defined in *IEEE 802.1D-2004,* which supports backward compatibility with STP. MSTP, per IEEE 802.1q, 2003 edition, as well as PVST+/RPVST+ are also supported. The STP-related commands are:

- stpServer — provides access to the STP part of a port's protocol server.
- stpBridge — represents a simulated STP bridge which holds a list of interfaces associated with the simulated bridge.
- stpInterface — a set of interfaces to be included in stpBridge.
- stpLan — a set of STP LANs to be included in stpServer.
- stpBridgeLearnedInfo — learned information associated with an stpBridge.
- stpInterfaceLearnedInfo — learned information associated with an interface on an stpBridge.
- stpMsti — a set of MSTIs to be included in stpBridge.
- stpMstiVlan — a set of MSTI VLANs to be included in stpMsti.
- stpBridgeLearnedInfo — learned information associated with an MSTI on an (MSTP) stpBridge.
- stpMstiInterfaceLearnedInfo — learned information associated with an MSTI for an interface on an (MSTP) stpBridge.
- stpBridgeCistLearnedInfo — learned information associated with a CIST on an (MSTP) stpBridge.
- stpCistInterfaceLearnedInfo — learned information associated with a CIST for an interface on an (MSTP) stpBridge.
- stpVlan — a set of STP VLANs for use with PVST+/RPVST+ to be included in stpBridge.
- stpBridgeVlanLearnedInfo — learned information associated with a VLAN on a PVST+/RPVST+ stpBridge.
- stpVlanInterfaceLearnedInfo — learned information associated with a VLAN for an interface on a PVST+/RPVST+ stpBridge.

These commands and the data that they maintain are arranged in an hierarchy, as shown in the following figure.

STP Command Hierarchy


stpServer
stpBridge
stpLan
stpInterface
stpBridge LearnedInfo
stpInterface LearnedInfo
stpMsti
stpMstiVlan
stpBridgeMsti LearnedInfo
stpMstiInterface LearnedInfo
stpBridgeCist LearnedInfo
stpCistInterface LearnedInfo
(For MSTP ONLY)
stpVlan
stpBridgeVlan LearnedInfo
stpVlanInterface LearnedInfo
(For PVST+/RPVST ONLY)

stpServer

Refer to NAME - stpServer for a full description of this command. This command is necessary in order to access the STP component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
stpServer select 1 5 2
```

will access the STP server for chassis 1, card 5, port 2.

This command holds a list of the simulated bridges. The definition of these bridges occurs using the *stpBridge* and subsidiary commands. The important subcommands of this command are:

stpServer Subcommands

Category	Member	Usage
	select	Selects the chassis, card, and port to operate on.
	write	Writes the changes to the server.
Bridges	clearAllBridges	Removes all bridges from the list of bridges.
	addBridge	Adds a bridge to the list of bridges. The bridge must have been previously configured through the use of the stpBridge. command.
	getBridge getFirstBridge getNextBridge	Accesses a particular bridge from the list, either directly by name or by iterating through all of the bridges. *getFirstBridge* should be called before *getNextBridge.* The data appears in the stpBridge. command.
	delBridge	Deletes a particular bridge from the list.
	setBridge	This command allows you to change a bridge's configuration on the fly while the STP protocol is running. Changes to the bridge's configuration are made with the stpBridge. command. This call must be followed by a call to stpServer *write.*
	showBridgeNames	Returns name of bridges in the list on the selected port. Calling the *select* command is recommended before calling this command.
	updateBridgeParameters	Updates the current bridge. *Get* commands need to be

Category	Member	Usage
		called before calling this command. To effect changes in topology once the RSTP is up and running and the switching network has converged, the root ID can be changed. (The Update Parameters button in the GUI allows this change, once the user has changed the Root ID.)
LANs	clearAllLans	Clears the LAN list.
	addLan	Adds a LAN to the LAN list.
	getLan getFirstLan getNextLan	Gets "lanName" and refreshes the options. Gets the first LAN from the LAN list and refreshes the options. Gets the next LAN from the LAN list and refreshes the options. Accesses a particular LAN from the list, either directly by name or by iterating through all of the LANs. *getFirstLan* should be called before *getNextLan.* The data appears in the stpLan command.
	delLan	Deletes a LAN from the LAN list. If no lanName is specified, it deletes the current one.
	setLan	Edit on the fly "lanName." If no lanName is specified, the current one will be modified.
	showLanNames	Returns the names of LANs in the list on the selected port. Calling *select* command is recommended before calling this command.

stpBridge

Refer to NAME - stpBridge for a full description of this command. The *stpBridge* command represents a simulated STP bridge. In addition to some identifying options, it holds lists for the bridge:

- Interfaces — a list of physical interfaces to the bridge over which STP messages are sent, constructed in the stpInterface command.
- (For MSTP only) MSTIs — a list of multiple spanning tree instances (MSTIs) associated with this bridge.
- (For PVST+ and RPVST+ only) VLANs — a list of virtual LANs (VLANs) associated with this bridge. The first VLAN is put under the bridge by default, because VLAN 1 (Common Spanning Tree/CST) must be run for all interfaces on the bridge for PVST+/RPVST+.

The important options and subcommands of this command are:

stpBridge Options

Member	Usage
enable	Enables or disables the bridge's simulation. *(default = disabled)*
mode	The version of the STP protocol that is being used on the

Member	Usage
	bridge. One of the following options: • STP — Spanning Tree Protocol • RSTP — Rapid Spanning Tree Protocol *(default)* • MSTP — Multiple Spanning Tree Protocol • PVST+ — Per-VLAN Spanning Tree Plus Protocol. • RPVST+ — Rapid Per-VLAN Spanning Tree Plus Protocol.
bridgeMacAddress	The 6-byte bridge MAC Address for this bridge. *(default = 0F 00 00 00 00)*
bridgeSystemId	The system ID for the bridge. The valid range is 0 to 4,095. *(default = 0)*
bridgePriority	The bridge priority for this bridge.The valid range is 0 to 61,440, in multiples of 4,096. *(default = 32,768)*
enableAutoPickBridgeMac	In this mode, the state machine selects one of the MAC addresses among all of the attached interfaces for a particular emulated bridge as its bridge MAC address. *(default = enabled)*
rootMacAddress	(For STP and RSTP) The 6-byte MAC address for the root bridge. *(default = 00:00:00:00:00:00)*
rootSystemId	(For STP and RSTP) The system ID for the root bridge. The valid range is 0 to 4,095. *(default = 0)*
rootPriority	(For STP and RSTP) The bridge priority for the root bridge. The valid range is 0 to 61,440, in increments of 4,096. *(default = 32,768)*
rootCost	(For STP and RSTP) The administrative cost for the shortest path from this bridge to the root bridge. The valid range is 0 to 4,294,967,295. *(default = 0)*
helloInterval	The length of time between transmission of Hello messages from the root bridge (in milliseconds). The valid range is 500 msec to 255 sec. *(default = 2,000 msec (2 sec))*
forwardDelay	The delay used for a port's change to the forwarding state. (in milliseconds) The valid range is 500 msec to 255 sec. *(default = 15,000 msec (15 sec))*
maxAge	The maximum configuration message aging time (in milliseconds). The valid range is 500 msec to 255 sec. *(default = 20,000 msec (20 sec))*
portPriority	The port priority. The valid range is to 240, in multiples of 16. *(default = 0)*
extRootPriority	(For use with MSTP only) The priority value of the root bridge for the CIST/MSTP region (external). Part of the CIST External Root Identifier. The valid range is 0 to 61,440, in increments of 4096. *(default = 32,768)*
extRootMacAddress	(For use with MSTP only) The CIST external root MAC address. Part of the CIST external root identifier. A 6-byte bridge MAC address. *(default = 00:00:00:00:00:00)*
extRootCost	(For use with MSTP only) The CIST external root path cost.

Member	Usage
	The valid range is 0 to 4,294,967,295. *(default = 0)*
regRootPriority	(For use with MSTP only) The priority value of the root bridge for the CIST/MSTP region (internal). Part of the CIST regional root identifier. The valid range is 0 to 61,440, in increments of 4,096. *(default = 32,768)*
regRootMacAddress	(For use with MSTP only) The CIST regional root MAC address. Part of the CIST regional root identifier. A 6-byte bridge MAC address. *(default = 00:00:00:00:00:00)*
regRootCost	(For use with MSTP only) The CIST regional (internal) root path cost. The valid range is 0 to 4,294,967,295. *(default = 0)*
cstRootPriority	(For use with PVST+ and RPVST+ only) The common spanning tree (CST) priority of the root. The valid range is 0 to 61,440, in increments of 4,096. *(default = 32,768)*
cstRootMacAddress	(For use with PVST+ and RPVST+ only) The common spanning tree (CST) 6-byte root MAC address. *(default = 00:00:00:00:00:00)*
cstRootCost	(For use with PVST+ and RPVST+ only) The common spanning tree (CST) root path cost. The valid range is 0 to 4,294,967,295. *(default = 0)*
cstVlanPortPriority	(For use with PVST+ and RPVST+ only) The common spanning tree (CST) VLAN port priority. The valid range is 0 to 63. *(default = 32)*
mstcConfigName	(For use with MSTP only) The name of the multiple spanning tree configuration being used. Format = MSTC ID-n (editable by user).
mstcConfigRevisionNumber	(For use with MSTP only) The revision number of the multiple spanning tree configuration being used. A 2-byte unsigned integer. *(default = 0)*
messageAge	The message age time parameter in the BPDU (in milliseconds). (It should be less than the Max. Age.) The valid range is *500 msec to 255 sec*. *(default = 0)*
cistRemainingHops	(For use with MSTP only) The number of additional bridge-to-bridge hops that will be allowed for the MSTP BPDUs. The root sets the maximum hop count, and each subsequent bridge decrements this value by 1. The valid range is *1 to 255*. *(default = 20)*
name	*(Read-only)* The name of the bridge that will be used as a unique key to retrieve the object.

stpBridge Subcommands

Member	Usage
setDefault	Sets the options to default values.
clearAllInterfaces	Clears the interface list on the selected bridge.
addInterface	Adds a new interface. The interface must have been previously configured through the use of the stpInterface command.

Member	Usage
getInterface getFirstInterface getNextInterface	Accesses a particular interface either by name or by iterating through all of the interfaces. *getFirstInterface* should be called before *getNextInterface*. The data appears in the stpInterface command.
delInterface	Deletes a particular interface.
setInterface	This command may be used to change an individual interface on the fly while the STP protocol server is running. Changes to the interface are made with the stpInterface command. This command must be followed with a call to stpServer *write.*
showInterfaceNames	Returns names of interfaces in the list on the selected bridge. Calling *select* command is recommended before calling this command.
updateInterfaceParameters	To effect changes in link conditions and topology, the user may change the cost of an interface on the fly. Updates the current interface. *Get* commands should be called before calling *update* command.
clearAllMstis	Clears the MSTI list.
addMsti	Adds an MSTI to the list.
getMsti getFirstMsti getNextMsti	Accesses a particular MSTI from the list, either directly by name or by iterating through all of the MSTIs. *getFirstMsti* should be called before *getNextMsti.* The data appears in the stpMsti command.
delMsti	Deletes an MSTI from the MSTI list. If no MstiName is specified, it deletes the current one.
setMsti	Edits the "MstiName" on the fly. If no MstiName is specified, the current one will be modified.
showMstiNames	Returns the names of MSTIs in the list on the selected port. Calling the *select* command is recommended before calling this command.
updateMstiParameters	Updates the current MSTI. *Get* commands need to be called before calling this command.
clearAllVlans	Clears the VLAN list on the selected bridge.
addVlan	Adds a new VLAN. The VLAN must have been previously configured through the use of the stpVlan command.
getVlan getFirstVlan getFirstVlanWithCST getNextVlan	Accesses a particular VLAN either by name, or by iterating through all of the VLANs. *getFirstVlan* should be called before *getNextVlan*. The getFirstVlanWithCST subcommand is related to VLAN 1 (for the Common Spanning Tree/CST). The data appears in the stpVlan command.
delVlan	Deletes a particular VLAN.
setVlan	This command may be used to change an individual VLAN on the fly while the STP protocol server is running. Changes to the interface are made with the stpVlan command. This com-

Member	Usage
	mand must be followed with a call to stpServer *write.*
showVlanNames	Returns names of VLANs in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.
updateVlanParameters	Updates the current VLAN. *Get* commands need to be called before calling this command.
requestLearnedInfo	Requests learned info for the bridge.
getLearnedInfo	Gets learned info list. When it returns TCL_OK it means that we get learned info.
getCistLearnedInfo	Gets learned info for the MSTP common and internal spanning tree (CIST).
getFirstMstiLearnedInfo	Gets learned info for the first MSTI configured for this bridge.
getFirstVlanLearnedInfo	Gets learned info for VLAN 1 configured for this bridge.
getNextVlanLearnedInfo	Gets learned info for the next VLAN configured for this bridge — iterates through the list of VLANs on this bridge.
cistTopologyChange	Generates topology change BPDUs for the MSTP CIST, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree.
generateTopologyChange	Generates topology change BPDUs, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree.

stpInterface

Refer to NAME - stpInterface for a full description of this command. The *stpInterface* command represents an interface on a simulated STP bridge. An STP interface uses an interface defined with the stpBridge *addInterface* command. The important options and subcommands of this command are:

stpInterface Options

Member	Usage
enable	Enables or disables the use of the interface. (*default= disabled)*
name	*(Read-only)* The name of the interface that will be used as a unique key to retrieve the object.
protocolInterface Description	The description option associated with an *stpInterface* when it was created. The IP address and mask are read from the interface entry.
interfaceType	The type of link on the STP interface. One of: • Point to Point • Shared Link *(default)*
cost	The administrative path cost assigned to this interface. The valid range is 0 to 4,294,967,295. *(default = 1)*
interBpduGap	The length of time between transmissions of BPDUs, in milliseconds. The valid range is 0 msec to 60,000 msec*. (default = 0)*
enableJitter	Staggered transmit (jitter) for Hello messages. If set, then the

Member	Usage
	jitter feature is enabled. *(default = enabled)*
jitterPercentage	The maximum percentage of +/- variation (jitter) from the Hello message transmission interval.
mstiId	The identifier of the MSTP MSTI. An unsigned integer. The valid range is 1 to 4,094. *(default = 1)*
vlanId	The VLAN Identifier (ID). The valid range is 2 to 4,094. *(default = 2)*
pvId	The port VLAN ID. This value must be the same for all ports participating in the PVST+/RPVST+ protocol. The valid range is 1 to 4,094. *(default = 1)*
portNum	The port number associated with this STP interface. If enableAutoPickPortNum is set, the port number will be automatically assigned (not editable by the user). If enableAutoPickPortNum is not set, the port number can be configured by the user. The valid range is 1 to 4,095. *(default = 1)*
enableAutoPickPortNum	If set, then the Auto-Pick Port Number feature is enabled and each STP interface configured for the same bridge will be assigned a unique port number automatically.(*default = enabled*)

stpInterface Subcommands

Member	Usage
setDefault	Sets the options to default values.

stpLan

Refer to NAME - stpLan for a full description of this command. The *stpLan* command describes a list of bridged LANs associated with the STP/RSTP bridges. LANs are added into stpServer list using the stpServer *addLan* command. The important options and subcommands of this command are:

stpLan Options

Member	Usage
enable	Enables the use of this STP LAN. *(default = disabled)*
startMacAddress	The first 6-byte MAC address in the range. *(default = 00:00:00:00:00:00)*
count	The number of MAC addresses in the LAN range. The valid range is 1 to 500. *(default = 1)*
name	(Read-only) Name of the LAN that will be used as a unique key to retrieve the object.

stpLan Subcommands

Member	Usage
setDefault	Sets the options to default values.

stpBridgeLearnedInfo

Refer to NAME - stpBridgeLearnedInfo for a full description of this command. The *stpBridgeLearnedInfo* command fetches and describes the learned states for a the current bridge defined in stpBridge. The important options and subcommands of this command are:

stpBridgeLearnedInfo Options

Member	Usage
bridgeMacAddress	*(Read-only)* The 6-byte MAC address of the bridge.
rootPriority	*(Read-only)* The priority of the root bridge.
rootMacAddress	*(Read-only)* The 6-byte MAC address of the root bridge.
rootCost	*(Read-only)* The administrative cost associated with the root bridge.
designatedPriority	*(Read-only)* The priority of the designated bridge on the LAN segment.
designatedMacAddress	*(Read-only)* The 6-byte MAC address of the designated bridge on the LAN segment.
designatedPortId	*(Read-only)* The port ID of the designated bridge's designated port on the LAN segment.

stpBridgeLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.
getFirstInterfaceLearnedInfo	Gets first interface learned info for the selected Bridge.
getNextInterfaceLearnedInfo	Gets next interface learned info for the selected Bridge.

stpInterfaceLearnedInfo

Refer to NAME - stpInterfaceLearnedInfo for a full description of this command. The *stpInterfaceLearnedInfo* command fetches and describes the learned states for the current interface defined in stpInterface. The important options and subcommands of this command are:

stpInterfaceLearnedInfo Options

Member	Usage
protocolInterfaceDescription	*(Read-only)* The descriptive identifier of the protocol interface.
interfaceRole	*(Read-only)* The role of the Interface. One of the following options: • Disabled • Root • Designated • Alternate • Backup
interfaceState	*(Read-only)* The state of the interface. One of the following options:

Member	Usage
	- Discarding - Learning - Forwarding
rootPriority	*(Read-only)* The priority value of the root bridge.
rootMacAddress	*(Read-only)* The 6-byte MAC address of the root bridge.
rootCost	*(Read-only)* The administrative cost of the path to the root bridge.
designatedPriority	*(Read-only)* The priority of the designated bridge on the LAN segment.
designatedMacAddress	*(Read-only)* The 6-byte MAC address of the designated bridge on the LAN segment.
designatedPortId	*(Read-only)* The port ID of the designated bridge's designated port on the LAN segment.

stpInterfaceLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.

stpMsti

Refer to NAME - stpMsti for a full description of this command. The *stpMsti* command describes a list of multiple spanning tree instances (MSTIs) associated with an MSTP bridge. MSTIs are added into stpBridge list using the stpBridge *addMsti* command. In addition to some identifying options, it holds a list for the MSTI:

- mstiVlans — a list of VLAN ranges associated with the MSTI constructed in the stpMstiVlan command.

The important options and subcommands of this command are:

stpMsti Options

Member	Usage
enable	Enables the use of this MSTP MSTI. *(default = disabled)*
mstiId	The identifier for this MST instance (MSTI). The valid range is 1 to 4,094. *(default = 1)*
mstiRegionalRootId	The Regional Root ID value for this MSTI. A 6-byte MAC address. *(default = 00:00:00:00:00:00)*
mstiRootPriority	The MSTI root priority. This is part of the MSTI regional root identifier. The valid range is 0 to 61,440, in increments of 4,096. *(default = 32,768)*
mstiInternalRootPathCost	The MSTI internal root path cost. A 4-byte unsigned integer. *(default is 0)*
mstiName	The name of the MSTI which is configured from the list of MSTIs. Format: MSTI ID-n. (Editable by the user.)
mstiHops	The number of MSTI hops remaining. An unsigned integer. The valid range is 1 to 255. *(default = 20)*
name	*(Read-only)* The name of the MSTI that will be used as a unique key to retrieve the object.

Member	Usage
portPriority	The MSTI port priority. This is part of the MSTI regional root identifier. An unsigned integer; a multiple of 16. The valid range is *0 to 240*. *(default = 0)*

stpMsti Subcommands

Member	Usage
setDefault	Sets the options to default values.
generateTopologyChange	Generates topology change BPDUs, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree. When used with MSTP bridges, this topology change applies to all MSTIs related to those bridges.
addVlanRange	Adds an MSTI VLAN range to the list.
getVlanRange getFirstVlanRange getNextVlanRange	Accesses a particular MSTI VLAN range from the list, either directly by name or by iterating through all of the MSTI VLAN ranges. *getFirstVlanRange* should be called before *getNextVlan.* The data appears in the stpMstiVlan command.
setVlanRange	Edits the vlanRangeName on the fly. If no vlanRangeName is specified, the current one will be modified.
showVlanRangesNames	Returns the names of MSTI VLAN Ranges in the list on the selected port.

stpMstiVlan

Refer to NAME - stpMstiVlan for a full description of this command. The *stpMstiVlan* command describes a list of VLAN ranges for an MSTI associated with an MSTP bridge. VLAN ranges are added into the stpMsti list using the stpMsti *addVlanRanges* command. The important options and subcommands of this command are:

stpMstiVlan Options

Member	Usage
startVlanId	The ID for the first VLAN in the VLAN range to which the MSTI is mapped. An unsigned integer. The valid range is 1 to 4,094. *(default = 1)*
endVlanId	The ID for the last VLAN in the VLAN range to which the MSTI is mapped. An unsigned integer. The valid range is 1 to 4,094. *(default = 1)*
name	*(Read-only)* Name of the MSTI VLAN range that will be used as a unique key to retrieve the object.

stpMstiVlan Subcommand

Member	Usage
setDefault	Sets the options to default values.

stpBridgeMstiLearnedInfo

Refer to stpBridgeMstiLearnedInfo for a full description of this command. The *stpBridgeMstiLearnedInfo* command fetches and describes the learned information for the MSTI on the advertising MSTP bridge. The important options and subcommands of this command are:

stpBridgeMstiLearnedInfo Options

Member	Usage
rootMacAddress	*(Read-only)* The root bridge MAC address being advertised.
rootPriority	*(Read-only)* The priority being advertised for the root bridge.
rootCost	*(Read-only)* The cost for the shortest path from the advertising bridge to the root bridge.
mstiId	*(Read-only)* The MSTI identifier being advertised.

stpBridgeMstiLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.
getFirstInterfaceLearnedInfo	Gets the learned info for the first interface.
getNextInterfaceLearnedInfo	Gets the learned info for the next interface.

stpMstiInterfaceLearnedInfo

Refer to NAME - stpMstiInterfaceLearnedInfo for a full description of this command. The *stpMstiInterfaceLearnedInfo* command fetches and describes the learned information for the advertised MSTI interface. The important options and subcommands of this command are:

stpMstiInterfaceLearnedInfo Options

Member	Usage
interfaceRole	*(Read-only)* The role of the advertised interface. One of the following options: • Disabled • Root • Designated • Alternate • Backup • Master
interfaceState	*(Read-only)* The state of the advertised MSTP/MSTI interface. One of the following options: • Discarding (Discarding MAC frames) • Learning (MAC frame learning) • Forwarding (Forwarding MAC frames)
designatedPriority	*(Read-only)* The priority of the advertised designated MSTP bridge on the LAN segment.
designatedMacAddress	*(Read-only)* The 6-byte MAC address of the advertised designated MSTP bridge on the LAN segment.
designatedPortId	*(Read-only)* The port ID of the advertised designated MSTP bridge's port on the LAN segment.
protocolInterfaceDescription	*(Read-only)* The descriptive identifier of this advertised protocol interface.

stpMstiInterfaceLearnedInfo Subcommand

Member	Usage
setDefault	Sets the options to default values.

stpBridgeCistLearnedInfo

Refer to NAME - stpBridgeCistLearnedInfo for a full description of this command. The *stpBridgeCistLearnedInfo* command fetches and describes the The *stpBridgeCistLearnedInfo* command fetches and describes the learned information for the CIST on the advertising MSTP bridge. The important options and subcommands of this command are:

stpBridgeCistLearnedInfo Options

Member	Usage
rootMacAddress	*(Read-only)* The root bridge MAC address being advertised.
rootPriority	*(Read-only)* The priority being advertised for the root bridge.
rootCost	*(Read-only)* The cost for the shortest path from the advertising bridge to the root bridge.
regRootMacAddress	*(Read-only)* The regional root MAC address being advertised by the bridge.
regRootPriority	*(Read-only)* The regional root priority being advertised by the bridge.
regRootCost	*(Read-only)* The cost for the shortest path from the advertising bridge to the regional root bridge.

stpBridgeCistLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.
getFirstInterfaceLearnedInfo	Gets the learned info for the first interface.
getNextInterfaceLearnedInfo	Gets the learned info for the next interface.

stpCistInterfaceLearnedInfo

Refer to NAME - stpCistInterfaceLearnedInfo for a full description of this command. The *stpCistInterfaceLearnedInfo* command fetches and describes the learned information for the CIST interface associated with the advertising MSTP bridge. The important options and subcommands of this command are:

stpCistInterfaceLearnedInfo Options

Member	Usage
interfaceRole	*(Read-only)* The role of the advertised interface. One of the following options: • Disabled • Root • Designated • Alternate • Backup
interfaceState	*(Read-only)* The state of the advertised interface. One of the following options:

Member	Usage
	• Discarding (discarding MAC frames) • Learning (MAC frame learning) • Forwarding (forwarding MAC frames)
designatedPriority	*(Read-only)* The priority of the advertised designated MSTP bridge on the LAN segment.
designatedMac	*(Read-only)* The 6-byte MAC address of the advertised designated MSTP bridge on the LAN segment.
designatedPortId	*(Read-only)* The port ID of the advertised eesignated MSTP bridge's port on the LAN segment.
protocolInterfaceDescription	*(Read-only)* The descriptive identifier of this advertised protocol interface.

stpCistInterfaceLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.

stpVlan

Refer to NAME - stpVlan for a full description of this command. The *stpVlan* command describes a list of VLANs associated with a PVST+/RPVST+ bridge. VLAN ranges are added into the *stpBridge* list using the *stpBridge addVlan* command. VLAN 1 is added to the PVST+/RPVST+ bridge by default, because VLAN 1 (CST) must be run for all interfaces on the PVST+/RPVST+ bridge. The important options and subcommands of this command are:

stpVlan Options

Member	Usage
enable	Enables the use of this STP VLAN. (*default = disabled)*
vlanId	The identifier for this VLAN. The valid range is 2 to 4,094. *(default = 2)*
PortPriority	The root priority for this port. The valid range is 0 to 61,440, in increments of 4,096. *(default = 32,768)*
PortPathCost	The root path cost for this port. The valid range is 0 to 4,294,967,295. *(default = 0)*
vlanPortPriority	The VLAN port priority. The valid range is 0 to 63. *(default = 32)*
PortMac	The 6-byte MAC address of the port. *(default = 00:00:00:00:00:00)*

stpVlan Subcommands

Member	Usage
setDefault	Sets the options to default values.
generateTopologyChange	After the protocol is up and running, this subcommand generates topology change BPDUs, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree.

stpBridgeVlanLearnedInfo

Refer to NAME - stpBridgeVlanLearnedInfo for a full description of this command. The *stpBridgeVlanLearnedInfo* command fetches and describes the learned information for the common spanning tree (CST) for the advertising PVST+/RPVST+ bridge. The important options and subcommands of this command are:

stpBridgeVlanLearnedInfo Options

Member	Usage
rootMacAddress	*(Read-only)* The root bridge MAC address being advertised.
rootPriority	*(Read-only)* The priority being advertised for the root bridge.
rootCost	*(Read-only)* The cost for the shortest path from the advertising bridge to the root bridge.
vlanId	*(Read-only)* The VLAN identifier being advertised.

stpBridgeVlanLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.
getFirstInterfaceLearnedInfo	Gets the learned info for the first interface.
getNextInterfaceLearnedInfo	Gets the learned info for the next interface.

stpVlanInterfaceLearnedInfo

Refer to NAME - stpInterfaceLearnedInfo for a full description of this command. The *stpVlanInterfaceLearnedInfo* command fetches and describes the learned information for the VLAN interface associated with the advertising PVST+/RPVST+ bridge. The important options and subcommands of this command are:

stpVlanInterfaceLearnedInfo Options

Member	Usage
interfaceRole	*(Read-only)* The role of the advertised interface. One of the following options: • Disabled • Root • Designated • Alternate • Backup
interfaceState	*(Read-only)* The state of the advertised interface. One of the following options: • Discarding (Discarding MAC frames) • Learning (MAC frame learning) • Forwarding (Forwarding MAC frames) • Listening (Available for use with PSVT+/RPVST+ only)
designatedPriority	*(Read-only)* The priority of the advertising designated PVST+/RPVST+ bridge.
designatedMacAddress	*(Read-only)* The 6-byte MAC address of the advertising designated PVST+/RPVST+ bridge.

Member	Usage
designatedPortId	*(Read-only)* The port ID of the advertising designated PVST+/RPVST+ bridge's port on the LAN segment.
protocolInterfaceDescription	*(Read-only)* The descriptive identifier for the advertised protocol interface.
rootPriority	*(Read-only)* The priority being advertised for the Root bridge.
rootMac	*(Read-only)* The Root bridge MAC address being advertised by the bridge.
rootCost	*(Read-only)* The cost for the shortest path from the advertising bridge to the Regional Root bridge.

stpVlanInterfaceLearnedInfo Subcommands

Member	Usage
setDefault	Sets the options to default values.

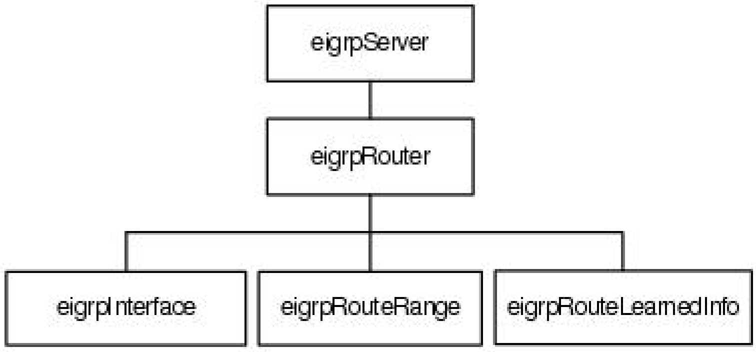
EIGRP

Ixia ports are used to simulate routers using the enhanced interior gateway routing protocol (EIGRP). The EIGRP-related commands are:

- eigrpServer — provides access to the EIGRP part of a port's protocol server.
- eigrpRouter — represents a simulated EIGRP router which holds a list of interfaces associated with the simulated router.
- eigrpInterface — an interface to be included in eigrpRouter.
- eigrpRouteRange — a set of EIGRP route ranges, to be included in eigrpRouter.
- eigrpRouteLearnedInfo — learned route information associated with an eigrpRouter.

These commands and the data that they maintain are arranged in a hierarchy, as shown in the following figure.

EIGRP Command Hierarchy



eigrpServer

Refer to NAME - eigrpServer for a complete description of this command. The *eigrpServer* command is necessary in order to access the EIGRP component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
eigrpServer select 1 5 2
```

will access the EIGRP server for chassis 1, card 5, port 2.

This command holds a list of the simulated routers. The definition of these routers occurs using the *eigrpRouter* command and subsidiary commands. See eigrpServer for full details. The important subcommands of this command are:

eigrpServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.
clearAllRouters	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the *eigrpRouter* command.
getRouter	Gets "routerName" and refreshes the options.
getFirstRouter	Gets the first router from the router list and refreshes the options.

Member	Usage
getNextRouter	Gets the first router from the router list and refreshes the options.
populateFirstRouteRange	Fetches and populates the first route range from the list.
delRouter	Deletes a particular router from the list.
setRouter	It is possible to change EIGRP router configuration on -the-fly. In order to do this, the following steps are necessary: 1. Modify the router's configuration with *eigrpRouter.* (*eigrpRouter* has capabilities for modifying elements of underlying objects as well). 2. Use the *eigrpServer setRouter* command to set the values from *eigrpRouter* into IxHal. 3. Use the *eigrpServer write* command to write the changes to the hardware.
showRouterNames	Returns names of routers in the list on the selected port. Calling the *select* command before calling this command is recommended.
write	Writes the changes to the server.

eigrpRouter

See NAME - eigrpRouter for full details. The *eigrpRouter* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Interface — a router interface, which is constructed in the *eigrpInterface* command.
- Route ranges — routes to be advertised by the simulated router, which is constructed in the *eigrpRouteRange* command.
- Learned routes — routes learned by the simulated router.

eigrpRouter Options

Member	Usage
enable	Enables the simulated router.
routerId	This is an IP-formatted string. Its default value is dependent on card/port type.
ASNumber	The identifier of the autonomous system (AS) where this emulated EIGRP router is located. The valid range is 1 to 4,294,967,295. *(default = 1)*
routeActiveTime	It determines the maximum time (in minutes) for which a route learned from a neighbor will be active in the topology table, if the neighbor stops sending Hellos. The valid range is 1 to 4,294,967,295. (*default = 3 minutes*)
enableDiscardLearnedRoutes	If enabled, the router will not store learned routes; it will discard the routes. *(default = false)*
eigrpAddress Family	Denotes ip address type, one of ipv4 or ipv6. *(default = ipv4)*
K1	Advanced parameter, only used in condition checking for establishing neighbor relationship. The valid range is 0 to

Member	Usage
	255. *(default = 1)*
K2	Advanced parameter, only used in condition checking for establishing neighbor relationship. The valid range is 0 to 255. *(default = 0)*
K3	Advanced parameter, only used in condition checking for establishing neighbor relationship. The valid range is 0 to 255. *(default = 1)*
K4	Advanced parameter, only used in condition checking for establishing neighbor relationship.The valid range is 0 to 255. *(default = 0)*
K5	Advanced parameter, only used in condition checking for establishing neighbor relationship.The valid range is 0 to 255. *(default = 0)*
eigrpMajorVersion	The major version level of the EIGRP software. The valid range is 0 to 255. *(default = 1)*
eigrpMinorVersion	The minor version level of the EIGRP software. The valid range is 0 to 255. *(default = 2)*.
iosMajorVersion	The major version level of the referenced software. The valid range is 0 to 255. *(default = 12)*
iosMinorVersion	The major version level of the referenced software. The valid range is 0 to 255. *(default = 3)*
name	Indicates the name of the router which will be used as a unique key to retrieve the object. It is READ-ONLY.

eigrpRouter Subcommands

Member	Usage
setDefault	Sets default values for all configuration options.
addInterface	Adds an interface to the router. Currently, one interface is supported per emulated EIGRP router. **NOTE:** If an interface is added to an existing router and then before the router is selected by the get command again, *eigrpServer select* should be called. *eigrpServerwrite* can be called immediately without calling the *setRouter* command. It will behave as add-on-fly.
delInterface	Deletes the interface from the interface list of a selected router. If no interfaceName is specified, it deletes the current one. **NOTE:***eigrpServer write* can be called immediately, without calling the *setRouter* command. It will behave as add-on-fly.
getInterface	Gets the interfaceName on the selected router and refreshes the options.
getFirstInterface	Gets the first interface on the selected router and refreshes the options.
setInterface	Edits on the fly the interfaceName on the selected router. If no interfaceName is specified, the current one is modified.

Member	Usage
clearAllInterfaces	Clears the interface list on the selected router.
showInterfaceNames	Returns name of interfaces in the list on the selected router.
addRouteRange	Adds a route range to the route ranges list of a router. **NOTE:** If a route range is added to an existing router and then the router is selected by the *get* command again, *eigrpServer select* command should be called. *eigrpServer write* can be called immediately without calling the *setRouter* command. It will behave as add-on-fly.
delRouteRange	Deletes a route range from the route ranges list of a selected router. If no routeRangeName is specified, it deletes the current one. **NOTE:***eigrpServer write* can be called immediately without calling the *setRouter* command. It will behave as add-on-fly.
getRouteRange	Gets the routeRangeName on the selected router and refreshes the options.
getFirstRouteRange	Gets the first route range from the route range list on the selected router and refreshes the options.
getNextRouteRange	Gets the next router range from the router range list on the selected router and refreshes the options.
setRouteRange	Edits on the fly the routeRangeName on the selected router. If no routeRangeName is specified, the current one will be modified.
clearAllRouteRanges	Clears the router range list on the selected router.
showRouteRangeNames	Returns the name of router ranges in the list on the selected router.
requestLearnedInfo	Requests the route learned info.
getLearnedRouteList getFirstLearnedRoute getNextLearnedRoute	Gets the learned route list. It should be called after the *requestLearnedInfo* command. Gets the first learned route. It should be called after *getLearnedRouteList* is successful. Gets next learned route.

eigrpInterface

Refer to the NAME - eigrpInterface for a full description of this command. The *eigrpInterface* holds the information related to a single interface on the simulated router. Interfaces are added into the *eigrpRouter* interface list using the *eigrpRouter addInterface* command. In addition, learned LSAs from the DUT are made available through this command. The important options and subcommands of this command are:

eigrpInterface Options

Member	Usage
enable	Enables the EIGRP interface. *(default = disabled)*
interfaceId	The local ID associated with the interface, which is unique per router.
helloInterval	The time interval between Hello packets sent over the inter-

Member	Usage
	face, in seconds. *(default = 5 seconds)*
holdTime	The amount of time starting from the reception of a Hello from a neighbor until the moment when the neighbor is to be dropped if no further Hello is received from it, in seconds. *(default = 15 seconds)*
poisonedReverse	Enables poisoned reverse. If enabled, it lets the router learn a route through a particular interface and then advertise the route through the same interface, but with an infinite metric. *(default = enabled)*
bandwidth	The amount of bandwidth available on this link, in Kbps. The valid range is 1 to 4,294,967,295. *(default = 10,000)*
delay	The total of delays on the path to the route/network, in microseconds. The valid range is 0 to 4,294,967,295. *(default = 0)*
load	The amount of load on the link. The valid range is 0 to 255. *(default = 0)*
reliability	The reliability factor. The valid range is 0 to 255. *(default =255, which means 100% reliable)*
maxTlvPerPacket	The maximum number of TLVs that will be packed into a single update packet, taking MTU into consideration. The valid range is 0 to 255. A value of 0 means that maximum possible packing will be used, which depends on the MTU of the link. *(default = 30)*
enableBfdRegistration	Indicates if a BFD session is to be created to the EIGRP peer IP address once the EIGRP session is established. This allows EIGRP to use BFD to maintain IPv4 connectivity the EIGRP peer.
protocolInterfaceDescription	(*Read-only*.) The descriptive identifier of the protocol interface.
name	(*Read-only.*) Indicates the name of the interface which will be used as a unique key to retrieve the object.

eigrpInterface Subcommand

Member	Usage
setDefault	Sets the options to default values.

eigrpRouteRange

The *eigrpRouteRange* command describes an individual set of routes. Route ranges are added into *eigrpRouter* lists using the *eigrpRouter addRouteRange* command. See NAME - eigrpRouteRange for full details. The important options of this command are:

eigrpRouteRange Options

Member	Usage
enable	Enables the route range. *(default = disabled)*
firstRoute	The first route of the route range, in IPv4/IPv6 format. *(default = 0.0.0.0 for IPv4 and 0:0:0:0:0:0:0:0 for IPv6)*

Member	Usage
maskWidth	The network mask width for the route range (in bits). The valid range is from 0 to 32 bits for IPv4. *(default = 24)* The valid range is from 0 to 128 bits for IPv6. (*default*= 64)
noOfRoutes	The number of routes to be generated for this route range, based on the network address plus the network mask. The valid range is 1 to 16,777,215 for IPv4. *(default = 1)* The valid range is 1-4294967295 for IPv6. *(default = 1)*
nextHop	The immediate next hop IP address on the way to the destination address, in IPv4/IPv6 dotted decimal format. (default = 0.0.0.0 for IPv4) (default = 0:0:0:0:0:0:0:0 for IPv6)
hopCount	The number of hops on the way to the destination address. The valid range is 0 to 255. *(default = 0)*
bandwidth	The minimum amount of bandwidth available on this link, in Kbps. The valid range is 1 to 4,294,967,295. *(default = 10,000 Kbps)*
delay	The total of delays on the path to the route/network, in microseconds. The valid range is 0 to 4,294,967,295. *(default = 0)*
load	The amount of load on the link. The valid range is 0 to 255. *(default = 0)*
reliability	The reliability factor. The valid range is 0 to 255 (100% reliable). *(default = 255)*
mtu	The maximum transmission unit (MTU) allowed on this link, in bytes. The valid range is 0 to 16,777,215. *(default = 1,500 bytes)*
type	The type of route range: internal or external to the AS. One of the following options: • eigrpExternal • eigrpInternal *(default)*
arbitraryTag	(Available only for external route ranges.) An administrative tag applied to the route when it is redistributed between EIGRP and an external protocol, to prevent routing loops. Used as a route mapping filter. The valid range is 0 to 4,294,967,295. *(default = 0)*
externalMetric	(Available only for external route ranges.) The EIGRP vector metric for the cost of the path to this route/network. The valid range is 1 to 4,294,967,295. *(default = 1)*
source	(Available only for external route ranges.) The IPv4/Ipv6 address for the external source of the route information, in dotted decimal format. *(default = 0.0.0.0)*
originatingAS	(Available only for external route ranges.) The external AS where this route was originated. The valid range is 1 to 4,294,967,295. *(default = 1)*
protocolId	(Available only for external route ranges.) The external protocol where the route was originated, if applicable. One of: • eigrpIGRP *(default)*

Member	Usage
	- eigrpEnhancedIGRP - eigrpStatic - eigrpRIP - eigrpHello - eigrpOSPF - eigrpISIS - eigrpEGP - eigrpBGP - eigrpIDRP - eigrpConnected
flag	(Available only for external route ranges.) The origin of the advertised route. One of: - eigrpExternalRoute *(default)* - eigrpCandidateDefault
enablePacking	Enables packing of multiple destinations into a single Internal/external TLV. If disabled, only one destination will be packed into a single Internal/external TLV. *(default = enabled)*
destCount	(Available only if packing is enabled.) If packing is enabled, it indicates the maximum number of destinations that can be packed into a single internal/external TLV. A value of 0 means that maximum possible packing will be used, which depends on the MTU of the link. The valid range is 0 to 255. (*default = 90)*
name	*(Read-only)* The name of the interface which will be used as a unique key to retrieve the object.

eigrpRouteRange Subcommand

Member	Usage
setDefault	Sets the options to default values.

eigrpRouteLearnedInfo

Refer to NAME - eigrpRouteLearnedInfo for a complete description of this command. *eigrpRouteLearnedInfo* is used to retrieve the list of learned routes from the *requestLearnedRoutes* and *getLearnedRouteList* commands of eigrpInterface. Each enabled type of route is considered a separate list and must be retrieved with separate calls to the *getFirst/getNext* subcommands. See NAME - eigrpRouteLearnedInfo for full details. The important options and subcommands of this command are:

eigrpRouteLearnedInfo Options

Member	Usage
destination	*(Read-only)* The IPv4/IPv6 destination network that was advertised in the learned route.
prefixLength	*(Read-only)* IP prefix length for the route.
type	*(Read-only)* Indicates whether it is an internal or external route.
FD	*(Read-only)* The feasible distance. The sum of the reported distance and the link cost of the interface.

Member	Usage
neighbor	*(Read-only)* The neighbor from which the route was learned.
RD	*(Read-only)* The reported distance of the route advertised by the neighbor. It is calculated based on bandwidth, load, delay, and reliability.
hop_count	*(Read-only)* A routing metric used to measure the distance between a source and a destination. This is the hop count of the route learned from the neighbor.
next_hop	*(Read-only)* The IPv4/IPv6 next hop on the path to the destination contained in the learned route.

eigrpRouteLearnedInfo Subcommand

Member	Usage
setDefault	Sets the options to default values.

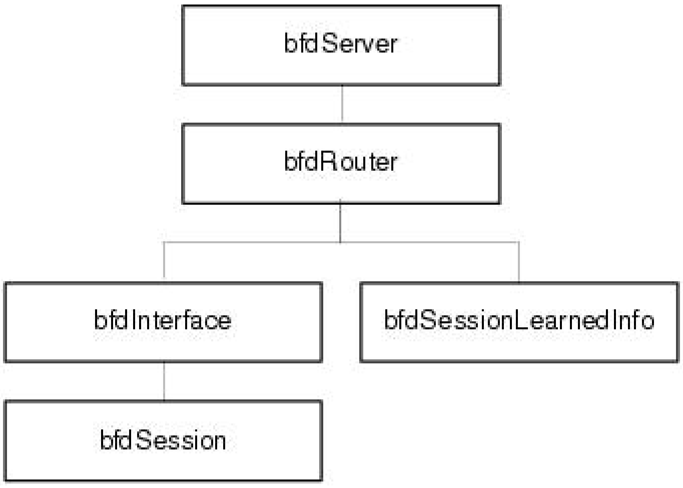
BFD

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to BFD. The BFD protocol is used to monitor the connectivity across multiple network hops. The BFD-related commands are:

- bfdServer — provides access to the BFD part of a port's protocol server.
- bfdRouter — a container used to hold three lists associated with the router: route ranges, interfaces, and link state advertisements (LSAs).
- bfdInterface — a network interface, which will be included in bfdRouter.
- bfdSession — configures a session under a BFD interface.
- bfdSessionLearnedInfo — views retrieved learned session information.

These commands and the data that they maintain are arranged in a hierarchy, as shown in the following figure.

BFD Command Hierarchy



bfdServer

Refer to NAME - bfdServer for a complete description of this command. The *bfdServer* command is necessary in order to access the bfdServer component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
bfdServer select 1 5 2
```

will access the BFD server for chassis 1, card 5, and port 2.

This command holds a list of the simulated routers. The definition of these routes occurs using the *bfdRouter* command and subsidiary commands. The important subcommands of this command are:

bfdServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.

Member	Usage
clearAllRouters	Removes all routers from the list of routers.
addRouter	Adds a router to the list of routers. The router must have been previously configured through the use of the *bfdRouter* command.
getRouter getFirstRouter getNextRouter	Accesses a particular router from the list, either directly by ID or by iterating through all of the routers. The data appears in the *bfdRouter* command.
delRouter	Deletes a particular router from the list.
setRouter	It is possible to change BFD router configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the router's configuration with *bfdRouter.* (*bfdRouter* has capabilities for modifying elements of underlying objects as well). 2. Use the *bfdServer bfdRouter* command to set the values from *bfdRouter* into IxHal. 3. Use the *bfdServer write* command to write the changes to the hardware.
showRouterNames	Returns names of routers in the list on the selected routers. Calling the *select* command getting the bridge is recommended before calling this command.
write	Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card, and port. Before using this command, use the *bfdServer select* command to select the port.

bfdRouter

Refer to NAME - bfdRouter for a complete description of this command. The *bfdRouter* command represents a simulated router. In addition to some identifying options, it holds one list for the router:

- Interfaces — router interface, constructed in the bfdInterface command.

bfdRouter Options

Member	Usage
enable	Enables or disables the simulated router.
routerId	The ID of the simulated router, expressed as an IP address.

bfdRouter Subcommands

Class	Member	Usage
Interface	setDefault	Sets the options to the default values.
	clearAllInterfaces	Clears the BFD interface list on the selected router.
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the bfdRouter command.
	delInterface	Deletes a particular interface from the interface list of a selected router.
	getInterface getFirstInterface	Accesses a particular interface either by ID or by iterating through all of the interfaces. The data appears in

Class	Member	Usage
	getNextInterface	the bfdInterface command.
	setInterface	It is possible to change interface configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with bfdRouter. 2. Use the *bfdRouter bfdInterface* command to set the values from bfdInterfaceinto IxHal. 3. Use the bfdInterface *write* command to write the changes to the hardware.
	showInterfaceName	Returns names of interfaces in the list on the selected router. Calling the *select* command getting the bridge is recommended before calling this command.
Interface Learned Info	requestLearnedInfo	Request for learned info for all BFD sessions under this interface.
	getLearnedInfoList getFirstLearnedInfo getNextLearnedInfo	Accesses a particular learned info list from the BFD router list on the selected BFD router. The data appears in the *bfdLearnedI Info* command.

bfdInterface

Refer to NAME - bfdInterface for a full description of this command. The bfdInterface holds the information related to a single interface on the simulated router. Interfaces are added into the bfdRouter interface list using the bfdRouter addInterface command. In addition, learned LSAs from the DUT are made available through this command.

This command holds a list of the simulated interfaces. The definition of these interfaces occurs using the bfdInterface command and subsidiary commands. The important options and subcommands of this command are:

bfdInterface Options

Member	Usage
enable	Enables the use of the simulated interface.
interfaceId	This is a local ID and is unique per router.
desiredMinRxIntv	This option indicates the desired minimum interval between received BFD control packets.*(default = 1,000)*
desiredTxIntv	This option indicates the desired interval between transmitted BFD control packets.*(default = 1,000)*
desiredEchoRxIntv	This option indicates the minimum interval between received BFD Echo packets that this interface is capable of supporting. If this value is zero, the transmitting system does not support the receipt of BFD Echo packets.
echoTxIntv	This option indicates the minimum interval that the interface would like to use when transmitting BFD Echo packets.
echoTimeOut	This is the minimum interval that the interface waits for a response to the last Echo packet sent out.

Member	Usage
echoConfigureSrcIp	If set to *True* (1), the configure Source IP Address option is enabled, and an IPv4 or IPv6 Source Address can be configured for an Echo packet.
echoSrcIPv4Addr	If *echoConfigureSrcIp* is enabled, this option is available for configuring an IPv4 Source Address for an Echo packet.
echoSrcIPv6Addr	If *echoConfigureSrcIp* is enabled, this option is available for configuring an IPv6 Source Address for an Echo packet.
multiplier	Multiplier * intv defines the timeout period. *(default = 3)*
flapTxIntvs	BFD sessions will flap every flapTxIntvs. *(default = 0)*
pollIntv	If in the Demand Mode, polling will take place every pollIntv interval. *(default = 1,000)*
enableDemandMode	Enables demand mode. 1 indicates demand mode enabled, and 0 indicates demand mode disabled.
enableCtrlPlaneIndependent	Set to 1 if the local system's BFD implementation is independent of the control plane.
protocolInterfaceDescription	The name of the defined *interfaceEntry* which describes the host interface to be simulated.

bfdInterface Subcommands

Member	Usage
addSession	Adds a session to the session list of an interface.
getSession getFirstSession getNextSession	Accesses a particular session from the BFD session list on the selected BFD interface. The data appears in the *bfdSession* command.
delSession	Deletes a particular session from the session list.
setSession	Allows the configuration values for a session to be overwritten on the fly.
setDefault	Selects the chassis, card, and port to operate on.
clearAllSessions	Clears the BFD session list on the selected BFD interface.
showSessionNames	Returns the names of the sessions in the list on the selected port. Calling *select* command is recommended before calling this command.

bfdSession

Refer to NAME - bfdSession for a complete description of this command. The *bfdSession* command holds the information related to a single session under an interface for a simulated BFD router. Sessions are added under the *bfdInterface* using the *bfdInterface addSession* command. The important options and subcommands of this command are:

bfdSession Options

Member	Usage
enable	Enables the use of this route range for the simulated router. The default is disable.
IpType	The session is created with the remote IP. IPv4 or IPv6 (*default = IPv4*).
remoteDiscLearned	The default is 0. If it is set to 0, then the Remote Discriminator will

Member	Usage
	be learned.
myDisc	Needs to be a unique value in node. This option is used to demultiplex multiple BFD sessions.
name	(*Read only*) The name of the session that will be used as a unique key to retrieve the object.
localBFDAddress	The first IP address that will be used for simulated routers. IPv4 or IPv6.
remoteBFDAddress	The remote address in which the BFD session is active.
remoteDisc	This is the discriminator used by the remote system to identify the BFD session. This must be initialized to zero.
enableAutoChooseSrc	If true, enables the session to automatically choose the source IP address for the BFD session.
sessionType	Indicates whether the mode is a single-hop session or a multihop session.

bfdSession Subcommand

Member	Usage
setDefault	Sets the options to default values.

bfdSessionLearnedInfo

Refer to NAME - bfdSessionLearnedInfo for a full description of this command. The *bfdSessionLearnedInfo* command fetches and describes the learned BFD session information for the current interface defined in bfdSessionLearnedInfo. The important options and subcommands of this command are:

bfdSessionLearnedInfo Options

Member	Usage
desMinTxIntv	(*Read-only*) Indicates the desired interval (in ms) between transmitted BFD control packets received from the peer.
myDisc	(*Read-only*) Identifies the session uniquely. This option is used to demultiplex multiple BFD sessions.
myIpAddress	*(Read-only)* The local IP address being used by the configured or auto-created BFD session.
peerDisc	(*Read-only*) The discriminator received from the peer. This field reflects back the received value of myDisc.
peerFlags	(*Read-only*) The peer flags are received (0x02 = Demand Mode, 0x04 = Authentication, 0x08 = Control Plane Independent, 0x10 = Final, 0x11 = Poll).
peerIPAddress	(*Read-only*) Indicates whether the peer is for IPv4 or IPv6.
peerState	(*Read-only*) The peer state is received from the peer (0 AdminDown, 1 Down, 2 Init, 3 Up).
peerUPtime	(*Read-only*) The time since the session last went to UP state.
protosUsingSession	(*Read only*) The protocols registered for this session. Containing one or more of the following protocols: • None

Member	Usage
	• BGP • OSPF • OSPFv3 • EIGRP • ISIS
reqMinEchoIntv	(*Read-only*) Indicates the minimum interval (in ms) between received BFD echo packets.
reqMinRxIntv	(*Read-only*) Indicates the minimum interval (in ms) between received BFD control packets.
sessionType	(*Read-only*) 0 indicates one hop and 1 indicates multihops.

bfdSessionLearnedInfo Subcommand

Member	Usage
setDefault	Sets the options to default values.

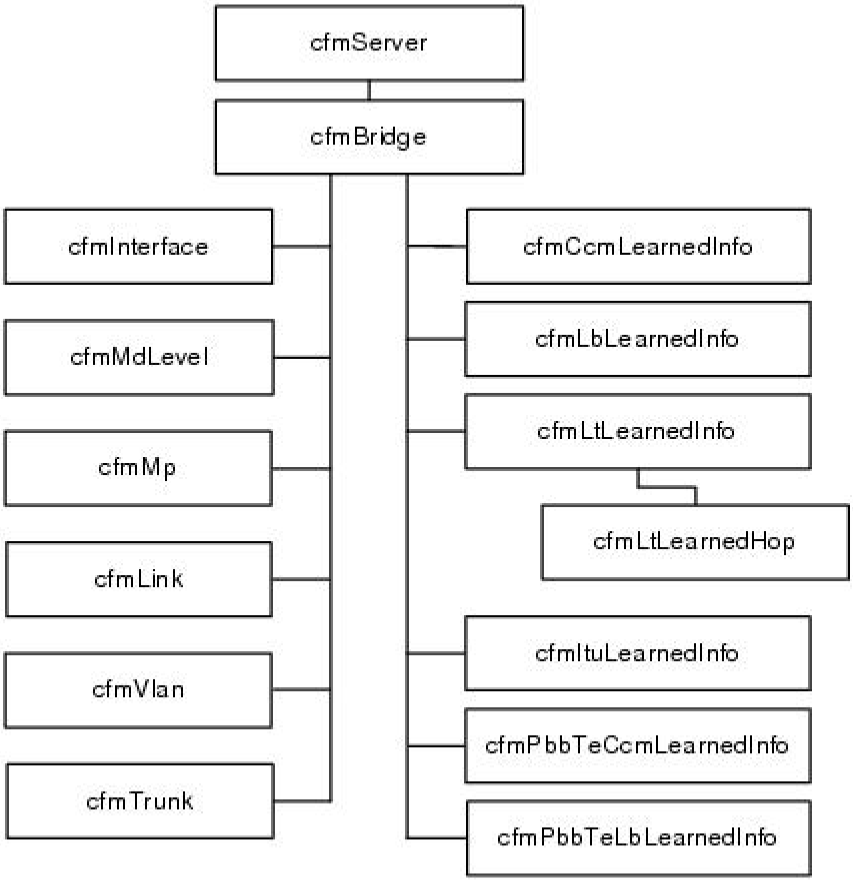
CFM

Please refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion of the Ixia protocol server's testing model with respect to CFM. The CFM protocol is used to monitor the connectivity across multiple Layer 2 network hops. The CFM-related commands are:

- cfmServer — provides access to the CFM part of a port's protocol server.
- cfmBridge — a container used to hold lists associated with the bridge: interfaces, MD levels, MPs, Links, VLANs, Trunks, and learned info.
- cfmInterface — a network interface, which will be included in cfmBridge.
- cfmMdLevel — configures Maintenance Domain Levels associated with the bridges.
- cfmMp — configures MIPs/MEPs associated with the bridges.
- cfmLink — configures Links associated with the bridges.
- cfmVlan — configures VLANs associated with the bridges.
- cfmTrunk — configures PBB-TE Trunks associated with the bridges.
- cfmCcmLearnedInfo — (read-only) view the learned CCM database for the respective bridge.
- cfmAisLearnedInfo — (read-only) view the learned AIS database for the respective bridge.
- cfmLckLearnedInfo — (read-only) view the learned LCKdatabase for the respective bridge.
- cfmTstLearnedInfo — (read-only) view the learned TST database for the respective bridge.
- cfmLbLearned Info — (read-only) view the learned LoopBack (LB)/Ping information for the respective bridge.
- cfmLtLearned Info — (read-only) view the learned Link Trace (LT) information for the respective bridge.
- cfmLtLearned Hop — (read-only) view the learned Link Trace (LT) Learned Hop information.
- cfmItuLearnedInfo — (read-only) view the learned Delay Measurement information for the respective bridge (applies to both ITU Y.1731 and PBB-TE).
- cfmPbtCcmLearnedInfo — (read-only) view the learned PBB-TE CCM database for the respective bridge.
- cfmPbtLbLearnedInfo — (read-only) view the learned PBB-TE LoopBack (LB)/Ping information for the respective bridge.

These commands and the data that they maintain are arranged in a hierarchy, as shown in the following figure.

CFM Command Hierarchy



cfmServer

Refer to NAME - cfmServer for a complete description of this command. The *cfmServer* command is necessary in order to access the cfmServer component of the protocol server for a particular port. The *select* subcommand **must** be used before all others in this category. For example,

```
cfmServer select 1 5 2
```

will access the CFM server for chassis 1, card 5, and port 2.

This command holds a list of the simulated bridges. The definition of these bridges occurs using the *cfmBridges* command and subsidiary commands. The important options and subcommands of this command are:

cfmServer Options

Member	Usage
enableOptionalTlvValidation	Enables the validation of Optional TLVs. *(True/False)*
receiveCcm	Enables the receipt of Continuity Check Messages (CCMs). *(True/False)*
sendCcm	Enables the transmission of Continuity Check Messages (CCMs). *(True/False)*

cfmServer Subcommands

Member	Usage
select	Selects the chassis, card, and port to operate on.

Member	Usage
clearAllBridges	Removes all bridges from the list of bridges.
addBridge	Adds a bridge to the list of bridges. The bridge must have been previously configured through the use of the *cfmBridge* command.
get	Gets the current configuration of the protocol server for the last selected port from its hardware. Call this command before calling *cfmServer* cget *option value* to get the value of the configuration option.
getBridge getFirstBridge getNextBridge	Accesses a particular bridge from the list, either directly by ID or by iterating through all of the bridges. The data appears in the *cfmBridge* command.
delBridge	Deletes a particular bridge from the list.
set	Sets the current configuration of the protocol server on the most recently selected port to its hardware. Call this command before calling *cfmServer* cget *option value* to get the value of the configuration option.
setBridge	It is possible to change CFM bridge configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the bridge's configuration with *cfmBridge.* (*cfmBridge* has capabilities for modifying elements of underlying objects as well). 2. Use the *cfmServer cfmBridge* command to set the values from *cfmBridge* into IxHal. 3. Use the *cfmServer write* command to write the changes to the hardware.
showBridgeNames	Returns names of bridges in the list of the selected bridges. Calling the *select* command getting the bridge is recommended before calling this command.
write	Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card, and port*.* Before using this command, use the *cfmServer select* command to select the port.

cfmBridge

Refer to NAME - cfmBridge for a complete description of this command. The *cfmBridge* command represents a simulated bridge. In addition to some identifying options, it holds the following lists for the bridge:

- Interfaces — CFM interfaces.
- MD Levels — Maintenance Domain Levels.
- MPs — Maintenance Points.
- Links — Links between CFM Maintenance Points.
- VLANs — Associated VLANs.
- Trunks — CFM Point-to-Point Trunks, used with Provider Backbone Bridging Traffic Engineering (PBB-TE).
- CCM Learned Info — Information learned from the CFM Continuity Check Messages (CCMs).

- LB Learned Info — Information learned from the CFM LoopBack (LB)/Ping messages.
- LT Learned Info — Information learned from the Link Trace (LT) messages.
- LT Learned Hop — Link Trace Learned Hop information.
- ITU (Delay) Learned Info — Learned information concerning frame delays (Applies to Y.1731 (ITU) and PBB-TE).
- PBB-TE CCM Learned Info — Information learned from the PBB-TE CCMs.
- PBB-TE LB Learned Info — Information learned from the PBB-TE LoopBack (PB)/Ping messages.

cfmBridge Options

Member	Usage
enable	Enables or disables the emulated CFM bridge. *(True / False)*
bridgeId	The ID of the emulated CFM bridge, expressed as a 6-octet MAC address.
operationMode	The mode of operation for the emulated CFM bridge. One of: • cfm (IEEE 802.1ag) (default) • y1731 (ITU) • pbbTe (PBB-TE)
function	The CFM function for the operation mode. One of: • faultManagement (default) • performanceMeasurement (available for Y.1731 only)
aisInterval	The interval between messages with Alarm Indication Signal (AIS) information. One of: • oneSec one second *(Default)* • oneMin one minute
encapsulation	The CFM bridge encapsulation type. Select one of: • ethernet (default) • llcSnap (LLC-SNAP)
enableAis	Enables or disables the transmission of Alarm Indication Signal messages. (Default = disabled)
etherType	The value for the EtherType to be used for the bridge. (Integer)
enableOutOfSequenceDetection	Enables or disables Out of Sequence error detection. Allows the bridge to detect when CCMs are out of sequence.
name	*(Read-only)* Indicates the name of the bridge which will be used as a unique key to retrieve the object.
User Input Required For Learned Info	
userBvlan	For use with PBB-TE. User selection for the B-VLAN, to filter on. One of: • noVlanId

Member	Usage
	- vlanId - allVlanId
userBvlanId	For use with PBB-TE. The identifier for this B-VLAN, to filter on. (Integer)
userBvlanPriority	For use with PBB-TE. The priority for the B-VLAN, to to filter on. Range: 0 to 7. (Default = 0)
userBvlanTpId	For use with PBB-TE. Value for the B-VLAN Tag Protocol ID, to filter on. (Integer)
userCvlan	User selection for the C-VLAN, to filter on. One of: - noVlanId - vlanId - allVlanId
userCvlanId	The identifier for this C-VLAN, to filter on. (Integer)
userCvlanTpId	Value for the C-VLAN Tag Protocol ID, to filter on. (Integer)
userCvlanPriority	The priority for the C-VLAN, to filter on. Range: 0 to 7. Default = 0.
userDelayMethod	The type of delay mesurement method to filter on. One of: - 1-oneWay - 0- twoWay **Note:,**for One-way, if DM is selected then one 1DM will be sent, if DVM is selected then 2 1DM will be sent one by one.
userDelayType	The type of Delay Measurement to filter on. One of: - dm (Delay measurement) - dvm (Delay variation measurement)
userDstMacAddress	Value for the Destination MEP MAC address filter. (6-octet MAC address)
userDstMepId	Value for the Destination MEP ID filter. (Integer)
userLearnedInfoTimeOut	The interval, in milliseconds (ms), for the learned record to timeout. (Integer) (Default = 5000)
userMdLevel	The Maintenance Domain (MD) Level to filter on. One of: - zeroMd - oneMd - twoMd - threeMd - fourMd - fiveMd - sixMd - sevenMd

Member	Usage
	- allMd
userPbbTeDelayMethod	The type of delay mesurement method. One of: - 1- oneWay - 0-twoWay **Note:,**for One-way, if DM is selected then one 1DM will be sent, if DVM is selected then 2 1DM will be sent one by one.
userPbbTeDelayType	For use with PBB-TE. The type of Delay Measurement. One of: - dm (Delay measurement) - dvm (Delay variation measurement)
userSelectDstMepById	Enable or disable Select by ID filter for Destination MEP. *(True / False)*
userSelectSrcMepById	Enable or disable Select by ID filter for Source MEP. *(True / False)*
userSendType	The type of loopback. One of: - unicast (only type available for CFM) - multicast (available for Y.1731)
mepId	The MEP identifier of the CCM message.
mipId	The MIP identifier.
mpType	Sets the MP type. Possible values include: - mip = *Maintenance Intermediate Point.* - mep = *Maintenance Point.*
macAddress	The MAC address of the MP.
cciInterval	Sets the Continuity Check Interval (CCI). Possible values include: - 3.33msec - 10msec - 100msec - 1sec - 10sec - 1min - 10min
userShortMaName	The Short MA Name to filter on. (String)
userShortMaNameFormat	Selection for the Short MA Name Format filter. One of: - allFormats - primaryVid - characterString - twoOctetInteger - rfc2685VpnId

Member	Usage
rdi	The Remote Defect Identification. Possible values include: • auto • on • off
userSrcMacAddress	A 6 -octet MAC address for the source MAC address.
userSvlan	Selection for the C-VLAN filter. One of: • noVlanId • vlanId • allVlanId
userSvlanId	The identifier for this S-VLAN, to filter on. (Integer)
userSvlanPriority	The priority for the S-VLAN, to filter on. Range: 0 to 7. (Default = 0)
userSvlanTpId	Value for the C-VLAN Tag Protocol ID, to filter on. (Integer)
userTtlInterval	The Time To Live value, in seconds. (Integer) (Default = 64)
userTransactionId	The Transaction identifier (sequence number) sent in the CFM message, if the configured MEP is not found. (Integer)
(Read-only)	
delayLearnedError String	Delay Learned Error string.
isDelayLearnedConfig Mep	*True* means that a configured MEP was found. *True / False.*
isDelayLearnedPacket Sent	*True* means that a packet was sent. *True / False.*
isLbLearnedConfigMep	*True* means that a configured MEP was found. *True / False.*
isLbLearnedPacketSent	*True* means that a packet was sent. *True / False.*
isLtLearnedConfigMep	*True* means that a configured MEP was found. *True / False.*
isLtLearnedPacketSent	*True* means that a packet was sent. *True / False.*
isPbbTeDelayLearned ConfigMep	*True* means that a configured MEP was found. *True / False.*
isPbbTeDelayLearned PacketSent	*True* means that a packet was sent. *True / False.*
isPbbTeLbLearned ConfigMep	*True* means that a configured MEP was found. *True / False.*
isPbbTeLbLearned PacketSent	*True* means that a packet was sent. *True / False.*
isPeriodicOamLearned InfoRefreshed	*True* means that the periodic OAM information is up to date. *True / False.*

Member	Usage
lbLearnedErrorString	LoopBack (LB)/Ping Learned Error string.
ltLearnedErrorString	LinkTrace (LT) Learned Error string.
pbbTeDelayLearned ErrorString	PBB-TE Delay Learned Error string.
pbbLbLearnedError String	PBB-TE LoopBack (LB)/Ping Learned Error string.

cfmBridge Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.
Interface	clearAllInterfaces	Clears the CFM interface list on the selected bridge.
	addInterface	Adds a new interface. The interface must have been previously configured through the use of the cfmInterface command.
	delInterface	Deletes a particular interface from the interface list of a selected bridge.
	getInterface getFirstInterface getNextInterface	Accesses a particular interface either by ID or by iterating through all of the interfaces. The data appears in the cfmInterface command.
	setInterface	It is possible to change interface configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the interface's configuration with cfmBridge. 2. Use the *cfmBridge cfmInterface* command to set the values from cfmInterface into IxHal. 3. Use the cfmInterface *write* command to write the changes to the hardware.
	showInterfaceName	Returns names of interfaces in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.
MD Levels	clearAllMdLevels	Clears the CFM MD Level list on the selected bridge.
	addMdLevel	Adds a new MD Level. The MD

Class	Member	Usage
		Level must have been previously configured through the use of the cfmMdLevel command.
	delMdLevel	Deletes a particular MD Level from the MD Level list of a selected bridge.
	getMdLevel getFirstMdLevel getNextMdLevel	Accesses a particular MD Level either by ID or by iterating through all of the MD Levels. The data appears in the cfmMdLevel command.
	setMdLevel	It is possible to change MD Level configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the MD Level's configuration with cfmBridge. 2. Use the *cfmBridge cfmMdLevel* command to set the values from cfmMdLevel into IxHal. 3. Use the cfmMdLevel *write* command to write the changes to the hardware.
	showMdLevelNames	Returns names of MD Levels in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.
MP	clearAllMps	Clears the CFM MP list on the selected bridge.
	addMp	Adds a new MP. The MP must have been previously configured through the use of the cfmMp command.
	delMp	Deletes a particular MP from the MP list of a selected bridge.
	getMp getFirstMp getNextMp	Accesses a particular MP either by ID or by iterating through all of the MPs. The data appears in the cfmMp command.
	setMp	It is possible to change MP configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the MP's configuration

Class	Member	Usage
		with cfmBridge. 2. Use the *cfmBridge cfmMp* command to set the values from cfmMp into IxHal. 3. Use the cfmMp *write* command to write the changes to the hardware.
	showMpNames	Returns names of MPs in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.
Link	clearAllLinks	Clears the CFM Link list on the selected bridge.
	addLink	Adds a new Link. The Link must have been previously configured through the use of the cfmLink command.
	delLink	Deletes a particular Link from the Link list of a selected bridge.
	getLink getFirstLink getNextLink	Accesses a particular Link either by ID or by iterating through all of the Links. The data appears in the cfmLink command.
	setLink	It is possible to change Link configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the Link's configuration with cfmBridge 2. Use the *cfmBridge cfmLink* command to set the values from cfmLink into IxHal. 3. Use the cfmLink *write* command to write the changes to the hardware.
	showLinkNames	Returns names of Links in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.
VLAN	clearAllVlans	Clears the CFM VLAN list on the selected bridge.
	addVlan	Adds a new VLAN. The VLAN must have been previously configured through the use of the cfmVlan

Class	Member	Usage
		command.
	delVlan	Deletes a particular VLAN from the VLAN list of a selected bridge.
	getVlan getFirstVlan getNextVlan	Accesses a particular VLAN either by ID or by iterating through all of the VLAN. The data appears in the cfmVlan command.
	setVlan	It is possible to change VLAN configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the Vlan's configuration with cfmBridge. 2. Use the *cfmBridge cfmVlan* command to set the values from cfmVlan into IxHal. 3. Use thecfmVlan *write* command to write the changes to the hardware.
	showVlanNames	Returns names of VLANs in the list on the selected CFM bridge. Calling the *select* command getting the bridge is recommended before calling this command.
Trunk	clearAllTrunks	Clears the CFM Trunk list on the selected bridge.
	addTrunk	Adds a new Trunk. The Trunk must have been previously configured through the use of the cfmTrunk command.
	delTrunk	Deletes a particular Trunk from the Trunk list of a selected bridge.
	getTrunk getFirstTrunk getNextTrunk	Accesses a particular Trunk either by ID or by iterating through all of the Trunks. The data appears in the cfmTrunk command.
	setTrunk	It is possible to change Trunk configuration on the fly. In order to do this, the following steps are necessary: 1. Modify the Trunk's configuration with cfmBridge. 2. Use the *cfmBridge cfmTrunk* command to set the values from cfmTrunk into IxHal. 3. Use the cfmTrunk *write* com-

Class	Member	Usage
		mand to write the changes to the hardware.
	showTrunkNames	Returns names of Trunks in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.
Custom TLV	addCustomTlv delCustomTlv getCustomTlv setCustomTlv getFirstCustomTlv getNextCustomTlv clearAllCustomTlvs	Adds and removes custom TLVs from the CFM bridge. Also, accesses custom TLV information from the bridge's learned information.
CCM Learned Info (common to CFM/ITU/PBB-TE)	requestCcmLearnedInfo	Requests all CCM learned info for this bridge.
	getCcmLearnedInfoList getFirstCcmLearnedInfo getNextCcmLearnedInfo	Accesses a particular learned info entry in the CCM Learned Info list on the selected CFM Bridge. The data appears in the *cfmCcmLearnedInfo* command.
Ping/Loopback (LB) Learned Info (common to CFM/ITU/PBB-TE)	startLoopback	Starts Ping/Loopback (LB) to acquire Ping/Loopback learned info for this bridge.
	getLoopbackLearnedInfoList getFirstLoopbackLearnedInfo getNextLoopbackLearnedInfo	Accesses a particular learned info entry in the LB Learned Info list on the selected CFM Bridge.
Link Trace Learned Info (common to CFM/ITU)	startLinkTrace	Starts Link Trace (LT) to acquire Link Trace learned info for this bridge.
	getLinkTraceLearnedInfoList getFirstLinkTraceLearnedInfo getNextLinkTraceLearnedInfo	Accesses a particular learned info entry in the LT Learned Info list on the selected CFM Bridge.
Delay Measurement Learned Info	startDelayMeasurement	Starts Delay Measurement to acquire Delay learned info for this bridge.

Class	Member	Usage
(common to ITU/PBB-TE)		
	getDelayMeasurement LearnedInfo getDelayMeasurement LearnedInfo getDelayMeasurement LearnedInfoList getFirstDelayMeasurementLearnedInfo getNextDelay MeasurementLearnedInfo	Accesses the learned info in the Delay Learned Info list on the selected CFM Bridge. The data appears in the *cfmDelayLearnedInfo* command.
Periodic OAM	requestPeriodicOam LearnedInfo getPeriodicOamLearned InfoList getFirstPeriodicOam LearnedInfo getNextPeriodicOam LearnedInfo	Accesses the learned info in the Periodic OAM Learned Info list on the selected CFM Bridge.

cfmInterface

Refer to NAME - cfmInterface for a full description of this command. The *cfmInterface* holds the information related to a single interface on the simulated bridge. Interfaces are added into the cfmBridge interface list using the cfmBridge *addInterface* command. The important options and subcommands of this command are:

cfmInterface Options

Member	Usage
enabled	Enables the use of the simulated interface. *(True/False)*
interfaceId	This is a local ID and is unique per bridge. (This attribute references an object of *Interface*.)
protocolInterfaceDescription	The name of the defined entry which describes the interface to be simulated. (String)
name	*(Read-only)* The name of the interface which will be used as a unique key to retrieve the object.

cfmInterface Subcommands

Member	Usage
setDefault	Sets the options to the default values.

cfmMdLevel

Refer to NAME - cfmMdLevel for a complete description of this command. The *cfmMdLevel* command represents a simulated Maintenance Domain (MD) Level.

cfmMdLevel Options

Member	Usage
enabled	Enables or disables the simulated MD Level. *(True/False)*
mdName	The name of the MD, based on the selection for the *mdNameFormat* (below). (String)
mdNameFormat	The naming format for each level instance. Options include the following: • set enumList [list cfmNoNamePresent cfmDomainNameString cfmMACAddressPlus2OctetInt cfmMANNameCharString] • setEnumValList $enumList enumValList • set enumsArray(cfmMdLevel,mdNameFormat) $enumValList
mdLevelId	The Level available on the bridge. Depending on the configuration, this can be a number from 0 to 7.
name	*(Read-only)* The name of the MD Level which will be used as a unique key to retrieve the object.

cfmMdLevel Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmMp

Refer to NAME - cfmMp for a complete description of this command. The *cfmMp* command represents a simulated Maintenance Point (MP).

cfmMp Options

Member	Usage
enabled	Enables or disables the simulated MP on this bridge. *(True/False)*
addCcmCustomTlvs	If true, adds a custom CCM TLV to messages.
addDataTlv	This adds a data TLV to messages. This TLV is applicable for LBM/LBR.
addLbmCustomTlvs	If true, adds a custom LBM TLV to messages.
addLbrCustomTlvs	If true, adds a custom LBR TLV to messages.
addLtmCustomTlvs	If true, adds a custom LTM TLV to messages.
addLtrCustomTlvs	If true, adds a custom LTR TLV to messages.
addInterfaceStatusTlv	If true adds an interface status TLV to messages. We do not allow user to change value of Interface Status TLV from here. However the user can always add Interface Status TLV as an optional TLV in Bridge and edit value. This TLV is applicable for CCM.

Member	Usage
	Default is true.
addOrganization SpecificTlv	If true, adds an organization specific TLV to messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.
addPortStatusTlv	If true, adds a port status TLV to messages. We do not allow user to change value of Port Status TLV from here. However user can always add Port Status TLV as an optional TLV in Bridge and edit value. This TLV is applicable for CCM. The default is true.
addSenderIdTlv	If true, adds a Sender TLV to PBB-TE messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.
autoDmIteration	The count for how many times DMMs will be transmitted. Default is 0 (no limit). Min: 0 Max: 2^32
autoDmTimeout	The timeout period in seconds to wait for a response to DMMs. This value should be less than the Auto LB Timer. Default is 30. Min: 1 Max: 65535
autoDmTimer	The time period in seconds between DMMs. Default is 60. Min: 1 Max: 65535
autoLbIteration	The count for how many times LBM will be transmitted. Default is 0 (no limit). Min: 0 Max: 2^32
autoLbTimeout	The timeout period in seconds to wait for a response to LBMs. This value should be less than the Auto LB Timer. Default is 30. Min: 1 Max: 65535
autoLbTimer	The time period in seconds between LBMs. Default is 60. Min: 1 Max: 65535
autoLtIteration	The count for how many times LTM will be transmitted. Default is 0 (no limit). Min: 0 Max: 2^32

Member	Usage
autoLtTimeout	The timeout period in seconds to wait for a response to LTMs. This value should be less than the Auto LT Timer. Default is 30. Min: 1 Max: 65535
autoLtTimer	The time period in seconds between LTMs. Default is 60. Min: 1 Max: 65535
cciInterval	The configured time between CCM transmissions. (Integer) (Default = 1 sec) One of: • cci3msec (3 milliseconds) • cci10msec (10 milliseconds) • cci100msec (100 milliseconds) • cci1sec (Default 1 second) • cci10sec (10 seconds) • cci1min (1 minute) • cci10min (10 minutes)
ccmPriority	Sets the priority for Continuity Check Messages. The default is 0. Min: 0 Max: 7
chassisId	Sets the Chassis identifier. Default is 00 00 00 00 00 00. This will take Hex value as input (0-255 byte).
chassisIdLength	Sets the length of the chassis identifier. Default is 6. Min: 0 Max: 255.
chassisIdSubType	Sets the chassis identifier sub-type for the optional TLV messages. Options are: • chassisComponent • interfaceAlias • portComponent • chassisMacAddress • networkAddress • interfaceName • locallyAssigned
dataTlvLength	Sets the length of the Data TLV. Default is 4. Min: 0 Max: 1500.

Member	Usage
dataTlvValue	This column will take Hex value of data. This data TLV will be added both for periodic LBM and requested LBM transmit. Default is 44 61 74 61.
dmMethod	Sets the delay mesaurement method. The available options are: • oneWay • twoWay
dmPriority	Sets the priority for DM Messages. This priority will be used only for periodic DMMs. The default is 0. Min: 0 Max: 7 **Note:** Backward compatibility is maintained for the legacy `dmmPriority' attribute.
enableAutoDm	If true, enables the automatic sending to DM Messages.
enableAutoLb	If true, enables the automatic sending to Loopback Messages.
enableAutoLt	If true, enables the automatic sending of Link Trace Messages.
lbmPriority	Sets the priority for Loopback Messages. This priority will be used only for periodic LBMs. The default is 0. Min: 0 Max: 7
ltmPriority	Sets the priority for Link Trace Messages. This priority will be used only for periodic LTMs. The default is 0. Min: 0 Max: 7
macAddress	The 6-octet MAC Address of the MP.
managementAddress	Sets the Managment Address. Input type is HEX (0-255 byte). Default is 01 02 03 03 04 05.
managementAddress Domain	Sets the Management Address Domain. This will take HEX input (0-255 byte). Default is 4d 61 6e 61 67 65 6d 65 6e 74 20 41 64 64 72 20 44 6f 6d 61 69 6e ("Management Addr Domain").
managementAddress DomainLength	Sets the length of the Management Address domain. Default is 22. Min: 0 Max: 255.
managementAddress	Sets the length of the Managment Address.

Member	Usage
Length	Default is 6. Min: 0 Max: 255.
mdLevel	The MD or MEG level assigned to the MP. (This attribute references an object of *cfmMdLevel*.) (String)
megId	The identifier of the Maintenance Entity Group (MEG). (For use with Y.1731.) The base of this depends on the *megIdFormat* selection (below). (String)
megIdFormat	Sets the format for the megId (for use with Y.1731). The only option is *iccBasedFormat*.
mepId	The number that is used to identify the Maintenance End Point. (Integer)
mpType	The type of Maintenance Point. One of: • cfmMIP (Maintenance Intermediate Point) • cfmMEP (Maintenance End Point)
organizationSpecificTlv Length	Sets the length for the Organizational TLV. Default is 4. Min: 4 Max: 1500
organizationSpecificTlvValue	Sets the value for the Organizational TLV. Default is NULL.
overrideVlanPriority	If true, overrides the set VLAN priority for this bridge, and uses the advanced settings instead. (true/false)
rdi	The Remote Defect Identification. Possible values include: • auto • on • off
shortMaName	The short name of the MA. The base of this depends on the selection for shortMaNameFormat (below). (String)
shortMaNameFormat	Sets the format for the short MA Name. One of: • primaryVid • characterString • twoOctetInteger • rfc2685VpnId
ttl	Sets the Time To Live for the period OAM. Default is 64. Min: 1 Max: 255
vlan	The VLAN assigned to the MP. (This attribute references an object of *cfmVlan*.) (String)
vlanLocalId	*(Read-only)* The VLAN Local Id. (Integer)
mdLevelLocalId	*(Read-only)* The MD Level Local ID. (Integer)

Member	Usage
name	*(Read-only)* The name of the MP which will be used as a unique key to retrieve the object.
aisEnableUnicastMac	If true, enables Ais unicast MAC address.
aisInterval	Sets the Ais interval.
aisMode	Indicates the Ais configuration mode.
aisPriority	Indicates the Ais priority value.
aisUnicastMac	Indicates ihe Ais unicast MAC address.
enableAisRx	If true, enables the Ais receiver port.
enableLckRx	If true, enables the Lck receiver port.
enableTstRx	If true, enables the Tst receiver port.
lckEnableUnicastMac	If true, enables Lck unicast MAC address.
lckInterval	Sets the Lck interval.
lckMode	Indicates the Lck configuration mode.
lckPriority	Indicates the Lck priority value.
lckSupportAisGeneration	Indicates Lck support for Ais generation.
lckUnicastMac	Indicates the Lck unicast MAC address.
tstEnableUnicastMac	If true, enables Tst unicast MAC address.
tstIncrPacketLength	Increments the Tst packet size, including the padding length.
tstIncrPacketLengthStep	Increments the Tst packet size, including the padding length by step.
tstInitialPatternValue	Indicates the initial value of Tst pattern.
tstInterval	Sets the Tst interval.
tstMode	Indicates the Tst configuration mode.
tstOverwriteSequenceNumber	Overwrites the Tst sequence number.
tstPacketLength	Indicates the Tst packet size, including the padding length.
tstPatternType	Indicates the type of Tst data pattern.
tstPriority	Indicates theTst priority value.
tstSequenceNumber	(read only) Indicates the sequence number of Tst.
tstTestType	Indicates the type of Tst test.
tstUnicastMac	Indicates the Ais unicast MAC address.

cfmMp Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmLink

Refer to NAME - cfmLink for a complete description of this command. The *cfmLink* command represents a simulated Link.

cfmLink Options

Member	Usage
enabled	Enables or disables the simulated Link on the bridge. *(True/False)*
mpTowardsIxia	Specifies the Maintenance Point (MP) for this link that is facing the chassis. (This attribute references an object of *MP*.)
linkType	Determines the link type of the link. One of: • PointToPoint • broadcast
moreMps	This option is only available for broadcast links. Broadcast links can have multiple MPs. (This attribute references a list of objects of *cfmMp*.) (List of objRef of *cfmMps*)
mpOutwardsIxia	Specifies the MP(s) for this link that is outbound from the chassis: • For the point-to-point link, corresponds to *cfmMp* outbound from the chassis (for the connected MP). This attribute references an object of *chmMp*. (objRef of *cfmMp*) • For shared (broadcast) links, corresponds to *moreMps* for the shared links outbound from the chassis to additional connected MPs. This attribute references a list of objects of *chmMp.* (List of objRef of *cfmMp*)
rightMpLocalId	*(Read-only)* The right MP Local ID. (String)
leftMpLocalId	*(Read-only)* The left MP Local ID. (String)
name	*(Read-only)* The name of the link which will be used as a unique key to retrieve the object.

cfmLink Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmMdLevel

Refer to NAME - cfmMdLevel for a complete description of this command. The *cfmMdLevel* command represents a simulated Maintenance Domain (MD) Level.

cfmMdLevel Options

Member	Usage
enabled	Enables or disables the simulated MD Level. *(True/False)*
mdName	The name of the MD, based on the selection for the *mdNameFormat* (below). (String)
mdNameFormat	The naming format for each level instance. Options include the following: • set enumList [list cfmNoNamePresent cfmDomainNameString

Member	Usage
	cfmMACAddressPlus2OctetInt cfmMANNameCharString] - setEnumValList $enumList enumValList - set enumsArray(cfmMdLevel,mdNameFormat) $enumValList
mdLevelId	The Level available on the bridge. Depending on the configuration, this can be a number from 0 to 7.
name	*(Read-only)* The name of the MD Level which will be used as a unique key to retrieve the object.

cfmMdLevel Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmVlan

Refer to NAME - cfmVlan for a complete description of this command. The *cfmVlan* command represents a simulated CFM VLAN.

cfmVlan Options

Member	Usage
enabled	Enables or disables the CFM VLAN on the bridge. *(True/False)*
cVlanId	A unique, 12-bit VLAN identifier which specifies the VLAN with which this frame is associated. (Integer)
cVlanPriority	The user priority of the tag: a value from 0 through 7. The use and interpretation of this field is defined in ISO/IEC 15802-3. (Integer)
cVlanTpId	The Tag Protocol ID. EtherTypes identify the protocol that follows the VLAN header. (String)
sVlanId	A unique, 12-bit VLAN identifier which specifies the VLAN with which this frame is associated. (Integer)
sVlanPriority	The user priority of the tag: a value from 0 through 7. The use and interpretation of this field is defined in ISO/IEC 15802-3. (Integer)
sVlanTpId	The Tag Protocol ID. EtherTypes identify the protocol that follows the VLAN header. (String)
type	The VLAN type. One of: - singleVlan - stackedVlan
name	*(Read-only)* The name of the VLAN which will be used as a unique key to retrieve the object.

cfmVlan Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmTrunk

Refer to NAME - cfmTrunk for a complete description of this command. The *cfmTrunk* command represents a simulated Trunk.

cfmTrunk Options

Member	Usage
enabled	Enables or disables the simulated MP on this bridge. *(True/False)*
addCcmCustomTlvs	If true, adds a custom CCM TLV to messages.
addDataTlv	This adds a data TLV to messages. This TLV is applicable for LBM/LBR.
addLbmCustomTlvs	If true, adds a custom LBM TLV to messages.
addLbrCustomTlvs	If true, adds a custom LBR TLV to messages.
addLtmCustomTlvs	If true, adds a custom LTM TLV to messages.
addLtrCustomTlvs	If true, adds a custom LTR TLV to messages.
addInterfaceStatusTlv	If true adds an interface status TLV to messages. We do not allow user to change value of Interface Status TLV from here. However the user can always add Interface Status TLV as an optional TLV in Bridge and edit value. This TLV is applicable for CCM. Default is true.
addOrganization SpecificTlv	If true, adds an organization specific TLV to messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.
addPortStatusTlv	If true, adds a port status TLV to messages. We do not allow user to change value of Port Status TLV from here. However user can always add Port Status TLV as an optional TLV in Bridge and edit value. This TLV is applicable for CCM. The default is true.
addSenderIdTlv	If true, adds a Sender TLV to PBB-TE messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.
autoDmIteration	The count for how many times DMMs will be transmitted. Default is 0 (no limit). Min: 0 Max: 2^32
autoDmTimeout	The timeout period in seconds to wait for a response to DMMs. This value should be less than the Auto LB Timer. Default is 30. Min: 1 Max: 65535
autoDmTimer	The time period in seconds between DMMs. Default is 60. Min: 1 Max: 65535
autoLbIteration	The count for how many times LBM will be transmitted. Default is 0 (no limit).

Member	Usage
	Min: 0 Max: 2^32
autoLbTimeout	The timeout period in seconds to wait for a response to LBMs. This value should be less than the Auto LB Timer. Default is 30. Min: 1 Max: 65535
autoLbTimer	The time period in seconds between LBMs. Default is 60. Min: 1 Max: 65535
autoLtIteration	The count for how many times LTM will be transmitted. Default is 0 (no limit). Min: 0 Max: 2^32
autoLtTimeout	The timeout period in seconds to wait for a response to LTMs. This value should be less than the Auto LT Timer. Default is 30. Min: 1 Max: 65535
autoLtTimer	The time period in seconds between LTMs. Default is 60. Min: 1 Max: 65535
cciInterval	The configured time between CCM transmissions. (Integer) (Default = 1 sec) One of: • cci3msec (3 milliseconds) • cci10msec (10 milliseconds) • cci100msec (100 milliseconds) • cci1sec (Default 1 second) • cci10sec (10 seconds) • cci1min (1 minute) • cci10min (10 minutes)
ccmPriority	Sets the priority for Continuity Check Messages. The default is 0. Min: 0 Max: 7
chassisId	Sets the Chassis identifier. Default is 00 00 00 00 00 00. This will take Hex value as input (0-255 byte).
chassisIdLength	Sets the length of the chassis identifier. Default is 6.

Member	Usage
	Min: 0 Max: 255.
chassisIdSubType	Sets the chassis identifier sub-type for the optional TLV messages. Options are: • chassisComponent • interfaceAlias • portComponent • chassisMacAddress • networkAddress • interfaceName • locallyAssigned
dataTlvLength	Sets the length of the Data TLV. Default is 4. Min: 0 Max: 1500.
dataTlvValue	This column will take Hex value of data. This data TLV will be added both for periodic LBM and requested LBM transmit. Default is 44 61 74 61.
dmPriority	Sets the priority for DM Messages. This priority will be used only for periodic DMMs. The default is 0. Min: 0 Max: 7 Note: Backward compatibility is maintained for the legacy `dmmPriority' attribute.
enableAutoDm	If true, enables the automatic sending to DM Messages.
enableAutoLb	If true, enables the automatic sending to Loopback Messages.
enableAutoLt	If true, enables the automatic sending of Link Trace Messages.
dmMethod	Sets the delay mesaurement method. The available options are: • oneWay • twoWay
lbmPriority	Sets the priority for Loopback Messages. This priority will be used only for periodic LBMs. The default is 0. Min: 0 Max: 7
ltmPriority	Sets the priority for Link Trace Messages. This priority will be used only for periodic LTMs. The default is 0.

Member	Usage
	Min: 0 Max: 7
macAddress	The 6-octet MAC Address of the MP.
managementAddress	Sets the Managment Address. Input type is HEX (0-255 byte). Default is 01 02 03 03 04 05.
managementAddress Domain	Sets the Management Address Domain. This will take HEX input (0-255 byte). Default is 4d 61 6e 61 67 65 6d 65 6e 74 20 41 64 64 72 20 44 6f 6d 61 69 6e ("Management Addr Domain").
managementAddress DomainLength	Sets the length of the Management Address domain. Default is 22. Min: 0 Max: 255.
managementAddress Length	Sets the length of the Managment Address. Default is 6. Min: 0 Max: 255.
mdLevel	The MD or MEG level assigned to the MP. (This attribute references an object of *cfmMdLevel*.) (String)
megId	The identifier of the Maintenance Entity Group (MEG). (For use with Y.1731.) The base of this depends on the *megIdFormat* selection (below). (String)
megIdFormat	Sets the format for the megId (for use with Y.1731). The only option is *iccBasedFormat*.
mepId	The number that is used to identify the Maintenance End Point. (Integer)
rdi	The Remote Defect Identification. Possible values include: • auto • on • off
mpType	The type of Maintenance Point. One of: • cfmMIP (Maintenance Intermediate Point) • cfmMEP (Maintenance End Point)
organizationSpecificTlv Length	Sets the length for the Organizational TLV. Default is 4. Min: 4 Max: 1500
organizationSpecificTlvValue	Sets the value for the Organizational TLV. Default is NULL.

Member	Usage
overrideVlanPriority	If true, overrides the set VLAN priority for this bridge, and uses the advanced settings instead. (true/false)
shortMaName	The short name of the MA. The base of this depends on the selection for shortMaNameFormat (below). (String)
shortMaNameFormat	Sets the format for the short MA Name. One of: • primaryVid • characterString • twoOctetInteger • rfc2685VpnId
ttl	Sets the Time To Live for the period OAM. Default is 64. Min: 1 Max: 255
vlan	The VLAN assigned to the MP. (This attribute references an object of *cfmVlan*.) (String)
vlanLocalId	*(Read-only)* The VLAN Local Id. (Integer)
mdLevelLocalId	*(Read-only)* The MD Level Local ID. (Integer)
name	*(Read-only)* The name of the MP which will be used as a unique key to retrieve the object.

cfmTrunk Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmCcmLearnedInfo

Refer to NAME - cfmCcmLearnedInfo for a complete description of this command.

cfmCcmLearnedInfo Options

Member	Usage
cVlan	*(Read-only)* The Stacked VLAN identifier. (String)
sVlan	*(Read-only)* The Single VLAN identifier. (String)
shortMaNameFormat	*(Read-only)* The short Maintenance Association name format. (Integer)
shortMaName	*(Read-only)* The short Maintenance Association name. (String)
errCcmDefect	*(Read-only)True* indicates that the received CCM from this MEP has some incorrect value. (*True / False)*
receivedRdi	*(Read-only)* Indicates the state of the RDI bit in the last received CCM. *True* if RDI = 1; *False* if none has been received. (*True / False)*
mepMacAddress	*(Read-only)* A 6-octet MAC Address for the MEP.
someRmepDefect	*(Read-only)* Indicates the state of the Remote MEP State Machines: • *True* at least one of the Remote MEP state Machines is not

Member	Usage
	receiving valid CCMs from its remote MEPs. • *False* all Remote MEP State Machines are receiving valid CCMs.
receivedAis	*(Read-only)True* indicates that an AIS message has been received from this MEP. (*True / False)*
rmepCcmDefect	*(Read-only)True* Indicates that no CCM is being received from this MEP. (*True / False)*
receivedPortTlvDefect	*(Read-only)* Indicates the port status of the remote MEP. *(True / False)*
receivedIfaceTlvDefect	*(Read-only)* Indicates the interface status of the remote MEP. *(True / False)*
allRmepDead	*(Read-only)True* indicates that this MEP is receiving none of the remote MEP's CCMs. *T(rue / False)*
mdLevel	*(Read-only)* The Maintenance Domain Level. (Integer)
cciInterval	*(Read-only)* The Continuity Check Message interval. (Integer)
rdiRxCount	The number of rdi received.
rdiRxState	Indicates the state of the RDI whether it is *Receiving* or *Idle.*
mepId	*(Read-only)* The MEP identifier. (Integer)
mdNameFormat	*(Read-only)* The Maintenance Domain name format. (Integer)
mdName	*(Read-only)* The Maintenance Domain name. (String)
outOfSequenceCcmCount	*(Read-only)* The number of Out of Sequence Continuity Check Messages. (Integer)

cfmCcmLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmAisLearnedInfo

Refer to NAME - cfmAisLearnedInfo for a complete description of this command.

cfmAisLearnedInfo Options

Member	Usage
sVlan	*(Read-only)* The outer vlan in the received AIS packet. (String)
cVlan	*(Read-only)* The inner vlan in the received AIS packet. (String)
remoteMacAddress	*(Read-only)* MAC address of remote peer mep
localMacAddress	*(Read-only)* MAC address of local peer mep
txState	Indicates whether AIS is being transmitted from this local peer mep or not *(string {Idle, Transmitting})*
txCount	The number of Tx *(integer)*
rxState	Indicates whether AIS is being received from remote peer mep or not *(string {Idle, Transmitting})*
rxCount	The number of Rx *(integer)*
rxInterval	Rx interval of the received AIS from remote mep. It will be shown in string format.

Member	Usage
PBB-TE bVlan	*(Read-only)* The outer vlan in the received AIS packet. (String)
PBB-TE remoteMacAddress	*(Read-only)* MAC address of remote peer mep
PBB-TE localMacAddress	*(Read-only)* MAC address of local peer mep
PBB-TE txState	Indicates whether AIS is being transmitted from this local peer mep or not *(string {Idle, Transmitting})*
PBB-TE txCount	The number of Tx *(integer)*
PBB-TE rxState	Indicates whether AIS is being received from remote peer mep or not *(string {Idle, Transmitting})*
PBB-TE rxCount	The number of Rx *(integer)*
PBB-TE rxInterval	Rx interval of the received AIS from remote mep. It will be shown in string format.

cfmLckLearnedInfo

Refer to NAME - cfmLckLearnedInfo for a complete description of this command.

cfmLckLearnedInfo Options

Member	Usage
sVlan	*(Read-only)* The outer vlan in the received AIS packet. (String)
cVlan	*(Read-only)* The inner vlan in the received AIS packet. (String)
remoteMacAddress	*(Read-only)* MAC address of remote peer mep
localMacAddress	*(Read-only)* MAC address of local peer mep
txState	Indicates whether LCK is being transmitted from this local peer mep or not *(string {Idle, Transmitting})*
txCount	The number of Tx *(integer)*
rxState	Indicates whether LCK is being received from remote peer mep or not *(string {Idle, Transmitting})*
rxCount	The number of Rx *(integer)*
rxInterval	Rx interval of the received LCK from remote mep. It will be shown in string format.
PBB-TE bVlan	*(Read-only)* The outer vlan in the received LCK packet. (String)
PBB-TE remoteMacAddress	*(Read-only)* MAC address of remote peer mep
PBB-TE localMacAddress	*(Read-only)* MAC address of local peer mep
PBB-TE txState	Indicates whether LCK is being transmitted from this local peer mep or not *(string {Idle, Transmitting})*
PBB-TE txCount	The number of Tx *(integer)*
PBB-TE rxState	Indicates whether LCK is being received from remote peer mep or not *(string {Idle, Transmitting})*
PBB-TE rxCount	The number of Rx *(integer)*
PBB-TE rxInterval	Rx interval of the received LCK from remote mep. It will be shown in string format.

cfmTstLearnedInfo

Refer to NAME - cfmTstLearnedInfo for a complete description of this command.

cfmTstLearnedInfo Options

Member	Usage
sVlan	*(Read-only)* The outer vlan in the received TST packet. (String)
cVlan	*(Read-only)* The inner vlan in the received TST packet. (String)
remoteMacAddress	*(Read-only)* MAC address of remote peer mep
localMacAddress	*(Read-only)* MAC address of local peer mep
txState	Indicates whether TST is being transmitted from this local peer mep or not *(string {Idle, Transmitting})*
txCount	The number of Tx *(integer)*
rxCount	The number of Rx *(integer)*
rxOutofSequenceTSTCount	The number of the number of out of sequence TST *(integer)*
PRBSBitErrorsCount	The number of TST packets received with RPBS bit error *(integer)*
PBB-TE bVlan	*(Read-only)* The outer vlan in the received TST packet. (String)
PBB-TE remoteMacAddress	*(Read-only)* MAC address of remote peer mep
PBB-TE localMacAddress	*(Read-only)* MAC address of local peer mep
PBB-TE txState	Indicates whether TST is being transmitted from this local peer mep or not *(string {Idle, Transmitting})*
PBB-TE txCount	The number of Tx *(integer)*
PBB-TE rxCount	The number of Rx *(integer)*
PBB-TE rxOutofSequenceTSTCount	The number of the out of sequence TST *(integer)*
PBB-TE PRBSBitErrorsCount	The number of TST packets received with RPBS bit error *(integer)*

cfmCustomTlv

Refer to NAME - cfmCustomTlv for a complete description of this command.

cfmCustomTlv Options

Member	Usage
type	Sets the type value for the TLV.
length	Sets the length value for the TLV.
value	Sets the data value for the TLV.
includeInCcm	If true, includes the custom TLV in Continuity Check Messages.
includeInLtm	If true, includes the custom TLV in Link Trace Messages.
includeInLtr	If true, includes the custom TLV in Link Trace Responses.
includeInLbm	If true, includes the custom TLV in Loopback Messages.

Member	Usage
includeInLbr	If true, includes the custom TLV in Loopback Responses.
name	(Read-only.) The name of the custom TLV which will be used as a unique key to retrieve the object.

cfmCcmLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmLbLearned Info

Refer to NAME - cfmLbLearnedInfo for a complete description of this command.

cfmLbLearned Info Options

Member	Usage
mdLevel	*(Read-only)* The MD/MEG level for the loopback. (Integer)
reachability	*(Read-only)* Indicates the status of the Ping. (*True / False)*
srcMacAddress	*(Read-only)* The source MAC address from the loopback. A 6-octet MAC Address.
dstMacAddress	*(Read-only)* The destination MAC address from the loopback. A 6-octet MAC Address.
transactionId	*(Read-only)* The identifier the LBM was sent with. (Integer)
sVlan	*(Read-only)* The single VLAN identifier. (String)
cVlan	*(Read-only)* The stacked VLAN identifier. (String)

cfmLbLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmLtLearned Info

Refer to NAME - cfmLtLearnedInfo for a complete description of this command.

cfmLtLearned Info Options

Member	Usage
hopCount	*(Read-only)* The number of hops in the link. (Integer)
mdLevel	*(Read-only)* The Maintenance Domain or Maintenance Entity Group identifier for the link. (Integer)
hops	*(Read-only)* List of hops to reach the particular MEP (MAC Address). List of 6-octet MAC Addresses.
srcMacAddress	*(Read-only)* The source MAC Address for the link. A 6-octet MAC Address.
dstMacAddress	*(Read-only)* The MEP destination MAC Address. A 6-octet MAC Address.
cVlan	(Read-only) The link stacked VLAN identifier. (String)
transactionId	*(Read-only)* Identifier sent with the Link Trace Message (LTM). (Integer)
sVlan	*(Read-only)* The link single VLAN identifier. (String)
replyStatus	*(Read-only)* Indicates the status of the reply. One of:

Member	Usage
	- Complete Reply - Partial Reply - No Reply

cfmLtLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmLtLearned Hop

Refer to NAME - cfmLtLearnedHop for a complete description of this command.

cfmLtLearnedHop Options

Member	Usage
replyTtl	*(Read-only)* The Time to Live/TTL (number of hops) in the Link Trace Reply (LTR) from the MEP or MIP. The TTL is one less than the TTL in the Link Trace Message (LTM) was received by that network element. (Integer)
ingressMac	*(Read-only)* The MAC Address of the MP, if any, residing on the egress port of the bridge. The LTM responder resides on this bridge.
egressMac	*(Read-only)* The MAC Address of the MP, if any, residing on the ingress port of the bridge. The LTM responder resides on this bridge.
self	*(Read-only)* If set to Yes/True (1), indicates that the MAC Address of the MIP is configured on the same bridge port as the MEP transmitting the LTM, and belongs to the path to reach the target. If set to No/False (0), indicates that the MAC address of the MIP (not configured on the same bridge port) is on the other end of the transmitting MEP. This MIP belongs to the path to reach the target and sends the LTR. (True / False)

cfmLtLearnedHop Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmItuLearnedInfo

Refer to NAME - cfmItuLearnedInfo for a complete description of this command.

cfmItuLearnedInfo Options

Member	Usage
srcMacAddress	The source MAC address of the CFM bridge.
dstMacAddress	The destination MAC address of the CFM bridge.
mdLevel	The Maintenance Domain level for the CFM bridge.
sVlan	The S-VLAN identifier of the CFM bridge.

Member	Usage
cVlan	The C-VLAN identifier of the CFM bridge.
valueInNanoSec	*(Read-only)* The measured amount of delay, in nanoseconds. (Integer)
valueInSec	*(Read-only)* The measured amount of delay, in seconds. (Integer)

cfmItuLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPbtCcmLearnedInfo

Refer to NAME - cfmPbtCcmLearnedInfo for a complete description of this command.

cfmPbbTeCcmLearnedInfo Options

Member	Usage
bVlan	*(Read-only)* The VLAN identifier. (String)
remoteMacAddress	*(Read-only)* The remote MAC address of the trunk. A 6-octet MAC Address.
srcMacAddress	*(Read-only)* The source MAC address of the trunk. A 6-octet MAC Address.
shortMaNameFormat	*(Read-only)* The short Maintenance Association name format. (Integer)
shortMaName	*(Read-only)* The short Maintenance Association name. (String)
remoteMepId	*(Read-only)* The remote Maintenance Endpoint identifier. (Integer)
srcMepId	*(Read-only)* The source Maintenance Endpoint identifier. (Integer)
mdLevel	*(Read-only)* The Maintenance Domain Level. (Integer)
mdNameFormat	*(Read-only)* The Maintenance Domain Name format. (Integer)
mdName	*(Read-only)* The Maintenance Domain Name. (String)
receivedRdi	*(Read-only)* Indicates the state of the RDI bit in the last received CCM. *True* if RDI = 1; *False* if none has been received. (*True / False*)
receivedPortTlvDefect	*(Read-only)* Indicates the port status of the remote MEP. (*True / False*)
receivedIfaceTlvDefect	*(Read-only)* Indicates the interface status of the remote MEP. (*True / False)*
errCcmDefect	*(Read-only)True* indicates that the received CCM from the remote MEP has some incorrect value. (*True / False*)
rmepCcmDefect	*(Read-only)* Indicates the state of the Remote MEP State Machines. *True* indicates that the Remote MEP State Machine is not receiving valid CCMs from its Remote MEP. *False* indicates that the Remote MEP State Machine is receiving valid CCMs.
cciInterval	*(Read-only)* The Continuity Check interval. (Integer).
rdiRxCount	The number of rdi received.
rdiRxState	Indicates the state of the RDI whether it is *Receiving* or *Idle.*

Member	Usage
rdiTxCount	The number of rdi transmitted.
rdiTxState	Indicates the state of the RDI whether it is *Transmitting* or *Idle.*
outOfSequenceCcmCount	*(Read-only)* The number of Out of Sequence Continuity Check Messages. (Integer)
errCcmDefectCount	*(Read-only)* The number of error defects in CCMs received. (Integer)
portTlvDefectCount	*(Read-only)* The number of Port TLV defects received. (Integer)
ifaceTlvDefectCount	*(Read-only)* The number of Interface TLV defects received. (Integer)
remoteMepDefectCount	*(Read-only)* The number of Remote MEP defects received. (Integer)

cfmPbbTeCcmLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPbtDmLearnedInfo

Refer to NAME - cfmPbtDmLearnedInfo for a complete description of this command.

cfmPbtDmLearnedInfo Options

Member	Usage
srcMacAddress	The source MAC address of the CFM bridge.
dstMacAddress	The destination MAC address of the CFM bridge.
mdLevel	The Maintenance Domain level for the CFM bridge.
sVlan	The S-VLAN identifier of the CFM bridge.
cVlan	The C-VLAN identifier of the CFM bridge.
valueInNanoSec	*(Read-only)* The measured amount of delay, in nanoseconds. (Integer)
valueInSec	*(Read-only)* The measured amount of delay, in seconds. (Integer)

cfmItuLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPbtLtLearnedInfo

Refer to NAME - cfmPbtLtLearnedInfo for a complete description of this command.

cfmPbtLtLearnedInfo Options

Member	Usage
bVlan	(Read-only.) The trunk VLAN identifier. (String)
dstMacAddress	(Read-only.) The 6-octet destination MAC Address for the trunk.
hopCount	(Read-only.) The number of hops in the link.
hops	(Read-only.) List of hops to reach the particular MEP (MAC address).

Member	Usage
mdLevel	(Read-only.) The MD level for the trunk. (Integer.
replyStatus	(Read-only.) Indicates the status of the reply.
srcMacAddress	(Read-only.) The 6-octet source MAC Address for the trunk.
transactionId	*(Read-only.)* The transaction identifier the LBM was sent with. (Integer)

cfmPbtLtLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.
getLtLearnedHopList		Gets the learned hops for the link.
getFirstLt LearnedHop		Gets the first hop in the link list.
getNextLtLearnedHop		Gets the next hop in the link list.

cfmPbtLbLearnedInfo

Refer to NAME - cfmPbtLbLearnedInfo for a complete description of this command.

cfmPbtLbLearnedInfo Options

Member	Usage
bVlan	*(Read-only)* The trunk VLAN identifier. (String)
mdLevel	*(Read-only)* The MD level for the trunk. (Integer)
srcMacAddress	*(Read-only)* The source MAC address for the trunk. A 6-octet MAC Address.
dstMacAddress	*(Read-only)* The destination MAC address for the trunk. A 6-octet MAC Address.
transactionId	*(Read-only)* The transaction identifier the LBM was sent with. (Integer)
reachability	*(Read-only)* Indicates the status of the Ping. (*True / False)*
rtt	*(Read-only)* The time difference between the Ping request and the Ping reply. (Integer)

cfmPbtLbLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPbtPeriodicOamDm LearnedInfo

Refer to NAME - cfmPbtPeriodicOamDmLearnedInfo for a complete description of this command.

cfmPbtPeriodicOamDmLearnedInfo Options

Member	Usage
averageDelayNanoSec	*(Read-only)* Retrieves the average delay for the Period OAM DM in nanoseconds.
averageDelaySec	*(Read-only)* Retrieves the average delay for the Period OAM DM in seconds.
bVlan	*(Read-only)* The trunk VLAN identifier. (String)

Member	Usage
mdLevel	*(Read-only)* The MD level for the trunk. (Integer)
srcMacAddress	*(Read-only)* The source MAC address for the trunk. A 6-octet MAC Address.
dstMacAddress	*(Read-only)* The destination MAC address for the trunk. A 6-octet MAC Address.
noReplyCount	*(Read-only)*The number of no replies for the Periodic OAM messages.
recentDelayNanoSec	*(Read-only)* Retrieves the the most recent delay for the Period OAM DM in nanoseconds.
recentDelaySec	*(Read-only)* Retrieves the most recent delay for the Period OAM DM in seconds
oneDMReceivedCount	The number of 1DM packets received.
dmmCountSent	The number of DMM packets sent. **Note:** For One-way the *DMM Sent Count* and *No Reply Count* will be *NA*. For Two-way, the *1DM Received Count* will be *NA*.
averageDelayVariationNanoSec	*(Read-only)* Retrieves the avrerage delay variation for the Period OAM DM in nanoseconds.
recentDelayVariationNanoSec	*(Read-only)* Retrieves the most recent delay variation for the Period OAM DM in nano seconds

cfmPbtPeriodicOamDmLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPbtPeriodicOamLbLearnedInfo

Refer to NAME - cfmPbtPeriodicOamLbLearnedInfo for a complete description of this command.

cfmPbtPeriodicOamLbLearnedInfo Options

Member	Usage
bVlan	*(Read-only)* The trunk VLAN identifier. (String)
mdLevel	*(Read-only)* The MD level for the trunk. (Integer)
srcMacAddress	*(Read-only)* The source MAC address for the trunk. A 6-octet MAC Address.
dstMacAddress	*(Read-only)* The destination MAC address for the trunk. A 6-octet MAC Address.
noReplyCount	(Read-only.) The number of loopback messages that haven't been replied to.
recentRtt	(Read-only.) The time difference between the most recent Ping request and the Ping reply, in microseconds (us). (Integer)
recentReachability	(Read-only.) Indicates the status of the Ping.
transactionId	(Read-only.) The transaction identifier the LBM was sent with. (Integer)
averageRtt	(Read-only.) The average time difference between the t Ping

Member	Usage
	requests and the Ping replies, in microseconds (us). (Integer)

cfmPbtPeriodicOamLbLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPbtPeriodicOamLtLearnedInfo

Refer to NAME - cfmPbtPeriodicOamLtLearnedInfo for a complete description of this command.

cfmPbtPeriodicOamLtLearnedInfo Options

Member	Usage
averageHopCount	(Read-only.) The average hop count for the Link trace.
bVlan	(Read-only.) The trunk VLAN identifier. (String)
completeReplyCount	(Read-only.) The number of replys to sent Link trace messages.
dstMacAddress	(Read-only.) The 6-octet destination MAC Address for the trunk.
ltmSentCount	(Read-only.) The number of link trace messages sent.
mdLevel	(Read-only.) The MD level for the trunk. (Integer.
noReplyCount	(Read-only.) The number of link trace messages sent that were not replied to.
partialReplyCount	(Read-only.) The number of link trace messages sent that received some sort of reply.
recentHopCount	(Read-only.) The number of hops in the link.
recentHops	(Read-only.) List of hops to reach the particular MEP (MAC address).
recentReplyStatus	(Read-only.) Indicates the status of the reply.
srcMacAddress	(Read-only.) The 6-octet source MAC Address for the trunk.
transactionId	*(Read-only.)* The transaction identifier the LBM was sent with. (Integer)

cfmPbtLtLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.
getLtLearnedHopList		Gets the learned hops for the link.
getFirstLt LearnedHop		Gets the first hop in the link list.
getNextLtLearnedHop		Gets the next hop in the link list.

cfmPeriodicOamDmLearnedInfo

Refer to NAME - cfmPeriodicOamDmLearnedInfo for a complete description of this command.

cfmPeriodicOamDmLearnedInfo Options

Member	Usage
averageDelayNanoSec	*(Read-only)* Retrieves the average delay for the Period OAM DM in nanoseconds.

Member	Usage
averageDelaySec	*(Read-only)* Retrieves the average delay for the Period OAM DM in seconds
cVlan	*(Read-only)* The C-VLAN identifier. (String)
sVlan	*(Read-only)* The S-VLAN identifier. (String)
mdLevel	*(Read-only)* The MD level. (Integer)
srcMacAddress	*(Read-only)* The source MAC address. A 6-octet MAC Address.
dstMacAddress	*(Read-only)* The destination MAC address for the trunk. A 6-octet MAC Address.
noReplyCount	*(Read-only)*The number of no replies for the Periodic OAM messages.
recentDelayNanoSec	*(Read-only)* Retrieves the the most recent delay for the Period OAM DM in nanoseconds.
recentDelaySec	*(Read-only)* Retrieves the most recent delay for the Period OAM DM in seconds
oneDMReceivedCount	The number of 1DM packets received.
dmmCountSent	The number of DMM packets sent. **Note:** For One-way the *DMM Sent Count* and *No Reply Count* will be *NA*. For Two-way, the *1DM Received Count* will be *NA*.
averageDelayVariationNanoSec	*(Read-only)* Retrieves the avrerage delay variation for the Period OAM DM in nanoseconds.
recentDelayVariationNanoSec	*(Read-only)* Retrieves the most recent delay variation for the Period OAM DM in nano seconds

cfmPeriodicOamDmLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPeriodicOamLbLearnedInfo

Refer to NAME - cfmPeriodicOamLbLearnedInfo for a complete description of this command.

cfmPeriodicOamLbLearnedInfo Options

Member	Usage
cVlan	*(Read-only)* The C-VLAN identifier. (String)
mdLevel	*(Read-only)* The MD level (Integer)
srcMacAddress	*(Read-only)* The source MAC address. A 6-octet MAC Address.
dstMacAddress	*(Read-only)* The destination MAC address. A 6-octet MAC Address.
noReplyCount	(Read-only.) The number of loopback messages that haven't been replied to.
recentRtt	(Read-only.) The time difference between the most recent Ping request and the Ping reply, in microseconds (us). (Integer)
recentReachability	(Read-only.) Indicates the status of the Ping.

Member	Usage
transactionId	(Read-only.) The transaction identifier the LBM was sent with. (Integer)
sVlan	*(Read-only)* The S-VLAN identifier. (String)
averageRtt	(Read-only.) The average time difference between the t Ping requests and the Ping replies, in microseconds (us). (Integer)

cfmPeriodicOamLbLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.

cfmPeriodicOamLtLearnedInfo

Refer to NAME - cfmPeriodicOamLtLearnedInfo for a complete description of this command.

cfmPeriodicOamLtLearnedInfo Options

Member	Usage
averageHopCount	(Read-only.) The average hop count for the Link trace.
cVlan	(Read-only.) The C-VLAN identifier. (String)
completeReplyCount	(Read-only.) The number of replys to sent Link trace messages.
dstMacAddress	(Read-only.) The 6-octet destination MAC Address for the trunk.
ltmSentCount	(Read-only.) The number of link trace messages sent.
mdLevel	(Read-only.) The MD level for the trunk. (Integer.
noReplyCount	(Read-only.) The number of link trace messages sent that were not replied to.
partialReplyCount	(Read-only.) The number of link trace messages sent that received some sort of reply.
recentHopCount	(Read-only.) The number of hops in the link.
recentHops	(Read-only.) List of hops to reach the particular MEP (MAC address).
recentReplyStatus	(Read-only.) Indicates the status of the reply.
srcMacAddress	(Read-only.) The 6-octet source MAC Address for the trunk.
sVlan	(Read-only.) The S-VLAN identifier. (String)
transactionId	*(Read-only.)* The transaction identifier the LBM was sent with. (Integer)

cfmPeriodicOamLtLearnedInfo Subcommands

Class	Member	Usage
setDefault		Sets the options to the default values.
getLtLearnedHopList		Gets the learned hops for the link.
getFirstLt LearnedHop		Gets the first hop in the link list.
getNextLtLearnedHop		Gets the next hop in the link list.

MPLS-TP

The Multi-Protocol Label Switching - Transport Profile (MPLS-TP) is the result of a joint effort by the Internet Engineering Task Force (IETF) and the International Telecommunication Union (ITU-T) based on their previously respective and separate work in the area of Provider Backbone Bridging (PBB) and (Transport MPLS)T-MPLS. MPLS is now a foundation of IP-based networks providing value added services such as traffic engineering and VPN services. The success and familiarity of MPLS in the core is driving service providers to deploy MPLS beyond the core of the network into access, aggregation and backhaul networks supporting broadband, business and mobility services.

The commands related to MPLS-TP are as follows:

- mplsTpServer — provides access to the MPLS-TP part of a port's protocol server.
- mplsTpRouter — add a simulated mpls-tp router.
- mplsTpLspPwRng — adds an lsp pw range.
- mplsTpInterface — a network interface to be included in mplsTpRouter.
- mplsTpGeneralLearnedInfo — learned general information associated with an mplsTpRouter.
- mplsTpLmLearnedInfo — learned lm information associated with an mplsTpRouter.
- mplsTpDmLearnedInfo — learned dm information associated with an mplsTpRouter.
- mplsTpPingLearnedInfo — learned ping information associated with an mplsTpRouter.
- mplsTpTracerouteLearnedInfo — learned traceroute information associated with an mplsTpRouter.

mplsTpServer

Refer to NAME - mplsTpServer for a complete description of this command. The *mplsTpServer* command is necessary in order to access the MPLS-TP protocol server for a particular port.

The important options and subcommands of this command are:

mplsTpServer Options

Member	Usage
bfdCcChannelType	The bfd cc channel type.
apsChannelType	The asp channel type.
onDemandCvChannelType	The on demand cv channel type.
faultManagementChannelType	The fault management channel type.
lossMeasurementChannelType	The loss measurement channel type.
y1731ChannelType	The y1731 channel type.
pwStatusChannelType	The PW status channel type.
delayManagementChannelType	The delay measurement channel type.

mplsTpServer Subcommands

Member	Usage
cget configure	The list of subcommands pertaining to mplsTpServer.

Member	Usage
select addRouter delRouter getRouter setRouter getFirstRouter getNextRouter clearAllRouters write setDefault set get showRouterNames	

mplsTpRouter

Refer to NAME - mplsTpRouter for a complete description of this command. The *mplsTpRouter* command adds a simulated mpls-tp router.

The important options and subcommands of this command are:

mplsTpRouter Options

Member	Usage
enabled	Enables this simulated mpls-tp router. This can be enabled/disabled based on its value set as true/false.
routerId	The ID of the simulated router, which is expressed as an IP address.
enableCccvPause	Enable cccv pause. This can be enabled/disabled based on its value set as true/false. Default = false
cccvPauseTriggerOption	The cccv pause trigger option.
enableCccvResume	Enables cccv resume. This can be enabled/disabled based on its value set as true/false. Default = false
cccvResumeTriggerOption	The cccv resume trigger option.
apsTriggerType	The aps trigger type. Possible values include: • clear

Member	Usage
	• forcedSwitch • manualSwitch • lockout • exercise • freeze
lmInterval	The lm interval.
lspTraceRouteTtlLimit	The lsp trace route ttl limit.
enableLspTraceRoute	Enables lsp trace route.
lmTrafficClass	The lm traffic class value.
enableLspPingFecStackValidation	Enables lsp ping fec stack validation.
enablePwStatusFault	Enables pw fault status.
alarmType	The type of alarm. Possible values include: • ietf • y1731
dmType	The DM type. Possible values include: • ietf • y1731
dmIterations	The total dm iterations.
lastDmResponseTimeout	The last dm response timeout.
lmTxStep	The increment value for lm transmit.
enableAlarm	Enables alarm.
counterType	The counter type. Possible values include: • 32Bit • 64Bit
dmPadLen	The dm pad length.
enableAlarmAis	Enables alarm ais.
periodicity	Indicates the periodicity.
dmTrafficClass	The dm traffic class.
enableDmTrigger	Enables dm trigger.
lspTraceRouteResponseTimeout	The lsp trace route response timeout.
enableAlarmLck	Enables alarm lck.
alarmTrigger	The alarm trigger. Possible values include: • clear • start
lmIterations	The lm iterations.
enableAlarmSetLdi	Enables alarm set ldi.
dmInterval	The dm interval value.
lmInitialRxValue	The initial lm receive value.
lmInitialTxValue	The initial lm transmit value.

Member	Usage
enableLspPing	Enables lsp ping.
lspPingTtlValue	The lsp ping ttl value.
lspPingResponseTimeout	The lsp ping response timeout.
enableApsTrigger	Enables APS trigger.
dmRequestPaddedReply	The dm request padded reply.
lastLmResponseTimeout	The last lm response timeout.
lmRxStep	The step value for lm receive.
dmMode	The dm mode. Possible values include: • noResponseExpected • responseExpected
dmTimeFormat	The dm time format. Possible values include: • ieee • ntp
enableLmTrigger	Enables lm trigger.
lmType	The lm type. Possible values include: • ietf • y1731
lmMode	The lm mode. Possible values include: • responseExpected • noResponseExpected
enableLspTraceRouteFecStackValidation	Enables lsp trace route fec stack validation.

mplsTpRouter Subcommands

Member	Usage
cget configure setDefault addInterface delInterface getInterface setInterface getFirstInterface getNextInterface clearAllInterfaces showInterfaceNames refreshLearnedInformation getGeneralLearnedInformationList	The list of subcommands pertaining to mplsTpRouter.

Member	Usage
getFirstGeneralLearnedInformation getNextGeneralLearnedInformation getDmLearnedInformationList getFirstDmLearnedInformation getNextDmLearnedInformation getLmLearnedInformationList getFirstLmLearnedInformation getNextLmLearnedInformation getPingLearnedInformationList getFirstPingLearnedInformation getNextPingLearnedInformation getTracerouteLearnedInformationList getFirstTracerouteLearnedInformation getNextTracerouteLearnedInformation addRecordForTrigger clearRecordsForTrigger trigger	

mplsTpLspPwRng

Refer to NAME - mplsTpLspPwRange for a complete description of this command. The *mplsTpLspPwRange* command adds an lsp pw range.

The important options and subcommands of this command are:

mplsTpLspPwRange Options

Member	Usage
enabled	Enables this LSP PW Range.
typeOfRange	The type of range. Possible values include: • lsp • pw
rangeRole	The role of the range. Possible values include: • none • working • protect
peerLspOrPwRange	The type of range.
numberOfLsp	The total number of lsps.
numberOfPwPerLsp	The total number of PWs per LSP.
lspOutgoingLabel	The outgoing LSP label.

Member	Usage
lspIncomingLabel	The incoming LSP label.
pwOutgoingLabel	The outgoing PW label.
pwIncomingLabel	The incoming PW label.
cccvInterval	The cccv interval value.
cccvType	The cccv type. Possible values include: • bfdCc • y1731
megIdPrefix	The prefix for the meg id.
srcMepId	The source MEP id.
destMepId	The destination MEP id.
ipType	The id type.Possible values include: • ipv4 • ipv6
ipAddress	The IP address value.
ipAddressMask	The IP address mask.
ipAddressStep	The increment value for the IP address.
trafficGroupId	The traffic group id.
skipZeroVlanId	If true, skips vlands with zero values.
vlanIncrementMode	The increment vaue for the vlan. Possible values include: • noIncrement • parallelIncrement • innerFirst • outerFirst
vlanTpId	The vlan TP id.
vlanPriority	The vlan priority value.
vlanId	The vlan id.
vlanCount	The vlan count.
enableVlan	If true, enables the vlan.
macAddress	The MAC address.
repeatMac	If true, repeats the MAC addresses.
macPerPw	The total MAC per PW.
lspOutgoingLabelStep	The increment value for the outgoing lsp label.
lspIncomingLabelStep	The increment value for the incoming lsp label.
pwOutgoingLabelStep	The increment value for the outgoing pw label.
pwIncomingLabelStep	The increment value for the incoming pw label.
megIdIntegerStep	The increment value for the MEG..
srcMepIdStep	The increment value for the source MEP.
destMepIdStep	The increment value for the destination MEP.
description	The description of the range.
pwIncomingLabelStepAcrossLsp	The increment value for the incoming pw label across

Member	Usage
	lsp.
pwOutgoingLabelStepAcrossLsp	The increment value for the outgoing pw label across lsp.
alarmType	The type of alarm. Possible values include: • ietf • y1731
dmType	The DM type. Possible values include: • ietf • y1731
lmType	The LM type. Possible values include: • ietf • y1731
ipHostPerLsp	The IP host per lsp.
cccvTrafficClass	The cccv traffic class value.
typeOfProtectionSwitching	The type of switching protection. Possible values include: • 1:1Unidirectional • 1+1Unidirectional • 1:1Bidirectional • 1+1Bidirectional
revertive	The revertive value.
waitToRevertTime	The wait time to revert.
onDemandCvTrafficClass	The on demand cv traffic class value.
pwStatusTrafficClass	The pw status traffic class value.
alarmTrafficClass	The alarm traffic class value.
dmTimeFormat	The dm time format. Possible values include: • ieee • ntp
dmTrafficClass	The dm traffic class value.
lmCounterType	The lm counter type. Possible values include: • 32Bit • 64Bit
lmInitialTxValue	The initial lm transmit value.
lmTxStep	The increment value for lm transmit.
lmInitialRxValue	The initial lm receive value.
lmRxStep	The increment value for lm receive.
lmTrafficClass	The lm traffic class value.
destGlobalId	The destination global id.
srcLspNumber	.The source LSP number.

Member	Usage
srcTunnelNumberStep	The increment value for the source tunnel number.
srcTunnelNumber	The source tunnel number value.
destNodeId	The destination node id.
destAcIdStep	The increment value for the destination ac id.
srcNodeId	The source node id.
destTunnelNumberStep	The increment value for the destination tunnel number.
destAcId	The destination ac id.
supportSlowStart	If true, support slow start.
pwStatusFaultReplyInterval	The interval value for the pw fault status.
destTunnelNumber	The destination tunnel number.
destLspNumber	The destination LSP number.
destLspNumberStep	The increment value for the destination LSP number.
apsType	The aps types. Possible values include: • ietf • y1731
apsTrafficClass	The aps traffic class value.
srcAcIdStep	The increment value for the source ac id.
srcGlobalId	The source global id.
srcAcId	The source ac id.
srcLspNumberStep	The increment value for the source LSP number.

mplsTpLspPwRange Subcommands

Member	Usage
cget configure setDefault	The list of subcommands pertaining to mplsTpLspPwRange.

mplsTpInterface

Refer to NAME - mplsTpInterface for a complete description of this command. The *mplstpInterface* holds the information related to a single interface on the simulated router.

The important options and subcommands of this command are:

mplsTpInterface Options

Member	Usage
enabled	Enables the use of the simulated interface.
dutMacAddress	The MAC address of the DUT.
interfaces	The number of interfaces.

mplsTpInterface Subcommands

Member	Usage
cget configure	The list of subcommands pertaining to mplsTpInterface.

Member	Usage
setDefault addLspPwRange delLspPwRange getLspPwRange setLspPwRange getFirstLspPwRange getNextLspPwRange clearAllLspPwRanges showLspPwRangeNames	

mplsTpGeneralLearnedInfo

Refer to NAME - mplsTpGeneralLearnedInfo for a complete description of this command.

mplsTpLmLearnedInfo

Refer to NAME - mplsTpLmLearnedInfo for a complete description of this command.

mplsTpDmLearnedInfo

Refer to NAME - mplsTpDmLearnedInfo for a complete description of this command.

mplsTpPingLearnedInfo

Refer to NAME - mplsTpPingLearnedInfo for a complete description of this command.

mplsTpTracerouteLearnedInfo

Refer to NAME - mplsTpTracerouteLearnedInfo for a complete description of this command.

Return Codes

All commands in the Tcl API use a common set of return codes. These codes are listed in Table 3-1. These codes are global Tcl variables, which may be referred with a preceding '$' (for example, $ixTcl_ok) in a global context or a preceding '$::' (for example, $::ixTcl_ok) in any context. The symbolic codes should be used in preference to literal values.

Tcl API Return Codes

Code	Value	Usage
ixTcl_ok	0	No error, successful return.
ixTcl_generalError	1	An error has occurred.
ixTcl_versionMismatch	2	The software version for the Tcl API does not match that used on a connected chassis.
ixTcl_chassisTimeout	3	A timeout occurred while connecting to a chassis.
ixTcl_notAvailable	100	A port may not exist of may be in use by another user.
ixTcl_unsupportedFeature	101	The port does not support a feature.
ixTcl_outOfMemory	102	The Tcl execution has run out of main memory.
ixTcl_addedAsDisabled	103	The entry was added, but was disabled due to quantity or volume constraints.
ixTcl_notLicensed	104	No license has been installed for a required product feature.
ixTcl_noWriteRequired	200	A write operation was performed when none was required.
ixTcl_invalidChassisChain	201	An invalid chassis chain was required.
ixTcl_hardwareConflict	202	When adding a load module, a duplicate load module serial number was encountered.

[This page intentionally left blank]

Appendix A: High-Level API

Arguments to the high-level APIs are passed in one of two ways:

- **By value** — denoted by (By value) in the description. By value arguments are either a constant or a $variable reference. For example:

```
{{1 1 1} {1 2 1}} -or- $portList
```

- **By reference** — denoted by (By reference) in the description. By reference arguments must be references to variables, **without** the `$'. For example, *pl* after *set pl {{1 1 1} [1 1 2}}*.

Almost all commands return a value of 0 on successful operation. This can be symbolically referred to as *$TCL_OK* in a global context, or *$TCL_OK* otherwise. In the examples in this section, a value of *0* will be used.

Similarly predefined quantities such as *one2oneArray* are defined in the global context. If your program is running in other than the global context, then it is necessary to include a double colon :: before the constant or variable name. For example, *::one2oneArray*.

NAME - ixExportLearnedInfoToCSV

`ixExportLearnedInfoToCsv` - exports the learned information into a file in CSV format.

SYNOPSIS

```
ixExportLearnedInfoToCsv
```

DESCRIPTION

This command is part of the IxTclProtocol library and used to export learned information into some file in csv format.

Arguments

portDataList

The list of ports on which the learned info is sought.

proto

The protocol for which the learned info is sought and required to be saved to a file.

FileName

The name of the file in which the learned info is to be saved.

SEE ALSO

NAME - ixIsArpInstalled

`ixIsArpInstalled` - check if the optional ARP package is installed.

SYNOPSIS

```
ixIsArpInstalled
```

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsBgpInstalled

ixIsBgpInstalled - check if the optional BGP package is installed.

SYNOPSIS

ixIsBgpInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsIgmpInstalled

`ixIsIgmpInstalled` - check if the optional IGMP package is installed.

SYNOPSIS

ixIsIgmpInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsIsisInstalled

`ixIsIsisInstalled` - check if the optional IS-IS package is installed.

SYNOPSIS

ixIsIsisInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsLdpInstalled

`ixIsLdpInstalled` - check if the optional LDP package is installed.

SYNOPSIS

ixIsLdpInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsMldInstalled

`ixIsMldInstalled` - check if the optional MLD package is installed.

SYNOPSIS

ixIsMldInstalled

DESCRIPTION

This command has been deprecated; it always returns true to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the `licenseManagement` command.

SEE ALSO

licenseManagement

NAME - ixIsOspfInstalled

`ixIsOspfInstalled` - check if the optional OSPF package is installed.

SYNOPSIS

ixIsOspfInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsOspfV3Installed

`ixIsOspfV3Installed` - check if the optional OSPFv3 package is installed.

SYNOPSIS

ixIsOspfV3Installed

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsPimsmInstalled

`ixIsPimsmInstalled` - check if the optional PIM-SM package is installed.

SYNOPSIS

ixIsPimsmInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME- ixIsRipInstalled

ixIsRipInstalled - check if the optional RIP package is installed.

SYNOPSIS

ixIsRipInstalled

DESCRIPTION

The **ixIsRipInstalled** command returns 0 or 1, depending on whether the RIP package is installed or not.

ARGUMENTS

None

RETURNS

true

The RIP package is installed.

false

The RIP package is not installed.

EXAMPLES

Check to see whether RIP is installed:

```
package require IxTclHal if {[ixIsRipInstalled]}{ ixPuts "Rip is installed"
} else { ixPuts "Rip is not installed"  }
```

SEE ALSO

NAME - ixIsArpInstalled NAME - ixIsBgpInstalled, NAME - ixIsIgmpInstalled, NAME - ixIsIs-isInstalled, NAME - ixIsLdpInstalled, NAME - ixIsMldInstalled, NAME - ixIsOspfInstalled, NAME - ixIsOspfV3Installed, NAME - ixIsPimsmInstalled, NAME - ixIsRipngInstalled, NAME - ixIsRsvpInstalled, NAME - IxIsVpnL2Installed, NAME - ixIsVpnL3Installed

NAME - ixIsRipngInstalled

`ixIsRipngInstalled` - check if the optional RIPng package is installed.

SYNOPSIS

ixIsRipngInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsRsvpInstalled

ixIsRsvpInstalled - check if the optional RSVP package is installed.

SYNOPSIS

ixIsRsvpInstalled

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - IxIsVpnL2Installed

`ixIsVpnL2Installed` - check if the optional Layer 2 VPN package is installed.

SYNOPSIS

ixIsVpnL2Installed

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixIsVpnL3Installed

`ixIsVpnL3Installed` - check if the optional Layer 3 VPN package is installed.

SYNOPSIS

ixIsVpnL3Installed

DESCRIPTION

This command has been deprecated; it always returns *true*to indicate that the feature is installed. Feature license checking is now performed on a chassis by chassis basis and is supported by the *licenseManagement* command.

SEE ALSO

licenseManagement

NAME - ixStartBFD

ixStartBFD - start BFD on a group of ports simultaneously.

SYNOPSIS

ixStartBFD *portList*

DESCRIPTION

The **ixStartBFD** command sends a message to the IxServer to start protocol server BFD operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

- one2oneArray, one2manyArray, many2oneArray, many2manyArray
- Or a reference to a list.

  For example, *pl* after
  set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
  set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
 package require IxTclHal

 set host galaxy

 set username user

 # Check if we're running on UNIX - connect to the TCL Server

 # which must be running on the chassis

 if [isUNIX] {

 if [ixConnectToTclServer $host] {

 ixPuts "Could not connect to $host"

 return 1

}

}

 # Now connect to the chassis

 if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return
}
 # Get the chassis ID to use in port lists
 set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB $portB]]
 set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo return 1
}
# Take ownership of the ports we'll use
 if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
 map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartBfd pl1] != 0} {
puts "Could not start Bfd for $pl1"
 }
 if {[ixStartBfd pl2] != 0} {
puts "Could not start Bfd for $pl2"
 }
 if {[ixStartBfd pl3] != 0} {
```

```
puts "Could not start Bfd for $pl3"
}
 if {[ixStartBfd pl4] != 0} {
 puts "Could not start Bfd for $pl4"
 }
if {[ixStartBfd one2oneArray] != 0}
{
 puts "Could not start Bfd for $one2oneArray"
}
 # Let go of the ports that we reserved ixClearOwnership $pl4
# Disconnect from the chassis we're using ixDisconnectFromChassis $host
 # If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopBFD

NAME - ixStartBGP4

`ixStartBGP4` - start BGP4 on a group of ports simultaneously.

SYNOPSIS

ixStartBGP4 *portList*

DESCRIPTION

The `ixStartBGP4` command sends a message to the IxServer to start protocol server BGP4 operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

or a reference to a list.

For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
 package require IxTclHal

 set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis
```

```
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists.
set chas [ixGetChassisID $host]
 set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
 $portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
 map config -type one2one
 map add $chas $cardA $portA $chas $cardB $portB
 map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartBGP4 pl1] != 0} {
puts "Could not start BGP4 for $pl1"
}
if {[ixStartBGP4 pl2] != 0} {
```

```
puts "Could not start BGP4 for $pl2"
}
if {[ixStartBGP4 pl3] != 0} {
 puts "Could not start BGP4 for $pl3"
 }
if {[ixStartBGP4 pl4] != 0} {
puts "Could not start BGP4 for $pl4"
}
if {[ixStartBGP4 one2oneArray] != 0} {
puts "Could not start BGP4 for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
 # If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopBGP4

NAME - ixStartCfm

ixStartCfm — start CFM on a group of ports simultaneously.

SYNOPSIS

ixStartCFM *portList*

DESCRIPTION

The **ixStartCfm** command sends a message to the IxServer to start protocol server CFM operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after

set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartCfm pl1] != 0} {
puts "Could not start Cfm for $pl1"
}
if {[ixStartCfm pl2] != 0} {
puts "Could not start Cfm for $pl2"
```

```
}
if {[ixStartCfm pl3] != 0} {
puts "Could not start Cfm for $pl3"
}
if {[ixStartCfm pl4] != 0} {
puts "Could not start Cfm for $pl4"
}
if {[ixStartCfm one2oneArray] != 0} {
puts "Could not start Cfm for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopCfm

NAME - ixStartEigrp

ixStartEigrp — start EIGRP on a group of ports simultaneously.

SYNOPSIS

ixStartEigrp *portList*

DESCRIPTION

The **ixStartEigrp** command sends a message to the IxServer to start protocol server EIGRP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartEigrp pl1] != 0} {
puts "Could not start Eigrp for $pl1"
}
if {[ixStartEigrp pl2] != 0} {
puts "Could not start Eigrp for $pl2"
```

```
}
if {[ixStartEigrp pl3] != 0} {
puts "Could not start Eigrp for $pl3"
}
if {[ixStartEigrp pl4] != 0} {
puts "Could not start Eigrp for $pl4"
}
if {[ixStartEigrp one2oneArray] != 0} {
puts "Could not start Eigrp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopEigrp

NAME - ixStartIsis

ixStartIsis — start ISIS on a group of ports simultaneously.

SYNOPSIS

ixStartIsis *portList*

DESCRIPTION

The **ixStartIsis** command sends a message to the IxServer to start protocol server ISIS operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLE

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartIsis pl1] != 0} {
puts "Could not start Isis for $pl1"
}
if {[ixStartIsis pl2] != 0} {
puts "Could not start Isis for $pl2"
```

```
}
if {[ixStartIsis pl3] != 0} {
puts "Could not start Isis for $pl3"
}
if {[ixStartIsis pl4] != 0} {
puts "Could not start Isis for $pl4"
}
if {[ixStartIsis one2oneArray] != 0} {
puts "Could not start Isis for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopIsis

NAME - ixStartLdp

ixStartLdp — starts LDP on a group of ports simultaneously.

SYNOPSIS

ixStartLdp *portList*

DESCRIPTION

The **ixStartLdp** command sends a message to the IxServer to start protocol server LDP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLE

```
package require IxTclHal
set host galaxy
set username user
# Check if we’re running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartLdp pl1] != 0} {
puts "Could not start Ldp for $pl1"
}
if {[ixStartLdp pl2] != 0} {
puts "Could not start Ldp for $pl2"
```

```
}
if {[ixStartLdp pl3] != 0} {
puts "Could not start Ldp for $pl3"
}
if {[ixStartLdp pl4] != 0} {
puts "Could not start Ldp for $pl4"
}
if {[ixStartLdp one2oneArray] != 0} {
puts "Could not start Ldp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopLdp

NAME - ixStartMld

ixStartMld — start MLD on a group of ports simultaneously.

SYNOPSIS

ixStartMld *portList*

DESCRIPTION

The **ixStartMld** command sends a message to the IxServer to start protocol server MLD operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartMld pl1] != 0} {
puts "Could not start Mld for $pl1"
}
if {[ixStartMld pl2] != 0} {
puts "Could not start Mld for $pl2"
```

```
}
if {[ixStartMld pl3] != 0} {
puts "Could not start Mld for $pl3"
}
if {[ixStartMld pl4] != 0} {
puts "Could not start Mld for $pl4"
}
if {[ixStartMld one2oneArray] != 0} {
puts "Could not start Mld for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopMld

NAME - ixStartOspf

ixStartOspf — start OSPF on a group of ports simultaneously.

SYNOPSIS

ixStartOspf *portList*

DESCRIPTION

The **ixStartOspf** command sends a message to the IxServer to start protocol server OSPF operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartOspf pl1] != 0} {
puts "Could not start Ospf for $pl1"
}
if {[ixStartOspf pl2] != 0} {
puts "Could not start Ospf for $pl2"
```

```
}
if {[ixStartOspf pl3] != 0} {
puts "Could not start Ospf for $pl3"
}
if {[ixStartOspf pl4] != 0} {
puts "Could not start Ospf for $pl4"
}
if {[ixStartOspf one2oneArray] != 0} {
puts "Could not start Ospf for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopOspf

NAME - ixStartOspfV3

ixStartOspfV3 — start OSPFv3 on a group of ports simultaneously.

SYNOPSIS

ixStartOspfV3 *portList*

DESCRIPTION

The **ixStartOspfV3** command sends a message to the IxServer to start protocol server OSPFv3 operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartOspfV3 pl1] != 0} {
puts "Could not start OspfV3 for $pl1"
}
if {[ixStartOspfV3 pl2] != 0} {
puts "Could not start OspfV3 for $pl2"
```

```
}
if {[ixStartOspfV3 pl3] != 0} {
puts "Could not start OspfV3 for $pl3"
}
if {[ixStartOspfV3 pl4] != 0} {
puts "Could not start OspfV3 for $pl4"
}
if {[ixStartOspfV3 one2oneArray] != 0} {
puts "Could not start OspfV3 for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopOspfV3

NAME - ixStartPimsm

ixStartPimsm — start PIM-SM on a group of ports simultaneously.

SYNOPSIS

ixStartPimsm *portList*

DESCRIPTION

The **ixStartPimsm** command sends a message to the IxServer to start protocol server PIM-SM operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartPimsm pl1] != 0} {
puts "Could not start Pimsm for $pl1"
}
if {[ixStartPimsm pl2] != 0} {
puts "Could not start Pimsm for $pl2"
```

```
}
if {[ixStartPimsm pl3] != 0} {
puts "Could not start Pimsm for $pl3"
}
if {[ixStartPimsm pl4] != 0} {
puts "Could not start Pimsm for $pl4"
}
if {[ixStartPimsm one2oneArray] != 0} {
puts "Could not start Pimsm for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopPimsm

NAME - ixStartRip

ixStartRip—start RIP on a group of ports simultaneously.

SYNOPSIS

ixStartRip *portList*

DESCRIPTION

The ixStartRip command sends a message to the IxServer to start protocol server RIP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example pl after

set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or

set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLE

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list 1,$cardA,$portA]
set pl2 [list 1,$cardA,$portA 1,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list 1 $cardB $portB]]
set pl4 [list [list 1,$cardA,$portA] [list 1,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add 1 $cardA $portA 1 $cardB $portB
map add 1 $cardB $portB 1 $cardA $portA
if {[ixStartRip pl1] != 0} {
puts "Could not start Rip for $pl1"
}
if {[ixStartRip pl2] != 0} {
puts "Could not start Rip for $pl2"
}
if {[ixStartRip pl3] != 0} {
```

```
puts "Could not start Rip for $pl3"
}
if {[ixStartRip pl4] != 0} {
puts "Could not start Rip for $pl4"
}
if {[ixStartRip one2oneArray] != 0} {
puts "Could not start Rip for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopRip

NAME - ixStartRipng

ixStartRipng — start RIPng on a group of ports simultaneously.

SYNOPSIS

ixStartRipng *portList*

DESCRIPTION

The **ixStartRipng** command sends a message to the IxServer to start protocol server RIPng operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLE

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list 1,$cardA,$portA]
set pl2 [list 1,$cardA,$portA 1,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list 1 $cardB $portB]]
set pl4 [list [list 1,$cardA,$portA] [list 1,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add 1 $cardA $portA 1 $cardB $portB
map add 1 $cardB $portB 1 $cardA $portA
if {[ixStartRipng pl1] != 0} {
puts "Could not start Ripng for $pl1"
}
if {[ixStartRipng pl2] != 0} {
puts "Could not start Ripng for $pl2"
}
if {[ixStartRipng pl3] != 0} {
```

```
puts "Could not start Ripng for $pl3"
}
if {[ixStartRipng pl4] != 0} {
puts "Could not start Ripng for $pl4"
}
if {[ixStartRipng one2oneArray] != 0} {
puts "Could not start Ripng for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopRipng

NAME - ixStartRsvp

ixStartRsvp — start RSVP on a group of ports simultaneously.

SYNOPSIS

ixStartRsvp *portList*

DESCRIPTION

The **ixStartRsvp** command sends a message to the IxServer to start protocol server RSVP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartRsvp pl1] != 0} {
puts "Could not start Rsvp for $pl1"
}
if {[ixStartRsvp pl2] != 0} {
puts "Could not start Rsvp for $pl2"
```

```
}
if {[ixStartRsvp pl3] != 0} {
puts "Could not start Rsvp for $pl3"
}
if {[ixStartRsvp pl4] != 0} {
puts "Could not start Rsvp for $pl4"
}
if {[ixStartRsvp one2oneArray] != 0} {
puts "Could not start Rsvp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopRsvp

NAME - ixStartStp

ixStartStp — start STP on a group of ports simultaneously.

SYNOPSIS

ixStartStp *portList*

DESCRIPTION

The **ixStartStp** command sends a message to the IxServer to start protocol server RSVP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStartStp pl1] != 0} {
puts "Could not start Stp for $pl1"
}
if {[ixStartStp pl2] != 0} {
puts "Could not start Stp for $pl2"
```

```
}
if {[ixStartStp pl3] != 0} {
puts "Could not start Stp for $pl3"
}
if {[ixStartStp pl4] != 0} {
puts "Could not start Stp for $pl4"
}
if {[ixStartStp one2oneArray] != 0} {
puts "Could not start Stp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStopStp

NAME - ixStopBFD

ixStopBFD — stop BFD on a group of ports simultaneously.

SYNOPSIS

ixStopBFD *portList*

DESCRIPTION

The **ixStopBFD** command sends a message to the IxServer to stop protocol server BFD operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after

set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB $portB]]
set pl4 [list [list $chas,$cardA,$portA] [list $chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $:: ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $:: ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixstopBfd pl1] != 0} {
puts "Could not stop Bfd for $pl1"
}
if {[ixstopBfd pl2] != 0} {
puts "Could not stop Bfd for $pl2"
}
if {[ixstopBfd pl3] != 0} {
```

```
puts "Could not stop Bfd for $pl3"
}
if {[ixstopBfd pl4] != 0} {
puts "Could not stop Bfd for $pl4"
}
if {[ixstopBfd one2oneArray] != 0} {
puts "Could not stop Bfd for $one2oneArray" }
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartBFD

NAME - ixStopBGP4

ixStopBGP4 — stop BGP4 on a group of ports simultaneously.

SYNOPSIS

ixStopBGP4 *portList*

DESCRIPTION

The **ixStopBGP4** command sends a message to the IxServer to stop protocol server BGP4 operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message. 1 Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list $chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStopBGP4 pl1] != 0} {
puts "Could not Stop BGP4 for $pl1"
}
if {[ixStopBGP4 pl2] != 0} {
puts "Could not Stop BGP4 for $pl2"
}
```

```
if {[ixStopBGP4 pl3] != 0} {
puts "Could not Stop BGP4 for $pl3"
}
if {[ixStopBGP4 pl4] != 0} {
puts "Could not Stop BGP4 for $pl4"
}
if {[ixStopBGP4 one2oneArray] != 0} {
puts "Could not Stop BGP4 for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartBGP4

NAME - ixStopCfm

ixStopCfm — stop CFM on a group of ports simultaneously.

SYNOPSIS

ixStopCfm *portList*

DESCRIPTION

The **ixStopCfm** command sends a message to the IxServer to stop protocol server CFM operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixstopCfm pl1] != 0} {
puts "Could not stop Cfm for $pl1"
}
if {[ixstopCfm pl2] != 0} {
puts "Could not stop Cfm for $pl2"
```

```
}
if {[ixstopCfm pl3] != 0} {
puts "Could not stop Cfm for $pl3"
}
if {[ixstopCfm pl4] != 0} {
puts "Could not stop Cfm for $pl4"
}
if {[ixstopCfm one2oneArray] != 0} {
puts "Could not stop Cfm for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartCfm

NAME - ixStopEigrp

ixStopEigrp — stop EIGRP on a group of ports simultaneously.

SYNOPSIS

ixStopEigrp *portList*

DESCRIPTION

The **ixStopEigrp** command sends a message to the IxServer to stop protocol server EIGRP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStopEigrp pl1] != 0} {
puts "Could not Stop Eigrp for $pl1"
}
if {[ixStopEigrp pl2] != 0} {
puts "Could not Stop Eigrp for $pl2"
```

```
}
if {[ixStopEigrp pl3] != 0} {
puts "Could not Stop Eigrp for $pl3"
}
if {[ixStopEigrp pl4] != 0} {
puts "Could not Stop Eigrp for $pl4"
}
if {[ixStopEigrp one2oneArray] != 0} {
puts "Could not Stop Eigrp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartEigrp

NAME - ixStopIsis

ixStopIsis — stop ISIS on a group of ports simultaneously.

SYNOPSIS

ixStopIsis *portList*

DESCRIPTION

The **ixStopIsis** command sends a message to the IxServer to stop protocol server ISIS operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal
set host galaxy
set username user
# Check if we’re running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStopIsis pl1] != 0} {
puts "Could not Stop Isis for $pl1"
}
if {[ixStopIsis pl2] != 0} {
puts "Could not Stop Isis for $pl2"
```

```
}
if {[ixStopIsis pl3] != 0} {
puts "Could not Stop Isis for $pl3"
}
if {[ixStopIsis pl4] != 0} {
puts "Could not Stop Isis for $pl4"
}
if {[ixStopIsis one2oneArray] != 0} {
puts "Could not Stop Isis for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartIsis

NAME - ixStopLdp

ixStopLdp — stop LDP on a group of ports simultaneously.

SYNOPSIS

ixStopLdp *portList*

DESCRIPTION

The **ixStopLdp** command sends a message to the IxServer to stop protocol server LDP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixstopLdp pl1] != 0} {
puts "Could not stop Ldp for $pl1"
}
if {[ixstopLdp pl2] != 0} {
puts "Could not stop Ldp for $pl2"
```

```
}
if {[ixstopLdp pl3] != 0} {
puts "Could not stop Ldp for $pl3"
}
if {[ixstopLdp pl4] != 0} {
puts "Could not stop Ldp for $pl4"
}
if {[ixstopLdp one2oneArray] != 0} {
puts "Could not stop Ldp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartLdp

NAME - ixStopMld

ixStopMld — stop MLD on a group of ports simultaneously.

SYNOPSIS

ixStopMld *portList*

DESCRIPTION

The **ixStopMld** command sends a message to the IxServer to stop protocol server MLD operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixstopMld pl1] != 0} {
puts "Could not stop Mld for $pl1"
}
if {[ixstopMld pl2] != 0} {
puts "Could not stop Mld for $pl2"
```

```
}
if {[ixstopMld pl3] != 0} {
puts "Could not stop Mld for $pl3"
}
if {[ixstopMld pl4] != 0} {
puts "Could not stop Mld for $pl4"
}
if {[ixstopMld one2oneArray] != 0} {
puts "Could not stop Mld for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartMld

NAME - ixStopOspf

ixStopOspf — stop OSPF on a group of ports simultaneously.

SYNOPSIS

ixStopOspf *portList*

DESCRIPTION

The **ixStopOspf** command sends a message to the IxServer to stop protocol server OSPF operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixstopOspf pl1] != 0} {
puts "Could not stop Ospf for $pl1"
}
if {[ixstopOspf pl2] != 0} {
puts "Could not stop Ospf for $pl2"
```

```
}
if {[ixstopOspf pl3] != 0} {
puts "Could not stop Ospf for $pl3"
}
if {[ixstopOspf pl4] != 0} {
puts "Could not stop Ospf for $pl4"
}
if {[ixstopOspf one2oneArray] != 0} {
puts "Could not stop Ospf for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartOspf

NAME - ixStopOspfV3

ixStopOspfV3 — stop OSPFv3 on a group of ports simultaneously.

SYNOPSIS

ixStopOspfV3 *portList*

DESCRIPTION

The **ixStopOspfV3** command sends a message to the IxServer to stop protocol server OSPFv3 operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. E.g. *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixstopOspfV3 pl1] != 0} {
puts "Could not stop OspfV3 for $pl1"
}
if {[ixstopOspfV3 pl2] != 0} {
puts "Could not stop OspfV3 for $pl2"
```

```
}
if {[ixstopOspfV3 pl3] != 0} {
puts "Could not stop OspfV3 for $pl3"
}
if {[ixstopOspfV3 pl4] != 0} {
puts "Could not stop OspfV3 for $pl4"
}
if {[ixstopOspfV3 one2oneArray] != 0} {
puts "Could not stop OspfV3 for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartOspfV3

NAME - ixStopPimsm

ixStopPimsm — stop PIM-SM on a group of ports simultaneously.

SYNOPSIS

ixStopPimsm *portList*

DESCRIPTION

The **ixStopPimsm** command sends a message to the IxServer to stop protocol server PIM-SM operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. E.g. *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStopPimsm pl1] != 0} {
puts "Could not Stop Pimsm for $pl1"
}
if {[ixStopPimsm pl2] != 0} {
puts "Could not Stop Pimsm for $pl2"
```

```
}
if {[ixStopPimsm pl3] != 0} {
puts "Could not Stop Pimsm for $pl3"
}
if {[ixStopPimsm pl4] != 0} {
puts "Could not Stop Pimsm for $pl4"
}
if {[ixStopPimsm one2oneArray] != 0} {
puts "Could not Stop Pimsm for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartPimsm

NAME - ixStopRip

ixStopRip — stop RIP on a group of ports simultaneously.

SYNOPSIS

ixStopRip *portList*

DESCRIPTION

The **ixStopRip** command sends a message to the IxServer to stop protocol server RIP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. E.g. *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list 1,$cardA,$portA]
set pl2 [list 1,$cardA,$portA 1,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list 1 $cardB $portB]]
set pl4 [list [list 1,$cardA,$portA] [list 1,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add 1 $cardA $portA 1 $cardB $portB
map add 1 $cardB $portB 1 $cardA $portA
if {[ixStopRip pl1] != 0} {
puts "Could not Stop Rip for $pl1"
}
if {[ixStopRip pl2] != 0} {
puts "Could not Stop Rip for $pl2"
}
if {[ixStopRip pl3] != 0} {
```

```
puts "Could not Stop Rip for $pl3"
}
if {[ixStopRip pl4] != 0} {
puts "Could not Stop Rip for $pl4"
}
if {[ixStopRip one2oneArray] != 0} {
puts "Could not Stop Rip for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

ixStartRip

NAME - ixStopRipng

ixStopRipng — stop RIPng on a group of ports simultaneously.

SYNOPSIS

ixStopRipng *portList*

DESCRIPTION

The **ixStopRipng** command sends a message to the IxServer to stop protocol server RIPng operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. E.g. *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLE

```
package require IxTclHal

set host galaxy

ixInitialize $host

set chas 1

set cardA 1

set portA 1

set cardB 1

set portB 2

set pl1 [list 1,$cardA,$portA]

set pl2 [list 1,$cardA,$portA 1,$cardB,$portB]

set pl3 [list [list $chas $cardA $portA] [list 1 $cardB $portB]]

set pl4 [list [list 1,$cardA,$portA] [list 1,$cardB,$portB]]

map new -type one2one
map config -type one2one
map add 1 $cardA $portA 1 $cardB $portB
map add 1 $cardB $portB 1 $cardA $portA
if {[ixStopRipng pl1] != 0} {
ixPuts "Could not start Ripng for $pl1"
}
if {[ixStopRipng pl2] != 0} {
ixPuts "Could not start Ripng for $pl2"
}
if {[ixStopRipng pl3] != 0} {
ixPuts "Could not start Ripng for $pl3"
}
if {[ixStopRipng pl4] != 0} {
ixPuts "Could not start Ripng for $pl4"
}
if {[ixStopRipng one2oneArray] != 0} {
ixPuts "Could not start Ripng for $one2oneArray"
}
```

SEE ALSO

NAME - ixStartRipng

NAME - ixStopRsvp

ixStopRsvp — stop RSVP on a group of ports simultaneously.

SYNOPSIS

ixStopRsvp *portList*

DESCRIPTION

The **ixStopRsvp** command sends a message to the IxServer to stop protocol server RSVP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. E.g. *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStopRsvp pl1] != 0} {
puts "Could not Stop Rsvp for $pl1"
}
if {[ixStopRsvp pl2] != 0} {
puts "Could not Stop Rsvp for $pl2"
```

```
}
if {[ixStopRsvp pl3] != 0} {
puts "Could not Stop Rsvp for $pl3"
}
if {[ixStopRsvp pl4] != 0} {
puts "Could not Stop Rsvp for $pl4"
}
if {[ixStopRsvp one2oneArray] != 0} {
puts "Could not Stop Rsvp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartRsvp

NAME - ixStopStp

ixStopStp — stops STP on a group of ports simultaneously.

SYNOPSIS

ixStopStp *portList*

DESCRIPTION

The **ixStopStp** command sends a message to the IxServer to stop protocol server STP operation on a group of ports simultaneously. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. For example, *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {
```

```
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
set pl1 [list $chas,$cardA,$portA]
set pl2 [list $chas,$cardA,$portA $chas,$cardB,$portB]
set pl3 [list [list $chas $cardA $portA] [list $chas $cardB
$portB]]
set pl4 [list [list $chas,$cardA,$portA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
if {[ixStopStp pl1] != 0} {
puts "Could not Stop Stp for $pl1"
}
if {[ixStopStp pl2] != 0} {
puts "Could not Stop Stp for $pl2"
```

```
}
if {[ixStopStp pl3] != 0} {
puts "Could not Stop Stp for $pl3"
}
if {[ixStopStp pl4] != 0} {
puts "Could not Stop Stp for $pl4"
}
if {[ixStopStp one2oneArray] != 0} {
puts "Could not Stop Stp for $one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $pl4
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - ixStartStp

NAME - ixTransmitIgmpJoin/ixStartIgmp

ixTransmitIgmpJoin/ixStartIgmp — transmit IGMP membership report messages on a group of ports simultaneously.

SYNOPSIS

ixTransmitIgmpJoin/ixStartIgmp *portList [groupId] [create] [destroy]*

DESCRIPTION

The **ixTransmitIgmpJoin/ixStartIgmp** command sends a message to the IxServer to start transmission of IGMP membership messages on a group of ports simultaneously using the protocol server. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. E.g. *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

groupId

(By value) The group number to be used in the join message. If omitted, the default value of 101064 will be used.

create

(By value) Create a new port group *(create)* or not *(nocreate)*. *(default = create)*

destroy

(By value) Clean up a created port group when command completes *(destroy)* or not *(nodestroy)*. *(default = destroy)*

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLES

```
package require IxTclHal

set host galaxy

set username user

# Check if we’re running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set cardA 1
set portA 1
set cardB 1
set portB 2
# Examples of four ways to make a port list
set portList1 [list $chas,$cardA,$cardA]
set portList2 [list $chas,$cardA,$cardA $chas,$cardB,$portB]
set portList3 [list [list $chas $cardA $cardA] [list $chas $cardB
$portB]]
set portList4 [list [list $chas,$cardA,$cardA] [list
$chas,$cardB,$portB]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $portList4] {
ixPuts $::ixErrorInfo
return 1
}
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
port setDefault
port set $chas $cardA $portA
port set $chas $cardB $portB
ip setDefault
ip set $chas $cardA $portA
ip set $chas $cardB $portB
ixWritePortsToHardware one2oneArray
after 1000
if {[ixCheckLinkState one2oneArray] != 0} {
ixPuts "Link is not up"
}
if {[ixTransmitIgmpJoin portList1] != 0} {
ixPuts "Could not Transmit Igmp Join on $portList1"
}
if {[ixTransmitIgmpJoin portList2] != 0} {
ixPuts "Could not Transmit Igmp Join on $portList2"
}
if {[ixTransmitIgmpJoin portList3] != 0} {
ixPuts "Could not Transmit Igmp Join on $portList3"
}
if {[ixTransmitIgmpJoin portList4] != 0} {
ixPuts "Could not Transmit Igmp Join on $portList4"
}
if {[ixTransmitIgmpJoin ::one2oneArray] != 0} {
ixPuts "Could not Transmit Igmp Join on ::one2oneArray"
}
# Let go of the ports that we reserved
ixClearOwnership $portList4
# Disconnect from the chassis we're using
```

```
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectFromTclServer $host
}
```

SEE ALSO

NAME - ixTransmitIgmpJoin/ixStopIgmp

NAME - ixTransmitIgmpJoin/ixStopIgmp

ixTransmitIgmpJoin/ixStopIgmp — transmit IGMP leave messages on a group of ports simultaneously.

SYNOPSIS

ixTransmitIgmpJoin/ixStopIgmp *portList [groupId] [create] [destroy]*

DESCRIPTION

The **ixTransmitIgmpJoin/ixStopIgmp** command sends a message to the IxServer to start transmission of IGMP membership leave messages on a group of ports simultaneously using the protocol server. The ports may span over multiple chassis.

ARGUMENTS

portList

(By reference) The list of ports in one of the following formats:

one2oneArray, one2manyArray, many2oneArray, many2manyArray

Or a reference to a list. E.g. *pl* after
set pl {{1 1 1} {1 1 2} {1 1 3} {1 1 4}} -or-
set pl {1,1,1 1,1,2 1,1,3 1,1,4}

groupId

(By value) The group number to be used in the leave message. If omitted, the default value of 101064 will be used.

create

(By value) Create a new port group *(create)* or not *(nocreate)*. *(default = create)*

destroy

(By value) Clean up a created port group when command completes *(destroy)* or not *(nodestroy)*. *(default = destroy)*

RETURNS

0 - No error; the command was successfully delivered to the IxServer.

1 - Error; the command was delivered to the IxServer but it could not process the message.

EXAMPLE

```
package require IxTclHal

set host galaxy

ixInitialize $host

set chas 1
set cardA 1
set portA 1
set cardB 1
set portB 2
# Examples of four ways to make a port list
set portList1 [list $chas,$cardA,$cardA]
set portList2 [list $chas,$cardA,$cardA $chas,$cardB,$portB]
set portList3 [list [list $chas $cardA $cardA] [list $chas $cardB
$portB]]
set portList4 [list [list $chas,$cardA,$cardA] [list
$chas,$cardB,$portB]]
map new -type one2one
map config -type one2one
map add $chas $cardA $portA $chas $cardB $portB
map add $chas $cardB $portB $chas $cardA $portA
port setDefault
port set $chas $cardA $portA
port set $chas $cardB $portB
ip setDefault
ip set $chas $cardA $portA
ip set $chas $cardB $portB
ixWritePortsToHardware one2oneArray
after 1000
if {[ixCheckLinkState one2oneArray] != 0} {
ixPuts "Link is not up"
}
if {[ixTransmitIgmpLeave portList1] != 0} {
ixPuts "Could not Transmit Igmp Leave on $portList1"
}
if {[ixTransmitIgmpLeave portList2] != 0} {
ixPuts "Could not Transmit Igmp Leave on $portList2"
}
if {[ixTransmitIgmpLeave portList3] != 0} {
ixPuts "Could not Transmit Igmp Leave on $portList3"
}
if {[ixTransmitIgmpLeave portList4] != 0} {
ixPuts "Could not Transmit Igmp Leave on $portList4"
}
if {[ixTransmitIgmpLeave one2oneArray] != 0} {
ixPuts "Could not Transmit Igmp Leave on one2oneArray"
}
```

SEE ALSO

NAME - ixTransmitIgmpJoin/ixStartIgmp

Appendix B: IxTclHAL Protocol Server Commands

NAME - arpAddressTableEntry

arpAddressTableEntry — configures the ARP address table entry parameters.

SYNOPSIS

arpAddressTableEntry *subcommand options*

DESCRIPTION

The **arpAddressTableEntry** command is used to configure the ARP address table entry parameters.

STANDARD OPTIONS

ipAddress

(Read-only.) IP address. *(default = 0.0.0.0)*

macAddress

(Read-only.) MAC address. *(default = 00 00 00 00 00 00)*

COMMANDS

The **arpAddressTableEntry** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

arpAddressTableEntry cget option

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **arpAddressTableEntry** command.

arpAddressTableEntry config *option value*

Modifies the configuration options of the arpAddressTableEntry. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for arpAddressTableEntry.

arpAddressTableEntry get

Gets the current ARP table entry. Call this command before calling **arpAddressTableEntry** cget *option value* to get the value of the configuration option.

arpAddressTableEntry setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *NAME - arpServer*

SEE ALSO

NAME - arpServer

NAME - arpServer

arpServer — configures the ARP server parameters.

SYNOPSIS

arpServer *subcommand options*

DESCRIPTION

The **arpServer** command is used to configure the ARP server parameters.

STANDARD OPTIONS

mode

Enable the ARP mode. Mode includes

Option	Value	Usage
arpGatewayOnly	0	(default) Sends a single ARP request to each gateway IP address using the first IP address found in the IP table entry as the source address
arpLearnOnly	1	Sends ARP requests using all of the addresses found in the IP address table as source addresses. On per-port Linux CPU-based ports, in this mode a *sendArpRequest* will result in the transmission of an ARP request only if the ARP table does not have an updated entry.
arpGatewayAndLearn	2	Performs both the *arpGatewayOnly* and *arpLearnOnly* operations.

rate

Frame rate in frames per second. *(default = 100)*

requestRepeatCount

When *mode* is set to *arpLearnOnly* or *arpGatewayAndLearn*, each ARP request is repeated this number of times with request gaps dictated by *rate*. *(default = 3)*

retries

Number of retries. Valid range is 1 65535. *(default = 3)*

COMMANDS

The **arpServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

arpServer cget option

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **arpServer** command.

arpServer clearArpTable chasID cardID portID

Clears the ARP table for port *portID* on card *cardID*, chassis *chasID*. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- Network problem between the client and chassis.

arpServer config option value

Modifies the configuration options of the arpServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for arpServer.

arpServer get chasID cardID portID

Gets the current configuration of the arpServer frame for port *portID* on card *cardID*, chassis *chasID* from its hardware. This command should be called before **arpServer** cget *option value* to get the configuration option. Specific errors are:

- No connection to a chassis.
- Invalid port number.

arpServer getEntry ipAddr

Gets the entry with IP address *ipAddr* out of the ARP table. The data is available through the *Appendix B: IxTclHAL Protocol Server Commands* command. Specific errors are:

- There is no arp table in the Arp Server.
- There are no entries in the Arp Table.
- The specified IP address is not in the Arp Table.

arpServer getFirstEntry

Gets the first entry out of the ARP table. The data is available through the *Appendix B: IxTclHAL Protocol Server Commands* command. Specific errors are:

- There is no arp table in the Arp Server.
- There are no entries in the Arp Table.

arpServer getNextEntry

Gets the next entry out of the ARP table. The data is available through the *arpAddressTableEntry* command.

arpServer sendArpRequest chasID cardID portID

Sends ARP request for port indicated. On per-port Linux CPU-based ports, if *mode* is set to *arpLearnOnly*, this subcommand will result in the transmission of an ARP request only if the ARP table does not have an updated entry.

Specific errors are:

- No connection to a chassis.
- Invalid port number.

- The port is being used by another user.
- Network problem between the client and chassis.

arpServer set chasID cardID portID

Sets the configuration of the arpServer in IxHAL for port with id *portID* on card *cardID*, chassis *chasID* by reading the configuration option values set by the **arpServer** config *option value* command. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- Configured parameters are not valid for this setting.

arpServer setDefault

Sets default values for all configuration options.

EXAMPLES

```
package require IxTclHal
# In this example, we assume that Card 1, ports 1 and 2 are
directly
# connected or through a L2 switch
# We'll set up IP address tables for both and start the arp server
# on both ports. For each port, this should obtain the MAC address
of
# the other party.
set host galaxy
set username user
# Check if we're running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
# Assume card to be used is in slot 1
set card 1
set port1 1
set port2 2
set portList [list \
[list $chas $card $port1] \
[list $chas $card $port2] \
]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
# IP addresses
set port1IP 148.0.0.1
set port2IP 148.0.0.2
# And MAC addresses
set port1MAC {00 11 11 00 00 11}
set port2MAC {00 11 11 00 00 22}
# Configure Port 1
# Reset the port
port setFactoryDefaults $chas $card $port1
port setDefault
port set $chas $card $port1
# Set up IP table with our IP-MAC and Gateway (= port2)
ipAddressTable setDefault
ipAddressTable config -defaultGateway $port2IP

ipAddressTableItem setDefault

ipAddressTableItem config -fromIpAddress $port1IP

ipAddressTableItem config -fromMacAddress $port1MAC

ipAddressTableItem config -numAddresses 1

ipAddressTableItem set

ipAddressTable addItem

ipAddressTable set $chas $card $port1

# Clear and set up the ARP server

arpServer setDefault

arpServer clearArpTable $chas $card $port1

arpServer set $chas $card $port1

# Now start the Protocol Server to allow arp responses

protocolServer setDefault

protocolServer config -enableArpResponse true

protocolServer set $chas $card $port1

# Configure Port 2

# Reset the port

port setFactoryDefaults $chas $card $port2

port setDefault

port set $chas $card $port2

# Set up IP table with our IP-MAC and Gateway (= port1)

ipAddressTable setDefault

ipAddressTable config -defaultGateway $port1IP

ipAddressTableItem setDefault

ipAddressTableItem config -fromIpAddress $port2IP

ipAddressTableItem config -fromMacAddress $port2MAC

ipAddressTableItem config -numAddresses 1

ipAddressTableItem set

ipAddressTable addItem

ipAddressTable set $chas $card $port2

# Clear and set up the ARP server

arpServer setDefault

arpServer clearArpTable $chas $card $port2
arpServer set $chas $card $port2
# Now start the Protocol Server to allow arp responses
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer set $chas $card $port2
# Ready to go - send our configuration to the hardware
ixWritePortsToHardware portList
# And then tell each port to ARP for their respective gateways
arpServer sendArpRequest $chas $card $port1
arpServer sendArpRequest $chas $card $port2
# Wait for ARPs to be serviced
after 5000
# Port1: Get the ARP table, get the first entry and print the
entry
arpServer get $chas $card $port1
if {[arpServer getFirstEntry]} \
{
ixPuts "Port 1: No ARP table entries"
}
arpAddressTableEntry get
set ipi [arpAddressTableEntry cget -ipAddress]
set maca [arpAddressTableEntry cget -macAddress]
ixPuts "Port 1: $ipi = $maca"
# Port2: Get the ARP table, get the first entry and print the
entry
arpServer get $chas $card $port2
if {[arpServer getFirstEntry]} \
{
ixPuts "Port 2: No ARP table entries"
}
arpAddressTableEntry get
set ipi [arpAddressTableEntry cget -ipAddress]
set maca [arpAddressTableEntry cget -macAddress]
ixPuts "Port 2: $ipi = $maca"
```

```
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - bfdInterface

bfdInterface — configures an interface for a BFD router.

SYNOPSIS

bfdInterface *subcommand options*

DESCRIPTION

The *bfdInterface* holds the information related to a single interface on the simulated router. Refer to *bfdInterface* for an overview of this command.

STANDARD OPTIONS

desiredMinRxIntv

This option indicates the desired interval between received BFD control packets. *(default = 1,000)*

desiredTxIntv

This option indicates the desired interval between transmitted BFD control packets.*(default = 1,000)*

desiredEchoRxIntv

This option indicates the minimum interval between received BFD Echo packets that this interface is capable of supporting. If this value is zero, the transmitting system does not support the receipt of BFD Echo packets.

echoTxIntv

This option indicates the minimum interval that the interface would like to use when transmitting BFD Echo packets.

echoTimeOut

This is the minimum interval that the interface waits for a response to the last Echo packet sent out.

echoConfigureSrcIp true / false

If set to *True(1)*, the configure Source IP address option is enabled, and an IPv4 or IPv6 Source address can be configured for an Echo packet. If set to *False(0)*, the option is disabled.

echoSrcIPv4Addr

If *echoConfigureSrcIp* is enabled, this option is available for configuring an IPv4 source address for an Echo packet.

echoSrcIPv6Addr

If *echoConfigureSrcIp* is enabled, this option is available for configuring an IPv6 source address for an Echo packet.

enable

Enables the use of BFD interface.

enableCtrlPlaneIndependent

Set to 1 if the local system's BFD implementation is independent of the control plane.

enableDemandMode

Enables the demand mode. 1 indicates that the demand mode is enabled and 0 indicates that the demand mode is disabled.*(default = 0)*

flapTxIntvs

BFD sessions will flap every flapTxIntvs. (default = 0)

interfaceId

This is a local ID for the interface and is unique per router.

multiplier

Multiplier*intv defines the timeout period. *(default = 3)*

pollIntv

If in the Demand Mode, the polling will take place every pollIntv interval.

protocolInterfaceDescription

The name of the defined *interfaceEntry* which describes the host interface to be simulated.

COMMANDS

The *bfdInterface* command is invoked with the following subcommands. If no subcommand is specified, then a list of all subcommands available is returned.

bfdInterface addSession *sessionName*

Adds a session to the session list of an interface. If a session is added to an existing interface, then the bfdServer select should be called before the interface is selected by the get command.

NOTE: bfdServer write can be called immediately without calling the setInterface command. It will behave as addOnFly.

bfdInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the bfdInterface command.

bfdInterface clearAllSessions

Clears the BFD session list on the selected BFD interface.

bfdInterface config option value

Modifies the configuration options of the bfdInterface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bfdInterface.

bfdInterface delSession *[sessionName]*

Deletes a session from the session list of a selected router. If no sessionName is specified, then it deletes the current one.

NOTE: bfdServer write can be called immediately without calling the setInterface command. It will behave as delOnFly.

bfdInterface getFirstSession

Gets the first session from the BFD session list on the selected BFD interface and refreshes the options.

bfdInterface getNextSession

Gets the next BFD session from the BFD session list on the selected BFD interface and refreshes the options.

bfdInterface getSession

Gets a particular session from the BFD session list on the selected BFD interface and refreshes the options.

bfdInterface setDefault

Sets the options to default values.

bfdInterface setSession

Allows the configuration values for a session to be overwritten on the fly.

bfdInterface showSessionNames

Returns the names of the sessions in the list on the selected port. Calling *select* command is recommended before calling this command.

EXAMPLES

See examples under NAME - bfdServer.

SEE ALSO

NAME - bfdRouter, *NAME - bfdServer*, *NAME - bfdSession*, *NAME - bfdSessionLearnedInfo*

NAME - bfdRouter

bfdRouter — configures a BFD router.

SYNOPSIS

bfdRouter *subcommand options*

DESCRIPTION

The *bfdRouter* command represents an emulated router. In addition to some identifying options, it holds two lists for the router:

- Interfaces — router interface, which is constructed in the *bfdInterface* command.
- Sessions — BFD routes to be advertised by the simulated router, which is constructed in the *bfdInterface* command.

Routers defined in this command are added to an *bfdServer* using the *bfdServer addRouter* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on BFD testing with Ixia equipment. Refer to *bfdRouter* for an overview of this command.

STANDARD OPTIONS

enable

Enables or disables the simulated router.

routerId

The ID of the simulated router, which is expressed as an IP address.

COMMANDS

The **bfdRouter** command is invoked with the following subcommands. If no subcommand is specified, then a list of all subcommands available is returned.

bfdRouter addInterface interfaceName

Adds an interface to the interface list of a router.

NOTE: If an interface is added to an existing router, then before the router is selected by the get command, bfdServer select should be called. bfdServer write can be called immediately without calling the setRouter command. It will behave as addOnFly.

bfdRouter cget option

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the bfdRouter command.

bfdRouter clearAllInterfaces

Clears the BFD interface list on the selected router.

bfdRouter config option value

Modifies the configuration options of the bfdRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bfdRouter.

bfdRouter delInterface [interfaceName]

Deletes an interface from the interface list of a selected router. If no interfaceName is specified, then it deletes the current one.

NOTE: bfdServer write can be called immediately without calling the setRouter command. It will behave as delOnFly.

bfdRouter getFirstInterface

Gets the first interface from the interface list on the selected router and refreshes the options.

bfdRouter getFirstLearnedInfo

Gets the first BFD session learned information.

bfdRouter **getInterface** *interfaceName*

Gets the interfaceName on the selected router. If no interfaceName is specified, then the current one will be modified.

bfdRouter getLearnedInfoList

Gets the learned session information list. It should be called after the requestLearnedInfo command.

bfdRouter getNextInterface

Gets the next interface from the interface list on the selected router and refreshes the options.

bfdRouter getNextLearnedInfo

Gets the next BFD learned session information.

bfdRouter requestLearnedInfo

Request for learned info for all BFD sessions under this interface.

bfdRouter setInterface interfaceName

Edits on the fly interfaceName on the selected router and refreshes the options.

bfdRouter setDefault

Sets the options to default values. Adds an interface to the interface list of a router.

NOTE: If an interface is added to an existing router, then before the router is selected by the get command, the bfdServer select should be on. bfdServer write can be called immediately without calling the setRouter command.

bfdRouter showInterfaceName

Returns names of interfaces in the list on the selected router. Calling the *select* command getting the bridge is recommended before calling this command.

EXAMPLES

See examples under *NAME - bfdServer*.

SEE ALSO

NAME - bfdInterface, *NAME - bfdServer*, *NAME - bfdSession*, *NAME - bfdSessionLearnedInfo*

NAME - bfdServer

bfdServer — accesses the BFD component of the protocol server for a particular port.

SYNOPSIS

bfdServer *subcommand options*

DESCRIPTION

The *bfdServer* command is necessary in order to access the BFD protocol server for a particular port. The *select* subcommand **must** be used before all other BFD commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on BFD server testing with Ixia equipment. Refer to *bfdServer* for an overview of this command.

STANDARD OPTIONS

enableRateControl

Enables rate control for BFD.

interval

Sets the rate control interval.

packetsPerInterval

Sets the numbers of packets to be transmitted per rate control interval.

COMMANDS

The **bfdServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bfdServer addRouter routerName

Adds a router to the router list.

bfdServer cget option

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the bfdServer command.

bfdServer clearAllRouters

Clears the router list.

bfdServer config option value

Modifies the configuration options of the bfdServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bfdServer.

bfdServer delRouter [routerName]

Adds a router to the router list.

bfdServer getFirstRouter

Gets the first router from the router list and refreshes the options.

bfdServer getNextRouter

Gets the next router from the router list and refreshes the options.

bfdServer getRouter routerName

Gets routerName and refreshes the options.

bfdServer select chasID cardID portID

Selects the port.

bfdServer setRouter routerName

Edits on the fly "routerName." If no routerName is specified, the current one will be modified.

bfdServer showRouterNames

Returns names of routers in the list on the selected port. Calling select commands is recommended before calling this command.

bfdServer write

Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card, and port. Before using this command, use the *bfdServer select* command to select the port.

EXAMPLE

```
############################################################
# This Script has been generated by Ixia ScriptGen
# Software Version : ixos 5.00.301.143 EB
############################################################
package req IxTclHal
# Command Option Mode - Full (generate full configuration)
if {[isUNIX]} {
if {[ixConnectToTclServer loopback]} {
ixPuts "Error connecting to Tcl Server loopback "
return 1
}
}
######### Chassis list - {loopback} #########
ixConnectToChassis {loopback}
```

```
set owner “IxNetwork/ixin-santanup/spaul.4372”
ixLogin $owner
set portList {}
######### Chassis-loopback #########
chassis get “loopback”
set chassis [chassis cget -id]
######### Card Type : 10/100/1000 STXS4-256MB ############
set card 2
card setDefault
card config -txFrequencyDeviation 0
card set $chassis $card
card write $chassis $card
######### Chassis-loopback Card-2 Port-1 #########
set port 1
port setFactoryDefaults $chassis $card $port
port setPhyMode $::portPhyModeCopper $chassis $card
$port
port config -speed 1000
port config -duplex full
port config -flowControl false
port config -directedAddress “01 80 C2
00 00 01”
port config -multicastPauseAddress “01 80 C2
00 00 01”
port config -loopback false
port config -transmitMode
portTxPacketStreams
port config -receiveMode [expr
$::portRxFirstTimeStamp|$::portRxModeWidePacketGroup]
port config -autonegotiate true
port config -advertise100FullDuplex true
port config -advertise100HalfDuplex true
port config -advertise10FullDuplex true
port config -advertise10HalfDuplex true
port config -advertise1000FullDuplex true
port config -portMode
portEthernetMode
port config -rxTxMode gigNormal
port config -ignoreLink false
port config -advertiseAbilities
portAdvertiseNone
port config -timeoutEnable true
port config -negotiateMasterSlave 1
port config -masterSlave portSlave
port config -enableSimulateCableDisconnect false
port config -enableAutoDetectInstrumentation false
port config -enableRepeatableLastRandomPattern false
port config -transmitClockDeviation 0
port config -preEmphasis preEmphasis0
port config -MacAddress “00 de bb
00 00 01”
port config -DestMacAddress “00 de bb
00 00 02”
port config -name ““
port config -numAddresses 1
port config -enableManualAutoNegotiate false
port config -enablePhyPolling true
port set $chassis $card $port
stat setDefault
stat config -mode statNormal
stat config -enableProtocolServerStats true
stat config -enableArpStats true
stat config -enablePosExtendedStats true
stat config -enableTemperatureSensorsStats true
stat config -enableAtmOamStats false
stat config -enableDhcpStats false
stat config -enableDhcpV6Stats false
stat config -includeRprPayloadFcsInCrc true
stat config -enableValidStats false
stat config -enableBgpStats false
stat config -enableIcmpStats true
stat config -enableOspfStats false
stat config -enableIsisStats false
stat config -enableRsvpStats false
stat config -enableLdpStats false
stat config -enableIgmpStats false
stat config -enableOspfV3Stats false
stat config -enablePimsmStats false
stat config -enableMldStats false
stat config -enableStpStats false
stat config -enableEigrpStats false
stat set $chassis $card $port
packetGroup setDefault
packetGroup config -signatureOffset 48
packetGroup config -signature "08 71 18 05"
packetGroup config -insertSignature false
packetGroup config -ignoreSignature false
packetGroup config -groupId 0
packetGroup config -groupIdOffset 52
packetGroup config -enableGroupIdMask false
packetGroup config -enableInsertPgid true
packetGroup config -groupIdMask 0
packetGroup config -latencyControl cutThrough
packetGroup config -preambleSize 8
packetGroup config -sequenceNumberOffset 44
packetGroup config -sequenceErrorThreshold 2
packetGroup config -insertSequenceSignature false
packetGroup config -allocateUdf true
packetGroup config -enableSignatureMask false
packetGroup config -signatureMask "00 00 00 00"
packetGroup config -enableRxFilter false
packetGroup config -headerFilter "00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00”
packetGroup config -headerFilterMask “00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00”
packetGroup config -enable128kBinMode false
packetGroup config -enableTimeBins false
packetGroup config -numPgidPerTimeBin 32
packetGroup config -numTimeBins 10
packetGroup config -timeBinDuration 1000000
packetGroup config -enableLatencyBins false
packetGroup config -latencyBinList ““
packetGroup config -groupIdMode
packetGroupCustom
packetGroup config -sequenceCheckingMode seqThreshold
packetGroup config -multiSwitchedPathMode
seqSwitchedPathPGID
packetGroup setRx $chassis $card $port
ipAddressTable setDefault
ipAddressTable config -defaultGateway
“0.0.0.0”
ipAddressTable set $chassis $card $port
arpServer setDefault
arpServer config -retries 3
arpServer config -mode
arpGatewayOnly
arpServer config -rate
2083333
arpServer config -requestRepeatCount 3
arpServer set $chassis $card $port
interfaceTable select $chassis $card $port
interfaceTable setDefault
interfaceTable config -dhcpV4RequestRate
0
interfaceTable config -dhcpV6RequestRate
0
```

```
interfaceTable config -
dhcpV4MaximumOutstandingRequests 100
interfaceTable config -
dhcpV6MaximumOutstandingRequests 100
interfaceTable set
interfaceTable clearAllInterfaces
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress {1.1.1.2}
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress {1.1.1.1}
interfaceEntry addItem addressTypeIpV4
dhcpV4Properties removeAllTlvs
dhcpV4Properties setDefault
dhcpV4Properties config -clientId ""
dhcpV4Properties config -serverId "0.0.0.0"
dhcpV4Properties config -vendorId ""
dhcpV4Properties config -renewTimer 0
dhcpV4Properties config -relayAgentAddress "0.0.0.0"
dhcpV4Properties config -relayDestinationAddress
"255.255.255.255"
dhcpV6Properties removeAllTlvs
dhcpV6Properties setDefault
dhcpV6Properties config -iaType
dhcpV6IaTypePermanent
dhcpV6Properties config -iaId 0
dhcpV6Properties config -renewTimer 0
dhcpV6Properties config -relayLinkAddress
"0:0:0:0:0:0:0:0"
dhcpV6Properties config -relayDestinationAddress
"FF05:0:0:0:0:0:1:3"
interfaceEntry config -enable true
interfaceEntry config -description
{ProtocolInterface - 100:01 - 1}
interfaceEntry config -macAddress {00 00 01 AE
9A 11}
interfaceEntry config -eui64Id {02 00 01 FF
FE AE 9A 11}
interfaceEntry config -atmEncapsulation
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmMode -1
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableDhcpV6 false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 1
interfaceEntry config -vlanPriority 5
interfaceTable addInterface 0
bfdServer select $chassis $card $port
bfdServer clearAllRouters
bfdSession setDefault
bfdSession config -enable true
bfdSession config -remoteDiscLearned 1
bfdSession config -iPType 17
bfdSession config -localBFDAddress “0.0.0.0”
bfdSession config -remoteBFDAddress “1.1.1.2”
bfdSession config -myDisc 1
bfdSession config -remoteDisc 1
bfdSession config -enableAutoChooseSrc true
bfdSession config -sessionType 1
bfdInterface addSession Session1
bfdInterface setDefault
bfdInterface config -enable true
bfdInterface config -interfaceId 1
bfdInterface config -desiredMinRxIntv
```

```
1000
bfdInterface config -desiredTxIntv
1000
bfdInterface config -desiredEchoIntv 0
bfdInterface config -enableDemandMode
false
bfdInterface config -enableCtrlPlaneIndependent
false
bfdInterface config -pollIntv 0
bfdInterface config -multiplier 3
bfdInterface config -flapTxIntvs 0
bfdInterface config -protocolInterfaceDescription
"ProtocolInterface - 100:01 - 1"
bfdRouter addInterface Interface1
bfdRouter setDefault
bfdRouter config -routerId "100.1.0.1"
bfdRouter config -enabled true
bfdServer addRouter Router1
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse
false
protocolServer config -enableIgmpQueryResponse
false
protocolServer config -enableOspfService
false
protocolServer config -enableBgp4Service
false
protocolServer config -enableIsisService
false
protocolServer config -enableRsvpService
false
protocolServer config -enableRipService
false
```

```
protocolServer config -enableLdpService
false
protocolServer config -enableRipngService
false
protocolServer config -enableMldService
false
protocolServer config -enableOspfV3Service
false
protocolServer config -enablePimsmService
false
protocolServer config -enableStpService
false
protocolServer config -enableEigrpService
false
protocolServer config -enableBfdService true
protocolServer config -enableBgp4CreateInterface
false
protocolServer config -enableIsisCreateInterface
false
protocolServer config -enableOspfCreateInterface
false
protocolServer config -enableRipCreateInterface
false
protocolServer config -enableRsvpCreateInterface
false
protocolServer config -enableIgmpCreateInterface
false
protocolServer set $chassis $card $port
flexibleTimestamp setDefault
flexibleTimestamp config -type
timestampBeforeCrc
flexibleTimestamp config -offset 23
flexibleTimestamp set $chassis $card $port
capture setDefault
capture config -fullAction
lock
capture config -sliceSize
8191
capture config -sliceOffset 0
capture config -trigger -1
capture config -filter -1
capture config -captureMode
captureTriggerMode
capture config -continuousFilter 0
capture config -beforeTriggerFilter
captureBeforeTriggerNone
capture config -afterTriggerFilter
captureAfterTriggerFilter
capture config -triggerPosition 1.0
capture config -enableSmallPacketCapture
false
capture set $chassis $card $port
filter setDefault
filter config -captureTriggerDA
anyAddr
filter config -captureTriggerSA
anyAddr
filter config -captureTriggerPattern
anyPattern
filter config -captureTriggerError
errAnyFrame
filter config -captureTriggerFrameSizeEnable
false
filter config -captureTriggerFrameSizeFrom 12
filter config -captureTriggerFrameSizeTo 12
filter config -captureFilterDA
anyAddr
filter config -captureFilterSA
```

```
anyAddr

filter config -captureFilterPattern

anyPattern

filter config -captureFilterError

errAnyFrame

filter config -captureFilterFrameSizeEnable

false

filter config -captureFilterFrameSizeFrom 12

filter config -captureFilterFrameSizeTo 12

filter config -userDefinedStat1DA

anyAddr

filter config -userDefinedStat1SA

anyAddr

filter config -userDefinedStat1Pattern

anyPattern

filter config -userDefinedStat1Error

errAnyFrame

filter config -userDefinedStat1FrameSizeEnable

false

filter config -userDefinedStat1FrameSizeFrom 12

filter config -userDefinedStat1FrameSizeTo 12

filter config -userDefinedStat2DA

anyAddr

filter config -userDefinedStat2SA

anyAddr

filter config -userDefinedStat2Pattern

anyPattern

filter config -userDefinedStat2Error

errAnyFrame

filter config -userDefinedStat2FrameSizeEnable 0

filter config -userDefinedStat2FrameSizeFrom 12

filter config -userDefinedStat2FrameSizeTo 12

filter config -asyncTrigger1DA

anyAddr
```

```
filter config -asyncTrigger1SA
anyAddr
filter config -asyncTrigger1Pattern
anyPattern
filter config -asyncTrigger1Error
errAnyFrame
filter config -asyncTrigger1FrameSizeEnable
false
filter config -asyncTrigger1FrameSizeFrom 12
filter config -asyncTrigger1FrameSizeTo 12
filter config -asyncTrigger2DA
anyAddr
filter config -asyncTrigger2SA
anyAddr
filter config -asyncTrigger2Pattern
anyPattern
filter config -asyncTrigger2Error
errAnyFrame
filter config -asyncTrigger2FrameSizeEnable
false
filter config -asyncTrigger2FrameSizeFrom 12
filter config -asyncTrigger2FrameSizeTo 12
filter config -captureTriggerEnable
true
filter config -captureFilterEnable
true
filter config -userDefinedStat1Enable
false
filter config -userDefinedStat2Enable
false
filter config -asyncTrigger1Enable
false
filter config -asyncTrigger2Enable
false
```

```
filter set $chassis $card $port
filterPallette setDefault
filterPallette config -DA1 “00
00 00 00 00 00”
filterPallette config -DAMask1 “00
00 00 00 00 00”
filterPallette config -DA2 “00
00 00 00 00 00”
filterPallette config -DAMask2 “00
00 00 00 00 00”
filterPallette config -SA1 “00
00 00 00 00 00”
filterPallette config -SAMask1 “00
00 00 00 00 00”
filterPallette config -SA2 “00
00 00 00 00 00”
filterPallette config -SAMask2 “00
00 00 00 00 00”
filterPallette config -pattern1 “DE
ED EF FE AC CA”
filterPallette config -patternMask1 “00
00 00 00 00 00”
filterPallette config -pattern2 00
filterPallette config -patternMask2 00
filterPallette config -patternOffset1 12
filterPallette config -patternOffset2 12
filterPallette config -matchType1
matchUser
filterPallette config -matchType2
matchUser
filterPallette config -patternOffsetType1
filterPalletteOffsetStartOfFrame
filterPallette config -patternOffsetType2
filterPalletteOffsetStartOfFrame
```

```
filterPallette config -gfpErrorCondition
gfpErrorsOr
filterPallette config -enableGfptHecError true
filterPallette config -enableGfpeHecError true
filterPallette config -enableGfpPayloadCrcError true
filterPallette config -enableGfpBadFcsError true
filterPallette set $chassis $card $port
lappend portList [list $chassis $card
$port]
ixWritePortsToHardware portList
ixCheckLinkState portList
#################################################################
##
######### Generating streams for all the ports from above
#########
#################################################################
##
######### Chassis-loopback Card-2 Port-1 #########
chassis get "loopback"
set chassis [chassis cget -id]
set card 2
set port 1
port reset $chassis $card $port
ixWriteConfigToHardware portList -noProtocolServer
```

NAME - bfdSession

bfdSession — configures a session under a BFD interface.

SYNOPSIS

bfdSession *subcommand options*

DESCRIPTION

The *bfdSession* command holds the information related to a single session under an interface for a simulated BFD router. Sessions are added under the *bfdInterface* using the *bfdInterface addSession* command. There can be multiple sessions to different neighbors configured under the bfdInterface object.

Refer to *bfdSession* for an overview of this command.

STANDARD OPTIONS

enable

Enables the use of this route range for the simulated router. The default is disable.

enableAutoChooseSrc

If true, enables the session to automatically choose the source IP address for the BFD session.

iPType

The session is created with the remote IP. IPv4 or IPv6 (*default* = *IPv4*).

localBFDAddress

The first IP address that will be used for simulated routers. IPv4 or IPv6.

myDisc

Needs to be unique in node. This option is used to demultiplex multiple BFD sessions.

name

(Read-only.) The name of the session that will be used as a unique key to retrieve the object.

remoteBFDaddress

The remote address in which the BFD session is active.

remoteDisc

This is a remote discriminator of peer. (default = 0). If it is set to 0, then the remote discriminator will be learned.

remoteDiscLearned

The default is 0. If it is set to 0, then the Remote Discriminator will be learned.

sessionType

Indicates whether the mode is a single-hop session or a multihop session.

COMMANDS

The *bfdSession* command is invoked with the following subcommands. If no subcommand is specified, then a list of all subcommands available is returned.

bfdSession cget option

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the bfdSession command.

bfdSession config option value

Modifies the configuration options of the bfdSession. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bfdSession.

bfdSession setDefault

Sets the options to default values.

EXAMPLES

See examples under NAME - bfdServer.

SEE ALSO

NAME - bfdRouter, *NAME - bfdServer*, *NAME - bfdSessionLearnedInfo*, *NAME - bfdInterface*

NAME - bfdSessionLearnedInfo

bfdSessionLearnedInfo — views retrieved learned session information.

SYNOPSIS

bfdSessionLearnedInfo *subcommand options*

DESCRIPTION

The *bfd SessionLearnedInfo* command is used to look at the session information that is retrieved when using *requestLearnedInfo* and *getLearnedInfoList* sub-commands of the *bfdRouter* command.

Refer to *bfdSessionLearnedInfo* for an overview of this command.

STANDARD OPTIONS

desMinTxIntv

(Read-only) Indicates the desired interval (in ms) between transmitted BFD control packets.

myDisc

(Read-only) Identifies the session uniquely. This option is used to demultiplex multiple BFD sessions.

myIpAddress

(Read-only) The local IP address being used by the configured or auto-created BFD session.

peerDisc

(Read-only)The discriminator received from the peer. This field reflects back the received value of myDisc.

peerFlags

(Read-only) The peer flags are received. The available options are:

Option	Value	Usage
Demand Mode	0x02	
Authentication	0x04	
Control Plane Independent	0x08	
Final	0x10	
Poll	0x11	

peerIPAddress

(Read-only) The peer IP address (Ipv4/Ipv6) may be learned.

peerState

(Read-only) The peer state is received from the peer (0 AdminDown, 1 Down, 2 Init, 3 Up).

peerUPtime

(Read-only) The time since the session last went to UP state.

protosUsingSession

(Read-only) The protocols registered for this session. Containing one or more of the following protocols:

- None
- BGP
- OSPF
- OSPFv3
- EIGRP
- ISIS

reqMinEchoIntv

(Read-only) The required minimum echo interval is received from the peer.

reqMinRxIntv

(Read-only) The required minimum receive interval is received from the peer.

sessionType

(Read-only) Indicates whether the mode is a single-hop session or a multihop session.

COMMANDS

The *bfdSession* command is invoked with the following subcommands. If no subcommand is specified, then a list of all subcommands available is returned.

bfdSessionLearnedInfo cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the bfdSessionLearnedInfo command.

bfdSessionLearnedInfo setDefault

Sets the options to default values.

EXAMPLES

```
################################################################
# This Script has been generated by Ixia ScriptGen
# Software Version : ixos 5.00.301.170 EB
################################################################
```

```
package req IxTclHal
# Command Option Mode - Full (generate full configuration)
if {[isUNIX]} {
if {[ixConnectToTclServer abhijit-400t]} {
ixPuts "Error connecting to Tcl Server abhijit-400t "
return 1
}
}
######### Chassis list - {abhijit-400t} #########
ixConnectToChassis {abhijit-400t}
set owner "IxNetwork/ixin-abhijitroy/ARoy.940"
ixLogin $owner
set portList {}
######### Chassis-abhijit-400t #########
chassis get "abhijit-400t"
set chassis [chassis cget -id]
######### Card Type : 10/100/1000 TXS4 ############
set card 1
card setDefault
card config -txFrequencyDeviation
0
card set $chassis $card
card write $chassis $card
######### Chassis-abhijit-400t Card-1 Port-1 #########
set port 1
port setFactoryDefaults $chassis $card $port
port config -speed 1000
port config -duplex full
port config -flowControl false
port config -directedAddress "01 80 C2 00
00 01"
port config -multicastPauseAddress "01 80 C2 00
00 01"
port config -loopback false
port config -transmitMode portTxPacket-
Streams
port config -receiveMode [expr $::portRxFirstTimeStamp|$::
portRxModeWidePacketGroup]
port config -autonegotiate true
port config -advertise100FullDuplex true
port config -advertise100HalfDuplex true
port config -advertise10FullDuplex true
port config -advertise10HalfDuplex true
port config -advertise1000FullDuplex true
port config -portMode portEthernetMode
port config -rxTxMode gigNormal
port config -ignoreLink false
port config -advertiseAbilities portAdvertise-
None
port config -timeoutEnable true
port config -negotiateMasterSlave 1
port config -masterSlave portSlave
port config -enableSimulateCableDisconnect false
port config -enableAutoDetectInstrumentation false
port config -enableRepeatableLastRandomPattern false
port config -transmitClockDeviation 0
port config -preEmphasis preEmphasis0
port config -MacAddress “00 de bb 00
00 01”
port config -DestMacAddress “00 de bb 00
00 02”
port config -name ““
port config -numAddresses 1
port config -enableManualAutoNegotiate false
port config -enablePhyPolling true
port set $chassis $card $port
stat setDefault
stat config -mode statNormal
stat config -enableProtocolServerStats true
stat config -enableArpStats true
stat config -enablePosExtendedStats true
stat config -enableTemperatureSensorsStats true
stat config -enableAtmOamStats false
stat config -enableDhcpStats false
stat config -enableDhcpV6Stats false
stat config -includeRprPayloadFcsInCrc true
stat config -enableValidStats false
stat config -enableBgpStats false
stat config -enableIcmpStats true
stat config -enableOspfStats false
stat config -enableIsisStats false
stat config -enableRsvpStats false
stat config -enableLdpStats false
stat config -enableIgmpStats false
stat config -enableOspfV3Stats false
stat config -enablePimsmStats false
stat config -enableMldStats false
stat config -enableStpStats false
stat config -enableEigrpStats false
stat set $chassis $card $port
packetGroup setDefault
packetGroup config -signatureOffset 48
packetGroup config -signature “08 71 18 05”
packetGroup config -insertSignature false
packetGroup config -ignoreSignature false
packetGroup config -groupId 0
packetGroup config -groupIdOffset 52
packetGroup config -enableGroupIdMask false
packetGroup config -enableInsertPgid true
packetGroup config -groupIdMask 0
packetGroup config -latencyControl cutThrough
packetGroup config -preambleSize 8
packetGroup config -sequenceNumberOffset 44
packetGroup config -sequenceErrorThreshold 2
packetGroup config -insertSequenceSignature false
packetGroup config -allocateUdf true
packetGroup config -enableSignatureMask false
packetGroup config -signatureMask “00 00 00 00”
packetGroup config -enableRxFilter false
packetGroup config -headerFilter “00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00”
packetGroup config -headerFilterMask “00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00”
packetGroup config -enable128kBinMode false
packetGroup config -enableTimeBins false
packetGroup config -numPgidPerTimeBin 32
packetGroup config -numTimeBins 10
packetGroup config -timeBinDuration 1000000
packetGroup config -enableLatencyBins false
packetGroup config -latencyBinList ““
packetGroup config -groupIdMode packetGroupCustom
packetGroup config -sequenceCheckingMode seqThreshold
packetGroup config -multiSwitchedPathMode seqSwitched-
PathPGID
packetGroup setRx $chassis $card $port
ipAddressTable setDefault
ipAddressTable config -defaultGateway
“0.0.0.0”
ipAddressTable set $chassis $card $port
arpServer setDefault
arpServer config -retries
3
arpServer config -mode
arpGatewayOnly
arpServer config -rate
2083333
```

```
arpServer config -requestRepeat-
Count 3
arpServer set $chassis $card $port
interfaceTable select $chassis $card $port
interfaceTable setDefault
interfaceTable config -dhcpV4RequestRate
0
interfaceTable config -dhcpV6RequestRate
0
interfaceTable config -
dhcpV4MaximumOutstandingRequests 100
interfaceTable config -
dhcpV6MaximumOutstandingRequests 100
interfaceTable set
interfaceTable clearAllInterfaces
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress
{2.2.2.2}
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress
{2.2.2.1}
interfaceEntry addItem addressTypeIpV4
dhcpV4Properties removeAllTlvs
dhcpV4Properties setDefault
dhcpV4Properties config -clientId ""
dhcpV4Properties config -serverId "0.0.0.0"
dhcpV4Properties config -vendorId ""
dhcpV4Properties config -renewTimer 0
dhcpV4Properties config -relayAgentAddress
"0.0.0.0"
dhcpV4Properties config -relayDestinationAddress
```

```
"255.255.255.255"

dhcpV6Properties removeAllTlvs

dhcpV6Properties setDefault

dhcpV6Properties config -iaType

dhcpV6IaTypePermanent

dhcpV6Properties config -iaId 0

dhcpV6Properties config -renewTimer 0

dhcpV6Properties config -relayLinkAddress

"0:0:0:0:0:0:0:0"

dhcpV6Properties config -relayDestinationAddress

"FF05:0:0:0:0:0:1:3"

interfaceEntry config -enable true

interfaceEntry config -description {ProtocolInterface

- 100:01 - 1}

interfaceEntry config -macAddress {00 00 16 1F

E3 14}

interfaceEntry config -eui64Id {02 00 16 FF

FE 1F E3 14}

interfaceEntry config -atmEncapsulation atmEn-
capsulationLLCBridgedEthernetFCS

interfaceEntry config -atmMode -1

interfaceEntry config -atmVpi 0

interfaceEntry config -atmVci 32

interfaceEntry config -enableDhcp false

interfaceEntry config -enableDhcpV6 false

interfaceEntry config -enableVlan false

interfaceEntry config -vlanId 1

interfaceEntry config -vlanPriority 5

interfaceTable addInterface 0

bfdServer select $chassis $card $port

bfdServer clearAllRouters

bfdSession setDefault

bfdSession config -enable true

bfdSession config -remoteDiscLearned 1

bfdSession config -iPType 17
bfdSession config -localBFDAddress
“0.0.0.0”
bfdSession config -remoteBFDAddress
“2.2.2.2”
bfdSession config -myDisc 1
bfdSession config -remoteDisc 1
bfdSession config -enableAutoChooseSrc true
bfdSession config -sessionType 1
bfdInterface addSession Session1
bfdInterface setDefault
bfdInterface config -enable true
bfdInterface config -interfaceId 1
bfdInterface config -desiredMinRxIntv 1000
bfdInterface config -desiredTxIntv 1000
bfdInterface config -desiredEchoIntv 0
bfdInterface config -enableDemandMode
false
bfdInterface config -enableCtrlPlaneIndependent
false
bfdInterface config -pollIntv 0
bfdInterface config -multiplier 3
bfdInterface config -flapTxIntvs 0
bfdInterface config -protocolInterfaceDescription
“ProtocolInterface - 100:01 - 1”
bfdRouter addInterface Interface1
bfdRouter setDefault
bfdRouter config -routerId “8.39.0.1”
bfdRouter config -enabled true
bfdServer addRouter Router1
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse true
protocolServer config -enableIgmpQueryResponse
false
```

```
protocolServer config -enableOspfService
false
protocolServer config -enableBgp4Service
false
protocolServer config -enableIsisService
false
protocolServer config -enableRsvpService
false
protocolServer config -enableRipService
false
protocolServer config -enableLdpService
false
protocolServer config -enableRipngService
false
protocolServer config -enableMldService
false
protocolServer config -enableOspfV3Service
false
protocolServer config -enablePimsmService
false
protocolServer config -enableStpService
false
protocolServer config -enableEigrpService
false
protocolServer config -enableBfdService true
protocolServer config -enableBgp4CreateInterface
false
protocolServer config -enableIsisCreateInterface
false
protocolServer config -enableOspfCreateInterface
false
protocolServer config -enableRipCreateInterface
false
protocolServer config -enableRsvpCreateInterface
```

```
false
protocolServer config -enableIgmpCreateInterface
false
protocolServer set $chassis $card $port
flexibleTimestamp setDefault
flexibleTimestamp config -type timestampBeforeCrc
flexibleTimestamp config -offset 23
flexibleTimestamp set $chassis $card $port
capture setDefault
capture config -fullAction lock
capture config -sliceSize 8191
capture config -sliceOffset 0
capture config -trigger -1
capture config -filter -1
capture config -captureMode captureTriggerMode
capture config -continuousFilter 0
capture config -beforeTriggerFilter captureBeforeTriggerNone
capture config -afterTriggerFilter captureAfterTriggerFilter
capture config -triggerPosition 1.0
capture config -enableSmallPacketCapture false
capture set $chassis $card $port
filter setDefault
filter config -captureTriggerDA anyAddr
filter config -captureTriggerSA anyAddr
filter config -captureTriggerPattern anyPattern
filter config -captureTriggerError errAny-
Frame
filter config -captureTriggerFrameSizeEnable false
filter config -captureTriggerFrameSizeFrom 12
filter config -captureTriggerFrameSizeTo 12
filter config -captureFilterDA anyAddr
filter config -captureFilterSA anyAddr
filter config -captureFilterPattern anyPattern
filter config -captureFilterError errAny-
```

```
Frame
filter config -captureFilterFrameSizeEnable false
filter config -captureFilterFrameSizeFrom 12
filter config -captureFilterFrameSizeTo 12
filter config -userDefinedStat1DA anyAddr
filter config -userDefinedStat1SA anyAddr
filter config -userDefinedStat1Pattern anyPattern
filter config -userDefinedStat1Error errAny-
Frame
filter config -userDefinedStat1FrameSizeEnable false
filter config -userDefinedStat1FrameSizeFrom 12
filter config -userDefinedStat1FrameSizeTo 12
filter config -userDefinedStat2DA anyAddr
filter config -userDefinedStat2SA anyAddr
filter config -userDefinedStat2Pattern anyPattern
filter config -userDefinedStat2Error errAny-
Frame
filter config -userDefinedStat2FrameSizeEnable 0
filter config -userDefinedStat2FrameSizeFrom 12
filter config -userDefinedStat2FrameSizeTo 12
filter config -asyncTrigger1DA anyAddr
filter config -asyncTrigger1SA anyAddr
filter config -asyncTrigger1Pattern anyPattern
filter config -asyncTrigger1Error errAny-
Frame
filter config -asyncTrigger1FrameSizeEnable false
filter config -asyncTrigger1FrameSizeFrom 12
filter config -asyncTrigger1FrameSizeTo 12
filter config -asyncTrigger2DA anyAddr
filter config -asyncTrigger2SA anyAddr
filter config -asyncTrigger2Pattern anyPattern
filter config -asyncTrigger2Error errAny-
Frame
filter config -asyncTrigger2FrameSizeEnable false
filter config -asyncTrigger2FrameSizeFrom 12
filter config -asyncTrigger2FrameSizeTo 12
filter config -captureTriggerEnable true
filter config -captureFilterEnable true
filter config -userDefinedStat1Enable false
filter config -userDefinedStat2Enable false
filter config -asyncTrigger1Enable false
filter config -asyncTrigger2Enable false
filter set $chassis $card $port
filterPallette setDefault
filterPallette config -DA1 “00 00 00
00 00 00”
filterPallette config -DAMask1 “00 00 00
00 00 00”
filterPallette config -DA2 “00 00 00
00 00 00”
filterPallette config -DAMask2 “00 00 00
00 00 00”
filterPallette config -SA1 “00 00 00
00 00 00”
filterPallette config -SAMask1 “00 00 00
00 00 00”
filterPallette config -SA2 “00 00 00
00 00 00”
filterPallette config -SAMask2 “00 00 00
00 00 00”
filterPallette config -pattern1 “DE ED EF
FE AC CA”
filterPallette config -patternMask1 “00 00 00
00 00 00”
filterPallette config -pattern2 00
filterPallette config -patternMask2 00
filterPallette config -patternOffset1 12
filterPallette config -patternOffset2 12
filterPallette config -matchType1 matchUser
filterPallette config -matchType2 matchUser
filterPallette config -patternOffsetType1 filter-
PalletteOffsetStartOfFrame
filterPallette config -patternOffsetType2 filter-
PalletteOffsetStartOfFrame
filterPallette config -gfpErrorCondition gfpErrorsOr
filterPallette config -enableGfptHecError true
filterPallette config -enableGfpeHecError true
filterPallette config -enableGfpPayloadCrcError true
filterPallette config -enableGfpBadFcsError true
filterPallette set $chassis $card $port
lappend portList [list $chassis $card $port]
######### Chassis-abhijit-400t Card-1 Port-2 #########
set port 2
port setFactoryDefaults $chassis $card $port
port config -speed 1000
port config -duplex full
port config -flowControl false
port config -directedAddress “01 80 C2 00
00 01”
port config -multicastPauseAddress “01 80 C2 00
00 01”
port config -loopback false
port config -transmitMode portTxPacketStreams
port config -receiveMode [expr $::portRxFirstTimeStamp|$::
portRxModeWidePacketGroup]
port config -autonegotiate true
port config -advertise100FullDuplex true
port config -advertise100HalfDuplex true
port config -advertise10FullDuplex true
port config -advertise10HalfDuplex true
port config -advertise1000FullDuplex true
port config -portMode portEthernet-
```

```
Mode
port config -rxTxMode gigNormal
port config -ignoreLink false
port config -advertiseAbilities portAdvertiseNone
port config -timeoutEnable true
port config -negotiateMasterSlave 1
port config -masterSlave portSlave
port config -enableSimulateCableDisconnect false
port config -enableAutoDetectInstrumentation false
port config -enableRepeatableLastRandomPattern false
port config -transmitClockDeviation 0
port config -preEmphasis preEmphasis0
port config -MacAddress “00 de bb 00
00 01”
port config -DestMacAddress “00 de bb 00
00 02”
port config -name ““
port config -numAddresses 1
port config -enableManualAutoNegotiate false
port config -enablePhyPolling true
port set $chassis $card $port
stat setDefault
stat config -mode statNormal
stat config -enableProtocolServerStats true
stat config -enableArpStats true
stat config -enablePosExtendedStats true
stat config -enableTemperatureSensorsStats true
stat config -enableAtmOamStats false
stat config -enableDhcpStats false
stat config -enableDhcpV6Stats false
stat config -includeRprPayloadFcsInCrc true
stat config -enableValidStats false
stat config -enableBgpStats false
stat config -enableIcmpStats true
stat config -enableOspfStats false
stat config -enableIsisStats false
stat config -enableRsvpStats false
stat config -enableLdpStats false
stat config -enableIgmpStats false
stat config -enableOspfV3Stats false
stat config -enablePimsmStats false
stat config -enableMldStats false
stat config -enableStpStats false
stat config -enableEigrpStats false
stat set $chassis $card $port
packetGroup setDefault
packetGroup config -signatureOffset 48
packetGroup config -signature “08 71 18 05”
packetGroup config -insertSignature false
packetGroup config -ignoreSignature false
packetGroup config -groupId 0
packetGroup config -groupIdOffset 52
packetGroup config -enableGroupIdMask false
packetGroup config -enableInsertPgid true
packetGroup config -groupIdMask 0
packetGroup config -latencyControl cutThrough
packetGroup config -preambleSize 8
packetGroup config -sequenceNumberOffset 44
packetGroup config -sequenceErrorThreshold 2
packetGroup config -insertSequenceSignature false
packetGroup config -allocateUdf true
packetGroup config -enableSignatureMask false
packetGroup config -signatureMask “00 00 00 00”
packetGroup config -enableRxFilter false
packetGroup config -headerFilter “00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00”
packetGroup config -headerFilterMask “00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00”
```

```
packetGroup config -enable128kBinMode false
packetGroup config -enableTimeBins false
packetGroup config -numPgidPerTimeBin 32
packetGroup config -numTimeBins 10
packetGroup config -timeBinDuration 1000000
packetGroup config -enableLatencyBins false
packetGroup config -latencyBinList "1.0 1.42 2.0 2.82
4.0 5.66 8.0"
packetGroup config -groupIdMode packetGroupCustom
packetGroup config -sequenceCheckingMode seqThreshold
packetGroup config -multiSwitchedPathMode seqSwitchedPath-
PGID
packetGroup setRx $chassis $card $port
ipAddressTable setDefault
ipAddressTable config -defaultGateway
"0.0.0.0"
ipAddressTable set $chassis $card $port
arpServer setDefault
arpServer config -retries
3
arpServer config -mode
arpGatewayOnly
arpServer config -rate
2083333
arpServer config -requestRepeat-
Count 3
arpServer set $chassis $card $port
interfaceTable select $chassis $card $port
interfaceTable setDefault
interfaceTable config -dhcpV4RequestRate
0
interfaceTable config -dhcpV6RequestRate
0
interfaceTable config -
```

```
dhcpV4MaximumOutstandingRequests 100

interfaceTable config -

dhcpV6MaximumOutstandingRequests 100

interfaceTable set

interfaceTable clearAllInterfaces

interfaceEntry clearAllItems addressTypeIpV6

interfaceEntry clearAllItems addressTypeIpV4

interfaceEntry setDefault

interfaceIpV4 setDefault

interfaceIpV4 config -gatewayIpAddress

{2.2.2.1}

interfaceIpV4 config -maskWidth 24

interfaceIpV4 config -ipAddress

{2.2.2.2}

interfaceEntry addItem addressTypeIpV4

dhcpV4Properties removeAllTlvs

dhcpV4Properties setDefault

dhcpV4Properties config -clientId ""

dhcpV4Properties config -serverId

"0.0.0.0"

dhcpV4Properties config -vendorId ""

dhcpV4Properties config -renewTimer 0

dhcpV4Properties config -relayAgentAddress

"0.0.0.0"

dhcpV4Properties config -relayDestinationAddress

"255.255.255.255"

dhcpV6Properties removeAllTlvs

dhcpV6Properties setDefault

dhcpV6Properties config -iaType

dhcpV6IaTypePermanent

dhcpV6Properties config -iaId 0

dhcpV6Properties config -renewTimer 0

dhcpV6Properties config -relayLinkAddress

"0:0:0:0:0:0:0:0"
```

```
dhcpV6Properties config -relayDestinationAddress
“FF05:0:0:0:0:0:1:3”
interfaceEntry config -enable true
interfaceEntry config -description
{ProtocolInterface - 100:02 - 2}
interfaceEntry config -macAddress {00
00 16 1F E3 15}
interfaceEntry config -eui64Id {02
00 16 FF FE 1F E3 15}
interfaceEntry config -atmEncapsulation
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmMode -1
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableDhcpV6
false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 1
interfaceEntry config -vlanPriority 5
interfaceTable addInterface 0
bfdServer select $chassis $card $port
bfdServer clearAllRouters
bfdSession setDefault
bfdSession config -enable true
bfdSession config -remoteDiscLearned 1
bfdSession config -iPType 17
bfdSession config -localBFDAddress
“0.0.0.0”
bfdSession config -remoteBFDAddress
“2.2.2.1”
bfdSession config -myDisc 1
bfdSession config -remoteDisc 1
bfdSession config -enableAutoChooseSrc
```

```
true
bfdSession config -sessionType 1
bfdInterface addSession Session1
bfdInterface setDefault
bfdInterface config -enable true
bfdInterface config -interfaceId 1
bfdInterface config -desiredMinRxIntv
1000
bfdInterface config -desiredTxIntv 1000
bfdInterface config -desiredEchoIntv 0
bfdInterface config -enableDemandMode
false
bfdInterface config -enableCtrlPlaneIndependent
false
bfdInterface config -pollIntv 0
bfdInterface config -multiplier 3
bfdInterface config -flapTxIntvs 0
bfdInterface config -protocolInterfaceDescription
"ProtocolInterface - 100:02 - 2"
bfdRouter addInterface Interface1
bfdRouter setDefault
bfdRouter config -routerId
"8.40.0.1"
bfdRouter config -enabled true
bfdServer addRouter Router1
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse true
protocolServer config -enableIgmpQueryResponse false
protocolServer config -enableOspfService false
protocolServer config -enableBgp4Service false
protocolServer config -enableIsisService false
protocolServer config -enableRsvpService false
protocolServer config -enableRipService false
protocolServer config -enableLdpService false
protocolServer config -enableRipngService false
protocolServer config -enableMldService false
protocolServer config -enableOspfV3Service false
protocolServer config -enablePimsmService false
protocolServer config -enableStpService false
protocolServer config -enableEigrpService false
protocolServer config -enableBfdService true
protocolServer config -enableBgp4CreateInterface false
protocolServer config -enableIsisCreateInterface false
protocolServer config -enableOspfCreateInterface false
protocolServer config -enableRipCreateInterface false
protocolServer config -enableRsvpCreateInterface false
protocolServer config -enableIgmpCreateInterface false
protocolServer set $chassis $card $port
flexibleTimestamp setDefault
flexibleTimestamp config -type timestampBeforeCrc
flexibleTimestamp config -offset 23
flexibleTimestamp set $chassis $card $port
capture setDefault
capture config -fullAction lock
capture config -sliceSize 8191
capture config -sliceOffset 0
capture config -trigger -1
capture config -filter -1
capture config -captureMode capture-
TriggerMode
capture config -continuousFilter 0
capture config -beforeTriggerFilter captureBeforeTriggerNone
capture config -afterTriggerFilter capture-
AfterTriggerFilter
capture config -triggerPosition 1.0
capture config -enableSmallPacketCapture false
capture set $chassis $card $port
```

```
filter setDefault
filter config -captureTriggerDA anyAddr
filter config -captureTriggerSA anyAddr
filter config -captureTriggerPattern anyPattern
filter config -captureTriggerError errAny-
Frame
filter config -captureTriggerFrameSizeEnable false
filter config -captureTriggerFrameSizeFrom 12
filter config -captureTriggerFrameSizeTo 12
filter config -captureFilterDA anyAddr
filter config -captureFilterSA anyAddr
filter config -captureFilterPattern anyPattern
filter config -captureFilterError errAny-
Frame
filter config -captureFilterFrameSizeEnable false
filter config -captureFilterFrameSizeFrom 12
filter config -captureFilterFrameSizeTo 12
filter config -userDefinedStat1DA anyAddr
filter config -userDefinedStat1SA anyAddr
filter config -userDefinedStat1Pattern anyPattern
filter config -userDefinedStat1Error errAny-
Frame
filter config -userDefinedStat1FrameSizeEnable false
filter config -userDefinedStat1FrameSizeFrom 12
filter config -userDefinedStat1FrameSizeTo 12
filter config -userDefinedStat2DA anyAddr
filter config -userDefinedStat2SA anyAddr
filter config -userDefinedStat2Pattern anyPattern
filter config -userDefinedStat2Error errAny-
Frame
filter config -userDefinedStat2FrameSizeEnable 0
filter config -userDefinedStat2FrameSizeFrom 12
filter config -userDefinedStat2FrameSizeTo 12
filter config -asyncTrigger1DA anyAddr
filter config -asyncTrigger1SA anyAddr
filter config -asyncTrigger1Pattern anyPattern
filter config -asyncTrigger1Error errAny-
Frame
filter config -asyncTrigger1FrameSizeEnable false
filter config -asyncTrigger1FrameSizeFrom 12
filter config -asyncTrigger1FrameSizeTo 12
filter config -asyncTrigger2DA anyAddr
filter config -asyncTrigger2SA anyAddr
filter config -asyncTrigger2Pattern anyPattern
filter config -asyncTrigger2Error errAny-
Frame
filter config -asyncTrigger2FrameSizeEnable false
filter config -asyncTrigger2FrameSizeFrom 12
filter config -asyncTrigger2FrameSizeTo 12
filter config -captureTriggerEnable true
filter config -captureFilterEnable true
filter config -userDefinedStat1Enable false
filter config -userDefinedStat2Enable false
filter config -asyncTrigger1Enable false
filter config -asyncTrigger2Enable false
filter set $chassis $card $port
filterPallette setDefault
filterPallette config -DA1 “00 00 00
00 00 00”
filterPallette config -DAMask1 “00 00 00
00 00 00”
filterPallette config -DA2 “00 00 00
00 00 00”
filterPallette config -DAMask2 “00 00 00
00 00 00”
filterPallette config -SA1 “00 00 00
00 00 00”
filterPallette config -SAMask1 “00 00 00
```

```
00 00 00”
filterPallette config -SA2 “00 00 00
00 00 00”
filterPallette config -SAMask2 “00 00 00
00 00 00”
filterPallette config -pattern1 “DE ED EF
FE AC CA”
filterPallette config -patternMask1 “00 00 00
00 00 00”
filterPallette config -pattern2 00
filterPallette config -patternMask2 00
filterPallette config -patternOffset1 12
filterPallette config -patternOffset2 12
filterPallette config -matchType1 matchUser
filterPallette config -matchType2 matchUser
filterPallette config -patternOffsetType1 filter-
PalletteOffsetStartOfFrame
filterPallette config -patternOffsetType2 filter-
PalletteOffsetStartOfFrame
filterPallette config -gfpErrorCondition gfpErrorsOr
filterPallette config -enableGfptHecError true
filterPallette config -enableGfpeHecError true
filterPallette config -enableGfpPayloadCrcError true
filterPallette config -enableGfpBadFcsError true
filterPallette set $chassis $card $port
lappend portList [list $chassis $card
$port]
ixWritePortsToHardware portList
ixCheckLinkState portList
###################################################################
######### Generating streams for all the ports from above #########
###################################################################
######### Chassis-abhijit-400t Card-1 Port-1 #########
chassis get “abhijit-400t”
```

```
set chassis [chassis cget -id]

set card 1

set port 1

port reset $chassis $card $port

######### Chassis-abhijit-400t Card-1 Port-2 #########

set card 1

set port 2

port reset $chassis $card $port

ixWriteConfigToHardware portList -noProtocolServer

################# Chassis-abhijit-400t BFD LearnedInfo #########

set port1 1

set port2 2

set pList {}

lappend pList [list $chassis $card $port1]

lappend pList [list $chassis $card $port2]

######## Start BFD on both ports #########

set val [ixStartBfd pList]

if { $val == 0 } {

puts "Started BFD protocol on both ports"

} else {

puts "Failed to start the BFD protocol on both ports"

}

######## Request for LearnedInfo on first port #########

bfdServer select $chassis $card $port1

bfdServer getRouter Router1

set no_of_attempts 0

set flag_retrieve [bfdRouter requestLearnedInfo]

while { $flag_retrieve != 0 } {

set flag_retrieve [bfdRouter requestLearnedInfo]

puts "flag_retrieve = $flag_retrieve"

incr no_of_attempts

if { $no_of_attempts 40 } {

puts "Failed to retrieve learnedSession info for BFD

router : Router1 , timeout"
break
}
}
set learnedList [bfdRouter getLearnedInfoList]
set val 0
if { $learnedList } {
set val [bfdRouter getFirstLearnedInfo]
}
set count 1
while { $val == 0 } {
puts “---------------------------------------------------------
---------------”
puts “LearnedInfo $count :”
set myDisc [bfdSessionLearnedInfo cget -myDisc]
puts “My discriminator : $myDisc”
set peerDisc [bfdSessionLearnedInfo cget -peerDisc]
puts “Peer Discriminator : $peerDisc”
set myIPAddress [bfdSessionLearnedInfo cget -myIPAddress]
puts “My IP Address : $myIPAddress”
set peerIPAddress [bfdSessionLearnedInfo cget -peerIPAddress]
puts “Peer IP Address : $peerIPAddress”
set sessionType [bfdSessionLearnedInfo cget -sessionType]
puts “Session Type : $sessionType”
set reqMinRxIntv [bfdSessionLearnedInfo cget -reqMinRxIntv]
puts “Received Min Rx Interval : $reqMinRxIntv”
set desMinTxIntv [bfdSessionLearnedInfo cget -desMinTxIntv]
puts “Received Tx Interval : $desMinTxIntv”
set reqMinEchoIntv [bfdSessionLearnedInfo cget -reqMinEchoIntv]
puts “Received Tx Interval : $desMinTxIntv”
set peerState [bfdSessionLearnedInfo cget -peerState]
puts “Peer State : $peerState”
set peerFlags [bfdSessionLearnedInfo cget -peerFlags]
puts “Peer Flags : $peerFlags”
set peerUPtime [bfdSessionLearnedInfo cget -peerUPtime]
```

```
  puts “Peer Up Time : $peerUPtime”
  set protosUsingSession [bfdSessionLearnedInfo cget -protosUsingSession]
  puts “Protocols using session : $protosUsingSession”
  after 10000
  incr count
  set val [bfdRouter getNextLearnedInfo]
  puts $val
```

SEE ALSO

NAME - bfdInterfaceNAME - bfdRouter, *NAME - bfdServer*,*NAME - bfdSession*

NAME - bgp4AsPathItem

bgp4AsPathItem — contains list-based BGP4 Autonomous System (AS) path attributes, including AS set and AS sequence.

SYNOPSIS

bgp4AsPathItem *subcommand options*

DESCRIPTION

The *bgp4AsPathItem* is used to construct AS Path related items. These items must be added to a route item through the use of the *bgp4RouteItem addASPathItem* command.

NOTE: **bgp4AsPathItem** builds a list of AS Path segments each time you call it. If you do not want to add a list of AS Path segments, call **bgp4RouteItem** and use the subcommand **clearASPathList** before calling **bgp4ASPathItem**.

Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on BGP4 testing with Ixia equipment. Refer to bgp4AsPathItem for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

STANDARD OPTIONS

asList

A Tcl list of AS numbers. *(default = { })*

asSegmentType

The type of list. One of:

Option	Value	Usage
	0	(default)
bgpSegmentAsSet	1	AS Path Set
bgpSegmentAsSequence	2	AS Path Sequence
bgpSegmentAsConfedSequence	3	AS Path Confederation Sequence
bgpSegmentAsConfedSet	4	AS Path Confederation Set

enableAsSegment *true | false*

Enables the generation of an AS segment, as described by *asList* and *asSegmentType. (default = 0)*

DEPRECATED STANDARD OPTIONS

asPathConfedSeqList
asPathConfedSetList
asPathSeqList
asPathSetList
enableAsPath
ConfedSeq

enableAsPath
ConfedSet
enableAsPathSeq
enableAsPathSet
enableAsSegment

COMMANDS

The **bgp4AsPathItem** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4AsPathItem cge toption

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **bgp4AsPathItem** command.

bgp4AsPathItem config option value

Modifies the configuration options of the bgp4AsPathItem. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bgp4AsPathItem.

bgp4AsPathItem setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under NAME - bgp4Server and NAME - bgp4VpnTarget.

SEE ALSO

NAME - bgp4LearnedRoute, NAME - bgp4Neighbor, NAME - bgp4RouteFilter, NAME - bgp4RouteItem, NAME - bgp4Server, NAME - bgp4StatsQuery, NAME - bgp4VpnL3Site, NAME - bgp4VpnRouteRange, NAME - bgp4VpnTarget

NAME - bgp4ExtendedCommunity

bgp4ExtendedCommunity — associates BGP4 extended community attributes with a route item

SYNOPSIS

bgp4ExtendedCommunity *subcommand options*

DESCRIPTION

The *bgp4ExtendedCommunity* is used to construct an extended community attribute for a route item. The community must be added to an route item through the use of the bgp4RouteItem *addExtendedCommunity* command.

STANDARD OPTIONS

subType

The value of the low-order type byte. Some common values are:

Name	Value	Usage
	0	(default)
Route target community	2	
Route origin community	3	
Link bandwidth community	4	

type

The value of the high-order type byte. Some common values are:

Name	Value	Usage
IANA bit	0x80	This bit may be or'd with any other values. 0 indicates that this is an IANA assignable type using First Come First Serve policy. 1 indicates that this is an IANA assignable type using the IETF Consensus policy.
Transitive bit	0x40	This but may be or'd with any other values. 0 indicates that the community is transitive across ASes and 1 indicates that it is non-transitive.
Two-octet AS specific	0	(default) Value holds a two-octet global ASN followed by a four-octet local admin value.
IPv4 address specific	1	Value holds a four-octet IP address followed by a two-octet local administrator value.
Four-octet AS specific	2	Value holds a four-octet global ASN followed by a two-octet local admin value.
Generic	3	Value holds six-octets.

value

The value associated with the extended community. *(default = {00 00 00 00 00 00})*

COMMANDS

The **bgp4ExtendedCommunity** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4ExtendedCommunity cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **bgp4ExtendedCommunity** command.

bgp4ExtendedCommunity config *option value*

Modifies the configuration options of the bgp4ExtendedCommunity. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bgp4ExtendedCommunity.

bgp4ExtendedCommunity setDefault

Sets default values for all configuration options.

EXAMPLES

SEE ALSO

NAME - bgp4LearnedRoute, *NAME - bgp4Neighbor, NAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4ExternalNeighborItem

bgp4ExternalNeighborItem — configures BGP4 external neighbor's routes.

SYNOPSIS

bgp4ExternalNeighborItem *subcommand options*

DEPRECATED

This command has been deprecated and has been replaced by options in the *NAME - bgp4Server* and *NAME - bgp4Neighbor* command. Programs that use this command will continue to operate. Documentation for this command may be found in the *Ixia TCL Development Guide for Release 3.55*.

NAME - bgp4ExternalTable

bgp4ExternalTable — configures BGP4 external neighbor routers.

SYNOPSIS

bgp4ExternalTable *subcommand options*

DEPRECATED

This command has been deprecated and has been replaced by options in the *NAME - bgp4Server* and *NAME - bgp4Neighbor* command. Programs that use this command will continue to operate. Documentation for this command may be found in the *Ixia TCL Development Guide for Release 3.55*.

NAME - bpg4IncludePrefixFilter

bpg4IncludePrefixFilter — views retrieved learned routes.

SYNOPSIS

bpg4IncludePrefixFilter *subcommand options*

DESCRIPTION

The *bpg4IncludePrefixFilter* command is used to filter the learned routes associated with a BGP neighbor. The options in this command are added to the prefix filter list in the *NAME - bgp4Neighbor* command using the *addPrefixFilter* subcommand. Refer to *bgp4LearnedRoute* for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

STANDARD OPTIONS

addressType

The IP addressing type that the entry is for, one of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

enableExactPrefix true | false

If *true*, then only learned BGP prefixes (routes) that exactly match the prefix(es) and mask in this command will be stored and all other prefixes will be discarded. If *false*, only learned BGP prefixes that exactly match or are more specific than the prefix(es) and masks in this command will be stored and all other prefixes will be discarded. *(default = false)*

firstPrefix

The first IP address prefix/route to use as a basis for filtering learned BGP routes. (default = 0.0.0.0)

maskWidth

The length, in bits, of the network mask to be used with firstPrefix for creating a range of routes/ network addresses to use for filtering routes. (default = 24)

numPrefixes

The total number of BGP prefixes/routes to be used for filtering routes. (default = 1)

COMMANDS

The *bpg4IncludePrefixFilter* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bpg4IncludePrefixFilter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bpg4IncludePrefixFilter* command.

bpg4IncludePrefixFilter config*option value*

Modifies the configuration options of the *bpg4IncludePrefixFilter*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bgp4ExtendedCommunity.

bpg4IncludePrefixFilter setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *NAME - bgp4Server*.

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4Neighbor, NAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4InternalNeighborItem

bgp4InternalNeighborItem — configures internal BGP4 neighbor routers.

SYNOPSIS

bgp4InternalNeighborItem *subcommand options*

DEPRECATED

This command has been deprecated and has been replaced by options in the *NAME - bgp4Server* and *NAME - bgp4Neighbor* command. Programs that use this command will continue to operate. Documentation for this command may be found in the *Ixia TCL Development Guide for Release 3.55*.

NAME - bgp4InternalTable

bgp4InternalTable — configures internal BGP neighbors.

SYNOPSIS

bgp4InternalTable *subcommand options*

DEPRECATED

This command has been deprecated and has been replaced by options in the *NAME - bgp4Server* and *NAME - bgp4Neighbor* command. Programs that use this command will continue to operate. Documentation for this command may be found in the *Ixia TCL Development Guide for Release 3.55*.

NAME - bgp4LearnedRoute

bgp4LearnedRoute — views retrieved learned BGP4 routes.

SYNOPSIS

bgp4LearnedRoute *subcommand options*

DESCRIPTION

The *bgp4LearnedRoute* command is used to look at the routes retrieved when using the *requestLearnedRoutes* and *getLearnedRoutesList* subcommands of the *NAME - bgp4Neighbor* command. Only the learned routes that were enabled with the *NAME - bgp4RouteFilter* command are retrieved.

Each of the enabled types of routes is logically considered as a separate list that must be retrieved with separate sets of calls to *getFirst* and *getNext.* Multiple address types may be simultaneously retrieved at the same time.

Refer to *bgp4LearnedRoute* for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

STANDARD OPTIONS

description

(Read-only.) A textual description including all of the other items.

ipAddress

(Read-only.) The IP prefix for the route.

label

(Read-only.) The MPLS label, for use with IPv4 and IPv6 MPLS address types.

neighbor

(Read-only.) The local IP address of the neighbor.

prefixLength

(Read-only.) The IP prefix length for the route.

routeDistinguisher

(Read-only.) The route distinguisher for the route, for use with IPv4 and IPv6 MPLS VPN address types.

site

(Read-only.)

neighborAddress

IP prefix for the route.

remotePeAddress

IP prefix for the route.

remoteVplsId

String prefix for the route.

supportedLocally

The displaying whether VPLS is supported locallly.

remoteVsiId

PE Address or Assigned Number.

routeDistinguisher

IP or AS prefix for the route.

routeTarget

IP or AS prefix for the route

nextHopAddress

IP prefix for the route.

peerAddress

The peer address in IP format.

vplsId

The VPLS ID in IP or AS format.

sourceAii

The 4 byte unsigned number of the Source AII.

targetAii

The 4 byte unsigned number of the Target AII.

groupId

The 4 byte unsigned number of the Group Id.

label

The 4 byte unsigned number of the label.

pwState

The boolean value of the PW State.

localPwSubState

The 4 byte unsigned number of the local PW Sub State.

remotePeSubState

The 4 byte unsigned number of the Remote PE Sub State.

cBit - boolean

The boolean value of the C Bit.

mtu

The 2 byte value for the maximum Transmission Unit (MTU).

COMMANDS

The *bgp4LearnedRoute* command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

bgp4LearnedRoute cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bgp4LearnedRoute* command.

bgp4LearnedRoute getFirst *familyId [afi safi]*

Retrieves the first learned route for the address type indicated by *familyId. FamilyId* may be one of:

Option	Value	Usage
bgp4FamilyIpV4Unicast	1	IP V4 unicast address family.
bgp4FamilyIpV4Multicast	2	IP V4 multicast address family.
bgp4FamilyIpV4Mpls	3	IP V4 MPLS address family.
bgp4FamilyIpV4MplsVpn	4	IP V4 MPLS VPN address family.
bgp4FamilyIpV6Unicast	5	IP V6 unicast address family.
bgp4FamilyIpV6Multicast	6	IP V6 multicast address family.
bgp4FamilyIpV6Mpls	7	IP V6 MPLS address family.
bgp4FamilyIpV6MplsVpn	8	IP V6 MPLS VPN address family.
bgp4FamilyUserDefined	9	The address family is defined by the *afi* and *safi* arguments.
bgp4FamilyIpVpls	10	IP(v4) VPLS address family.

The value of the label can be accessed through the options associated with this command. Specific errors include:

- No learned labels in the list of this type.

bgp4LearnedRoute getNext *familyId [afi safi]*

Retrieves the next learned route of the type indicated by *familyId*. See the *getFirst* sub-command for a list of legal values of *familyId.* The value of the label can be accessed through the options associated with this command. Specific errors include:

- *getFirst* has not been called for this *familyId.*
- No more learned labels in the list of this type.

bgp4LearnedRoute getFirst bgp4FamilyIpV4MulticastVpn

Retrieves the first bgp4 Family IpV4 Multicast Vpn id.

bgp4LearnedRoute getFirst bgp4FamilyIpV4MulticastVpn

Retrieves the first bgp4 Family IpV6 Multicast Vpn id.

EXAMPLES

See examples under *NAME - bgp4Server*

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4Neighbor, NAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4MplsRouteRange

bgp4MplsRouteRange — configures an MPLS route range with attributes to be associated with BGP neighbors.

SYNOPSIS

bgp4MplsRouteRange *subcommand options*

DESCRIPTION

The *bgp4MplsRouteRange* command holds a route range that is associated with a *bgp4Neighbor* command. It includes all of the options and subcommands of the *NAME - bgp4RouteItem* as well as additional MPLS labels which designate a label mapping for the route range.

The labels generated by the options described below are used to generate corresponding MPLS labels. Each iteration through the route range is matched to an iteration through the label range.

NOTE: Only the additional label related options are described for this command. Refer to *bgp4RouteItem* for the remainder of the options.

STANDARD OPTIONS

labelEnd

If the value of *labelMode* is *bgp4VpnIncrementLabel,* then this is the last label that will be generated. Subsequent labels will start again from *labelStart*. *(default = 1,046,400)*

labelMode

Indicates whether a single label will be used for all routes, or each route will get a unique label. One of:

Option	Value	Usage
bgp4VpnFixedLabel	0	All route ranges use the value indicated in *labelStart.*
bgp4VpnIncrementLabel	1	*(default)* The label generated is *labelStart*. Subsequent labels are incremented by *labelStep*, until *labelEnd* is reached - at which point *labelStart* is used again.

labelSpaceId

The label space ID to be associated with all the labels. *(default = 0)*

labelStart

The first label to be generated. If the value of *labelMode* is *bgp4VpnFixedLabel,* then all labels have this value. *(default = 16)*

labelStep

If the value of *labelMode* is *bgp4VpnIncrementLabel,* then this is the increment applied between generated labels. *(default = 1)*

enableAdvertiseNextHopAsV4

This option is exposed an option to advertise v4 next-hop address as v4 in case of IPv6 route range. It should be enabled if Route Range type is IPv6 and Enable NextHop is selected. An IPv4 next address should be set, either manually or by same as local IP as IPv4.

EXAMPLES

See examples under *NAME - bgp4Server*

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4Neighbor NAME - bgp4RouteFilter*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery* , *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4Neighbor

bgp4Neighbor — configures BGP neighbors.

SYNOPSIS

bgp4Neighbor *subcommand options*

DESCRIPTION

The *bgp4Neighbor* command holds information about a BGP4 internal or external neighbor router. The IP type (IPv4 or IPv6) of the neighbor router is dictated by the *ipType* option.

In addition to a number of options related to the neighbor itself, this command holds four lists:

- A list of route ranges. A *NAME - bgp4RouteItem* is added to the neighbor with the *addRouteRange* subcommand.
- A list of MPLS route ranges. A *NAME - bgp4MplsRouteRange* is added to the neighbor with the *addMplsRouteRange* subcommand.
- A list of L3 VPN sites. A *NAME - bgp4VpnL3Site* is added to the neighbor with the *addL3Site* subcommand. Only **internal** neighbors may have VPN sites.
- A list of L2 VPN sites. A *NAME - bgp4VpnL2Site* is added to the neighbor with the *addL2Site* subcommand.
- A list of opaque route ranges. A *NAME - bgp4OpaqueRouteRange* information is imported with the *opaqueRouteRange* subcommand.
- A list of route import options. The *NAME - bgp4RouteImportOptions* are added to the neighbor with the routeImportOptions subcommand.

When all route ranges, and all L2 and L3 sites, are added to the *bgp4Neighbor*, then the neighbor is added to the *NAME - bgp4Server* as either an internal or external neighbor with the *NAME - bgp4Server addNeighbor* command.

Routes learned from a network are available through the use of the *requestLearnedRoutes* and *getLearnedRoutesList*. The types of routes learned are controlled through the use of the *NAME - bgp4RouteFilter* and *NAME - bpg4IncludePrefixFilter* commands. The latter command is used to establish a single prefix filter, added to a prefix filter list in this command using the *addFilter* subcommand; the *enablePrefixFilter* option must be *true* in order for the prefix list to be used.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on BGP4 testing with Ixia equipment. Refer to *bgp4Neighbor*for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

STANDARD OPTIONS

asNumMode

(External neighbor only) When *rangeCount* is greater than 1, this controls the AS number assigned to additional routers. When set to 0, all simualted routers will have the same AS

number as specified with the *externalNeighborAsNum* option. When set to 1, routers will be assigned incrementing AS numbers starting at *neighborAsNum*. *(default = 1)*

authenticationType

The cryptographic authentication type used by the neighbor, one of:

Option	Value	Usage
bgp4NULL	0	*(default)* No cryptographic authentication will be used
bgp4MD5	1	TheMessage Digest 5 (MD5) algorithm will be used for cryptographic authentication. If selected, an MD5 key must be configured. See *md5Key* option below.

bgpId

If *enableBgpId* is *true*, then this is the BGP ID to use expressed in IPv4 format. *(default = 0.0.0.0)*

bfdModeOfOperation

Indicates whether to use a single-hop or a multi-hop mode of operation for the BFD session being created with a BGP peer. One of:

Option	Value	Usage
asynchronous mode	0	In asynchronous mode, both endpoints periodically send Hello packets to each other. If a number of those packets are not received, the session is considered down.
demand mode	1	In demand mode, no Hello packets are exchanged after the session is established; it is assumed that the endpoints have another way to verify connectivity to each other, perhaps on the underlying physical layer. However, either host may still send Hello packets if needed.

dutIpAddress

The DUT router's IP address. *(default = 0.0.0.0)*

enable true | false

Enables the use of the neighbor. *(default = true)*

enableActAsRestarted true | false

If *true*, the neighbor will initially act as if it had been restarted. *(default = false)*

enableBFDRegistrationtrue | false

Indicates if a BFD session is to be created to the BGP peer IP address once the BGP session is established. This allows BGP to use BFD to maintain IPv4 connectivity with the BGP peer.

enableBgpId true | false

Enables use of the *bgpId* field. If *false,* and *iptype* is set to *addressTypeIpV4*, then *localIpAddress* will be used. If *iptype* is set to *addressTypeIpV6*, then this value must be set to *true* and *bgpId* set to an IPv4 address. *(default = true)*

enableGracefulRestart true | false

If *true*, enables the graceful restart feature for the neighbor. *(default = false)*

enableIpV4Mdt true | false

Indicates that BGP will use a new SAFI called the MDT-SAFI (*value 66*) to carry the Data-MDT group address (*IPv4*) in the MP_REACH_NLRI field of the update-packet, instead of using an external-community.

enableIpV4Mpls true | false

If *true,* support for IPv4 MPLS is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableIpV4MplsVpn true | false

If *true,* support for IPv4 MPLS VPN is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableIpV4Multicast true | false

If *true,* support for IPv4 Multicast is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableIpV4Unicast true | false

If *true,* support for IPv4 Unicast is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableIpV6Mpls true | false

If *true,* support for IPv6 MPLS is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableIpV6MplsVpn true | false

If *true,* support for IPv6 MPLS VPN is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableIpV6Multicast true | false

If *true,* support for IPv6 Multicast is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableIpV6Unicast true | false

If *true,* support for IPv6 Unicast is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

enableLinkFlap true | false

Enables or disables the flapping of link for the neighbor. If *true*, then the link to the router is logically disconnected and reconnected based on the settings of the *linkFlapDownTime* and *linkFlapUpTime*. *(default = false)*

enableNextHop true | false

This is used for IPv4 traffic and enables the use of the NEXT_HOP attributes. *(default = false)*

enableOptional Parameters true | false

If *false*, values received in the OPEN message are ignored. If *true*, the BGP4 connection is dropped if the incoming OPEN message contains any optional parameters. *(default = false)*

enablePrefixFilter true | false

Enables use of the prefix filter use. *(default = false)*

enableStaggeredStart true | false

Enables staggered start of neighbors. *(default = false)*

enableVpls true | false

Enables VPLS for L2 VPNs. *(default = false)*

enable4ByteAsNumber true | false

Enables the 4-byte autonomous system number. *(default = false)*

localAsNumber

The AS number for the simulated router. This may be set for external neighbors for all port types, and for internal neighbors on Linux-based ports. The valid range is 0 to 65,535. *(default = 0)*

holdTimer

Specifies the amount of time for which information about better routes is ignored. Configures the hold time for BGP sessions for this neighbor. Keepalives are sent out every 1/3rd of this interval. With the default value of 90, Keepalive messages will be sent every 30 seconds. *(default = 90)*

ipType

The IP addressing type of the neighbor, one of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

linkFlapDownTime

If *enableLinkFlap* is set to *true*, then this is the amount of time in seconds that the link is disconnected. *(default = 0)*

linkFlapUpTime

If *enableLinkFlap* is set to *true*, then this is the amount of time in seconds that the link is connected. *(default = 0)*

localIpAddress

The first IP address that will be used for simulated routers. *(default = 0.0.0.0)*

md5Key

Used with MD5 authentication. A user-defined string; maximum = 255 characters.

nextHop

If *enableNextHop* is *true,* this is the IPv4 address used as the next hop. *(default = 0.0.0.0)*

numUpdatesPer Iteration

When the protocol server operates on older ports that do not possess a local processor, this tuning parameter controls how many Update messages will be sent at a time. When many routers are being simulated on such a port, changing this value may help to increase or decrease performance. *(default = 1)*

rangeCount

The number of routers to be simulated. *(default = 1)*

restartTime

If *enableGracefulRestart* is *true*, then this is the amount of time, in seconds, following a restart operation that is allowed to re-establish a BGP session. *(default = 0)*

staggeredStartPeriod

Duration of the start process, measured in seconds. *(default = 0)*

staleTime

If *enableGracefulRestart* is *true*, then this is the amount of time, in seconds, after which an End-Of-RIB marker is sent in an Update message to the peer — to allow time for routing convergence via IGP and BGP route selection. Stale routing information for that address family is then deleted by the receiving peer. *(default = 0)*

tcpWindowSize

(External neighbor only) The TCP window used for communications from the neighbor. *(default = 8,192)*

type

Indicates whether the neighbor is an internal or external neighbor. The options are:

Option	Value	Usage
bgp4NeighborInternal	0	*(default)* Internal neighbor.
bgp4NeighborExternal	1	External neighbor.

updateInterval

The time intervals at which UPDATE messages are sent to the DUT, expressed in the number of milliseconds between UPDATE messages. *(default = 0)*

enableIpV4MulticastMplsVpn

If *true*, support for IPv4 MPLS VPN is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default =true)*

enableIpV6MulticastMplsVpn

If *true*, support for IPv6 MPLS is advertised in the Capabilities Optional Parameter/Multiprotocol Extensions parameter of the OPEN message. *(default = true)*

DEPRECATED OPTIONS

externalNeighbor ASNum

(External neighbor only) The AS number for the simulated router.

COMMANDS

The *bgp4Neighbor* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4Neighbor addL2Site *l2SiteLocalId*

Adds the L2 site created with the *NAME - bgp4VpnL2Site* command to this neighbor. The L2 site is tagged with *l2SiteLocalId*.

bgp4Neighbor addL3Site *l3SiteLocalId*

Adds the L3 site created with the *NAME - bgp4VpnL3Site* command to this neighbor. The L3 site is tagged with *l3SiteLocalId*. Specific errors are:

- Invalid route range parameters.
- The route range already exists. (Delete an old entry before adding it again.)

bgp4Neighbor addMplsRouteRange *mplsRouteRangeLocalId*

Adds the MPLS route range created with the *NAME - bgp4MplsRouteRange* command to this neighbor. The MPLS route range is tagged with *mplsRouteRangeLocalId*. Specific errors are:

- Invalid route range parameters.
- The route range already exists. (Delete an old entry before adding it again.)

bgp4Neighbor addPrefixFilter

Adds the prefix filter created with the *NAME - bpg4IncludePrefixFilter* command to this neighbor. Prefix filters are only used if the *enablePrefixFilter* is *true.* Specific errors are:

- Invalid prefix filter parameters.

bgp4Neighbor addRouteRange *routeRangeLocalId*

Adds the route range created with the *NAME - bgp4RouteItem* command to this router. The route range is tagged with the *routeRangeLocalId*. Specific errors are:

- Invalid route range parameters.
- The route range already exists. (Delete an old entry before adding it again.)

bg4Neighbor addRouteImportOptions *[routeImportOptionLocalID]*

Adds the route import option created with the bg4RouteImportOption command to this router.

bgp4Neighbor cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bgp4Neighbor* command.

bgp4Neighbor clearAllL2Sites

Clears all of the L2 sites associated with the command.

bgp4Neighbor clearAllL3Sites

Clears all of the L3 sites associated with the command.

bgp4Neighbor clearAllMplsRouteRanges

Clears all of the MPLS route ranges associated with the command.

bgp4Neighbor clearAllPrefixFilters

Clears all of the prefix filters associated with the command.

bgp4Neighbor clearAllRouteRanges

Clears all of the route ranges associated with the command.

bgp4Neighbor clearAllOpaqueRouteRange

Clears all of the opaque route ranges associated with the command.

bgp4Neighbor clearAllRouteImportOptions

Clears all of the route import options associated with the command.

bgp4Neighbor config *option value*

Modifies the configuration options of the *bgp4Neighbor*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4Neighbor*.

bgp4Neighbor delL2Site *[l2SiteLocalId]*

Deletes the L2 site with the tag *l2SiteLocalId,* if specified. Otherwise the currently accessed L2 site is deleted.

bgp4Neighbor delL3Site *[l3SiteLocalId]*

Deletes the L3 site with the tag *l3SiteLocalId,* if specified. Otherwise the currently accessed L3 site is deleted.

bgp4Neighbor delMplsRouteRange *[mplsRouteRangeLocalId]*

Deletes the MPLS route range with the tag *mplsRouteRangeLocalId,* if specified. Otherwise the currently accessed MPLS route range is deleted.

bgp4Neighbor delPrefixFilter

Deletes the currently accessed prefix filter in the list.

bgp4Neighbor delRouteRange *[routeRangeLocalId]*

Deletes the route range with the tag *routeRangeLocalId,* if specified. Otherwise the currently accessed route range is deleted.

bg4Neighbor delOpaqueRouteRange *[opaqueRouteRangeLocalID]*

Deletes the opaque route range with the tag *opaqueRouteRangeLocalID*, if specified. Otherwise, the currently accessed opaque route range is deleted.

bg4Neighbor delRouteImportOption *[routeImportOptionLocalID]*

Deletes the route import option with the tag *routeImportOptionLocalID*, if spec ified. Otherwise, the currently accessed route import option is deleted.

bgp4Neighbor generateStreams *chasID cardID portID action*

Generates streams or creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replaces the port's current streams.
protocolServerStreamAppend	1	Adds the streams to the port's current streams.

A separate stream is generated for each enabled route range associated with this neighbor router; each stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.

- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address control are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

bgp4Neighbor getFirstL2Site

Gets the first L2 site from the list as the `current' item. The contents of the item are available in the *NAME - bgp4VpnL2Site* command.

bgp4Neighbor getFirstL3Site

Gets the first L3 site from the list as the `current' item. The contents of the item are available in the *NAME - bgp4VpnL3Site* command. Specific errors are:

- The neighbor does not have any L3 sites.

bgp4Neighbor getFirstMplsRouteRange

Gets the first MPLS route range from the list as the `current' item. The contents of the item are available in the *NAME - bgp4MplsRouteRange* command. Specific errors are:

- The neighbor does not have any MPLS route ranges.

bgp4Neighbor getFirstPrefixFilter

Gets the first prefix filter from the list as the `current' item. The contents of the item are available in the *NAME - bpg4IncludePrefixFilter* command. Specific errors are:

- The neighbor does not have any prefix filters.

bgp4Neighbor getFirstRouteRange

Gets the first route range from the list as the `current' item. The contents of the item are available in the *NAME - bgp4RouteItem* command. Specific errors are:

- The neighbor does not have any route ranges.

bgp4Neighbor getFirstOpaqueRouteRange

Gets the first opaque route range from the list as the current item.

bgp4Neighbor getFirstRouteImportOption

Gets the first routeImportOption from the list as the current item.

bgp4Neighbor getL2Site *l2SiteLocalId*

Gets the L2 site with the tag *l2SiteLocalId* from the list. The contents of the item are available in the *NAME - bgp4VpnL2Site* command.

bgp4Neighbor getL3Site *l3SiteLocalId*

Gets the L3 site with the tag *l3SiteLocalId* from the list. The contents of the item are available in the *NAME - bgp4VpnL3Site* command. Specific errors are:

- An item with tag *l3SiteLocalId* does not exist in the list.

bgp4Neighbor getLearnedRouteList

This subcommand should be called after *requestLearnedRoutes*. It must be called until it returns TCL_OK, usually with a wait between calls. Specific errors are:

- The list has not been completely retrieved yet.

bgp4Neighbor getMplsRouteRange *mplsRouteRangeLocalId*

Gets the MPLS route range with the tag *mplsRouteRangeLocalId* from the list. The contents of the item are available in the *NAME - bgp4MplsRouteRange* command. Specific errors are:

- An item with tag *mplsRouteRangeLocalId* does not exist in the list.

bgp4Neighbor getRouteImportOptions

Gets the route import options with the tag *routeImportOptionsLocalID* from the list.

bgp4Neighbor getNextL2Site

Gets the next L2 site in the list as the `current' item. The contents of the item are available in the *NAME - bgp4VpnL2Site* command.

bgp4Neighbor getNextL3Site

Gets the next L3 site in the list as the `current' item. The contents of the item are available in the *bgp4VpnL3Site* command. Specific errors are:

- The neighbor does not have any more L3 sites.

bgp4Neighbor getNextMplsRouteRange

Gets the next MPLS route range in the list as the `current' item. The contents of the item are available in the *bgp4MplsRouteRange* command. Specific errors are:

- The neighbor does not have any more MPLS route ranges.

bgp4Neighbor getNextPrefixFilter

Gets the next prefix filter in the list as the `current' item. The contents of the item are available in the *bpg4IncludePrefixFilter* command. Specific errors are:

- The neighbor does not have any more prefix filters.

bgp4Neighbor getNextRouteRange

Gets the next route item in the list as the `current' item. The contents of the item are available in the *NAME - bgp4RouteItem* command. Specific errors are:

- The neighbor does not have any more route ranges.

bgp4Neighbor getNextOpaqueRouteRange

Gets the next opaque route range in the list as the current item.

bgp4Neighbor getNextRouteImportOptions

Gets the next route import option in the list as the current item.

bgp4Neighbor getRouteRange *routeRangeLocalId*

Gets the route range with the tag *routeRangeLocalId* from the list. The contents of the item are available in the *NAME - bgp4RouteItem* command. Specific errors are:

- An item with tag *routeRangeLocalId* does not exist in the list.

bgp4Neighbor requestLearnedRoutes

Requests that the learned routes for this neighbor be retrieved from the protocol server. The *NAME - bgp4RouteFilter* must have been previously called to enable specific address family types to be retrieved. The *getLearnedRouteList* subcommand must be called after this subcommand to determine when the complete list of routes has been retrieved.

bgp4Neighbor setDefault

Sets default values for all configuration options.

bgp4Neighbor setL2Site *[l2SiteLocalId]*

Overwrites the L2 site with the tag *l2SiteLocalId,* if specified. Otherwise the current L2 site is overwritten. The data used is from the contents of the *l2SiteLocalId* in the *NAME - bgp4VpnL2Site* command.

bgp4Neighbor setL3Site *[l3SiteLocalId]*

Overwrites the L3 site with the tag *l3SiteLocalId,* if specified. Otherwise the current L3 site is overwritten. The data used is from the contents of the *l3SiteLocalId* in the *NAME - bgp4VpnL3Site* command. Specific errors are:

- The neighbor does not have any L3 sites.
- An item with tag *l3SiteLocalId* does not exist in the list.

bgp4Neighbor setMplsRouteRange *[mplsRouteRangeLocalId]*

Overwrites the MPLS route range with the tag *mplsRouteRangeLocalId,* if specified. Otherwise the current MPLS route range is overwritten. The data used is from the contents of the *mplsRouteRangeLocalId* in the *NAME - bgp4MplsRouteRange* command. Specific errors are:

- The neighbor does not have any L3 sites.
- An item with tag *mplsRouteRangeLocalId* does not exist in the list.

bgp4Neighbor bgp4VpnBgpAdVplsRange [enable]

Enables or disables simulation of the router.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vplsCount]

Adds the integer value that indicates the number of VPLS instance emulated using this VPLS range.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeTargetType]

Sets the RT format to AS or IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeTargetIpAddress]

An IP value, available for use only if the IPv4 Input is set to IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeTargetAsNumber]

An integer value, available for use only if Distinguish Type is set to AS.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeTargetAssignedNumber]

This is an integer value that is dependent on the routeTargetType.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeTargetStepIpAddress]

Available for use only if the IPv4 address is set to IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeTargetAsNumberStep]

This is an integer value available for use only if routeTargetType is set to AS.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeTargetAssignedNumberStep]

The target assigned number. this is an integer value that is dependent on the on the routeTargetType.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vplsIdType]

The VPLS Id. The format is AS or IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vplsIdIpAddress]

Available for use only if the route VPLS Id Type is set to IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vplsIdAsNumber]

Available for use only if VPLS Id Type is set to AS.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vplsIdAssignedNumber]

The indicated number for thevplsIdAssignedNumber attribute.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vplsIdIpAddressStep]

Available for use only if the route vplsIdType is set to IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vplsIdAsNumberStep]

Available for use only if vplsIdType is set to AS.

bgp4Neighbor bgp4VpnBgpAdVplsRange [useVplsIdAsRouteDistinguisher]

Enables the VPLS Id as the route distinguisher.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeDistinguisherType]

Sets the RD format to AS or IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeDistinguisherIpAddress]

Available for use only if the rIPv4 Input is set to IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeDistinguisherAsNumber]

Available for use only if Distinguish Type is set to AS.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeDistinguisherIpAddressStep]

Available for use only if the rIPv4 Input is set to IP.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeDistinguisherAsNumberStep]

Available for use only if Distinguish Type is set to AS.

bgp4Neighbor bgp4VpnBgpAdVplsRange [routeDistinguisherAssignedNumberStep]

The distinguisher assigned number.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vsiId]

The VSI Id. This value is Concatenate PE Address or Concatenate Assigned Number.

bgp4Neighbor bgp4VpnBgpAdVplsRange [vsiIdAssignedNumber]

The indicated number for the vsiIdAssignedNumber attribute.

bgp4Neighbor setRouteRange [routeRangeLocalId]

Overwrites the route range with the tag *routeRangeLocalId,* if specified. Otherwise the current route range is overwritten. The data used is from the contents of the route range in the *NAME - bgp4RouteItem* command. Specific errors are:

- The neighbor does not have any route ranges.
- An item with tag *routeRangeLocalId* does not exist in the list.

bgp4Neighbor setOpaqueRouteRange [opaqueRouteRangeLocalID]

Overwrites the route range with the tag *opaqueRouteRangeLocalID*, if specified. Otherwise, the currently accessed route range is overwritten.

bgp4Neighbor setRouteImportOptions [routeImportOptionsLocalID]

Overwrites the route range with the tag *routeImportOptionsLocalID*, if specified. Otherwise, the currently accessed route range is overwritten.

EXAMPLES

See examples under *NAME - bgp4Server*

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*,*NAME - bgp4Server*, *NAME - bgp4StatsQuery* , *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4RouteFilter

bgp4RouteFilter — configures filter to be used in retrieving learned routes.

SYNOPSIS

bgp4RouteFilter *subcommand options*

DESCRIPTION

The *bgp4RouteFilter* command is used to enable or disable the retrieval of particular types of learned routes in the *NAME - bgp4Neighbor* command. Refer to *bgp4RouteFilter* for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

STANDARD OPTIONS

afi

If *enableAdditional* is true, then this is used as the AFI (address format identifier) to filter on. *(default = 0)*

enableAdditional true | false

Enables the retreival of routes for a specified AFI (address format identifier) and SAFI (sub-AFI) located in the *afi* and *safi* options.*(default = false)*

enableIpV4Mdt true | false

Indicates that BGP will use a new SAFI called the MDT-SAFI (*value 66*) to carry the Data-MDT group address (*IPv4*) in the MP_REACH_NLRI field of the update-packet, instead of using an external-community.

enableIpV4Mpls true | false

Enables the retreival of routes for IPv4 MPLS.*(default = false)*

enableIpV4MplsVpn true | false

Enables the retreival of routes for IPv4 MPLS VPNs.*(default = false)*

enableIpV4Multicast true | false

Enables the retreival of routes for IPv4 multicast.*(default = false)*

enableIpV4Unicast true | false

Enables the retreival of routes for IPv4 unicast.*(default = false)*

enableIpV6Mpls true | false

Enables the retreival of routes for IPv6 MPLS.*(default = false)*

enableIpV6MplsVpn true | false

Enables the retreival of routes for IPv6 MPLS VPNs.*(default = false)*

enableIpV6Multicast true | false

Enables the retreival of routes for IPv6 multicast.*(default = false)*

enableIpV6Unicast true | false

Enables the retreival of routes for IPv6 unicast.*(default = false)*

enableVpls true | false

Enables the retreival of routes for VPLS.*(default = false)*

safi

If *enableAdditional* is true, then this is used as the SAFI (sub-address format identifier) to filter on. *(default = 0)*

enableIpV4MulticastMplsVpn

Enables the retreival of routes for IPv4 MPLS VPNs multicast. *(default = false)*

enableIpV6MulticastMplsVpn

Enables the retreival of routes for IPv6 MPLS VPNs multicast. *(default = false)*

COMMANDS

The *bgp4RouteFilter* command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

bgp4RouteFilter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **bgp4RouteFilter** command.

bgp4RouteFilter config *option value*

Modifies the configuration options of the bgp4RouteFilter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bgp4RouteFilter.

bgp4RouteFilter setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *NAME - bgp4Server*.

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4Neighbor*, *NAME - bgp4RouteItem*,*NAME - bgp4Server*, *NAME - bgp4StatsQuery* , *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*, *bgp4UmhTarget*

NAME - bgp4RouteItem

bgp4RouteItem — configures a route item, with attributes, to be associated with BGP neighbors.

SYNOPSIS

bgp4RouteItem *subcommand options*

DESCRIPTION

The *bgp4RouteItem* holds a route range that is associated with a *NAME - bgp4Neighbor* command. This command defines a set of routes and associated attributes. Two items require lists:

- An AS path item — included using the *addASPathItem* subcommand.
- An extended community item — included using the *addExtendedCommunityList* subcommand.

Ixia Reference Manual, Theory of Operations: Protocols chapter for a discussion on BGP4 testing with Ixia equipment. Refer to *bgp4RouteItem* for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

STANDARD OPTIONS

aggregatorASNum

The AS associated with the aggregator router ID in the AGGREGATOR attribute. *(default = 0).* Returns the specified dimensions of the aggregator router as an unsigned integer type.

aggregatorIDMode

Causes the AS field to be incremented for each neighbor session generated for the range of neighbor addresses in the AGGREGATOR attribute. *(default = 1)*

aggregatorIpAddress

The IP address of the router that aggregated two or more routes in the AGGREGATOR attribute. *(default = 0.0.0.0)*

asPathSetMode

Determines the handling of the Local AS# in generated AS sequences and sets and AS Confederation Sequences and Sets. The options are:

Option	Value	Usage
bgpRouteAsPath NoInclude	0	Do not include the Local AS#. When the route item is associated with an Internal router, this is only option available — and must be set in a *bgp4RouteItem config -asPathSetMode 0*.
bgpRouteAsPathInclude AsSet	2	Include the Local AS# in the AS Path Set only.
bgpRouteAsPathInclude	3	Include the Local AS# in the AS Path Confederation

Option	Value	Usage
AsSeqConf		Sequence only.
bgpRouteAsPathInclude AsSetConf	4	Include the Local AS# in the AS Path Confederation Sequence only.

clusterList

A list of clusters that a particular route has passed through; associated with the CLUSTER attribute. Expressed as a TCL list {...}. *(default = { })*

communityList

A list of communities associated with the route entry; associated with the COMMUNITY attribute. *(default = { })* The following values may also be used within the community list:

Option	Value	Usage
bgpCommunityNoExport	0xFFFFFF01	
bgpCommunityNoAdvertise	0xFFFFFF02	
bgpCommunityExportSubconfed	0xFFFFFF03	

delay

The delay, in seconds, before advertising the route range. *(default = 0)*

enableAggregator true | false

Generates an AGGREGATOR attribute using the *aggregatorIpAddress, aggregatorASNum,* and *aggregatorIDMode.(default = false)*

enableASPathtrue | false

Enables the generation of AS Path related items (AsSet, AsSequence, AsConfedSet, and AsConfedSequence) based on information added from the *bgp4AsPathItem.(default = true)*

enableAtomicAggregate true | false

Sets the attribute bit that indicates that the router has aggregated two or more prefixes in the AGGREGATOR attribute. *(default = false)*

enableClustertrue | false

Enables the generation of the CLUSTER attribute list based on information in *clusterList. (default = false)*

enableCommunity true | false

Enables the generation of a COMMUNITY attribute list based on information in *communityList*. *(default = false)*

enableGenerate UniqueRoutes true | false

When set to 1, each router generates a different IP address range. A *bgp4InternalNeighborItem* or *bgp4ExternalNeighborItem* item must be configured with *rangeCount* 1. When not enabled, each router will advertise the route range as is. When enabled, the first router advertises *numRoutes* routes starting at *networkAddress*, the next router advertises *numRoutes* routes starting at (*networkAddress* + *numRoutes*), and so on. *(default = false)*

enableIncludeLoopback true | false

If *true,* will include the loopback address (127.0.0.1) if it is in the generated network range. *(default = false)*

enableIncludeMulticast true | false

If *true,* will include multicast addresses if they are in the generated network range. The SAFI used for multicast addresses is dictated by the setting of the *enableProperSafi* option. *(default = false)*

enableLocalPref true | false

ables the generation of a LOCAL PREF attribute based on the information in *localPref*. This value should be set to true only for EBGP. *(default = false)*

enableMED true | false

Enables the generation of a MULTI EXIT DISCRIMINATOR attribute, based on the information in *MED*. *(default = false)*

enableNextHop true | false

Enables the generation of a NEXT HOP attribute, based on information in *nextHopIpAddress* and *nextHopMode(default = true)*

enableOrigin true | false

Enables the generation of an ORIGIN attribute, based on information in *originProtocol*. *(default = true)*

enableOriginatorId true | false

Enables the generation of an ORIGINATOR-ID attribute, based on information in *originatorId*. *(default = false)*

enableProperSafi true | false

If *false*, generated multicast addresses will use a unicast SAFI. If *true*, generated multicast addresses will use a proper multicast SAFI. *(default = false)*

enableRouteFlap true | false

Enables the flapping functions described by *routeFlapTime, routFlapDropTime, routesToFlapFrom,* and *routesToFlapTo*. *(default = false)*

enableRouteRange true | false

Enables the use of this route item as an advertised range. *(default = false)*

enableTraditionalNlri Update true | false

If checked, use the traditional NLRI in the UPDATE message, instead of using the MP_ REACH_NLRI Multi-protocol extension to advertise the routes. (Not applicable for MPLS and MPLS VPN Route Ranges.) *(default =)*

endOfRIB true | false

If *true*, this indicates the end of routing information for a particular Address Family, Sub-Address Family (AFI, SAFI) when BGP peers re-advertise routes during graceful restart. *(default = false)*

fromPacking

The minimum number of routes to pack into an UPDATE message. Random numbers are chosen from the range *fromPacking* to *toPacking*. *(default = 0)*

The following chart indicates the range of packing depending on the setting of this and the *toPacking* values:

fromPacking	toPacking	Packing Ranges
0	0	As many as possible
0	b	Random from 1 to b
a	0	Always a routes
a	b	Random from a to b

fromPrefix

The first prefix length to generate based on the *networkAddress* and *numRanges.(default = 24)*

ipType

The type of IP address in *nextworkAddress*. One of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

iterationStep

During prefix generation, the increment between prefixes. *(default = 1)*

localPref

The local preference value for the routes with the LOCAL PREF attribute. *(default = 0)*

med

The multi-exit discriminator value in the MULTI EXIT DISCRIMINATOR attribute. *(default = 0)*

networkAddress

The network address used for the generated prefixes, in either IPv4 or IPv6 format. *(default = 0.0.0.0)*

nextHopIpAddress

The IP address, in either IPv4 or IPv6 format of the next hop associated with the NEXT HOP attribute. *(default = 0.0.0.0)*

nextHopIpType

The type of IP address in *nextHopIpAddress*. One of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

nextHopMode

Indicates that the *nextHopIpAddress* may be incremented for each neighbor session generated for the range of neighbor addresses. Three types of increment are available:

Option	Value	Usage
bgpRouteNextHopFixed	0	Do not change next Hop values.
bgpRouteNextHopIncrement	1	*(default)* Increment entire address.
bgpRouteNextHopIncrement PerPrefix	2	The next hop will be increment for each advertised prefix.

nextHopSetMode

Indicates now to set the next hop IP address. One of:

Option	Value	Usage
bgpRouteNextHopSetManually	0	The value is read from *nextHopIpAddress*.
bgpRouteNextHopSetSame AsLocalIp	1	*(default)* The value is the same as the local IP address.

numRoutes

The number of prefixes (routes) to generate for this *routeItem*. *(default = 1)*

originatorId

The router that originated a particular route; associated with the ORIGINATOR-ID attribute. *(default = 0.0.0.0)*

originProtocol

An indication of where the route entry originated. One of:

Option	Value	Usage
bgpOriginIGP	0	*(default)* Interior Gateway Protocol
bgpOriginEGP	1	Exterior Gateway Protocol
bgpOriginIncomplete	2	learned by some other means

routeFlapDropTime

During flapping operation, the period expressed in seconds during which the route will be withdrawn from its neighbors. *(default = 0)*

routeFlapTime

During flapping operation, the time between flap cycles, expressed in seconds. *(default = 0)*

routesToFlapFrom

During flapping operation, the first route prefix in a range to flap. *(default = 0)*

routesToFlapTo

During flapping operation, the last route prefix in a range to flap. *(default = 0)*

thruPacking

The maximum number of routes to pack into an UPDATE message. Random numbers are chosen from the range *fromPacking* to *toPacking*. See the discussion under *fromPacking* above. *(default = 0)*

thruPrefix

The last prefix length to generate based on the *networkAddress* and *numRanges.(default = 24)*

COMMANDS

The *bgp4RouteItem* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4RouteItem addASPathItem

Adds the AS Path items specified via config option calls to the *NAME - bgp4AsPathItem* list. The AS path item must have been previously configured through the use of the *NAME - bgp4AsPathItem* command. Specific errors are:

- Invalid AS path parameters

bgp4RouteItem addExtendedCommunity

Adds an extended community attribute built with the *NAME - bgp4ExtendedCommunity* command.

bgp4RouteItem cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bgp4RouteItem* command.

bgp4RouteItem clearASPathList

Clears all of the AS Path items associated with the command.

bgp4RouteItem clearExtendedCommunityList

Clears all of the extended community items associated with the command.

bgp4RouteItem config *option value*

Modify the configuration options of the bgp4RouteItem. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bgp4RouteItem.

bgp4RouteItem generateStreams*chasID cardID portID action*

Generate streams or creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each this route range, the stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address control are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

bgp4RouteItem getFirstASPathItem

Gets the first AS path item from the list. The contents of the item are available in the *NAME - bgp4AsPathItem* command. This and *getNextASPathItem* command may only be used if the route item was retrieved with the *NAME - bgp4Server getFirst/GetNextNeighbor* command. Specific errors are:

- There are no items in the list.

bgp4RouteItem getExtendedCommunity

Gets the first extended community item from the list. The contents of the item are available in the *NAME - bgp4ExtendedCommunity* command. This and *getNextExtendedCommunity* command may only be used if the route item was retrieved with the *NAME - bgp4Server getFirst/GetNextNeighbor* command. Specific errors are:

- There are no items in the list.

bgp4RouteItem getNextASPathItem

Gets the next AS path item in the list. The contents of the item are available in the *NAME - bgp4AsPathItem* command. Specific errors are:

- There are no more items in the list.

bgp4RouteItem getNextExtendedCommunity

Gets the next extended community item in the list. The contents of the item are available in the *NAME - bgp4ExtendedCommunity* command. Specific errors are:

- There are no more items in the list

bgp4RouteItem setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *NAME - bgp4Server*

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery* , *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4Server

bgp4Server — accesses the BGP4 component of the protocol server for a particular port.

SYNOPSIS

bgp4Server *subcommand options*

DESCRIPTION

The *bgp4Server* command is necessary in order to access the BGP4 component of the protocol server for a particular port. The *select* subcommand **must** be used before all other BGP4 commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on BGP4 testing with Ixia equipment. Refer to *bgp4Server* for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

NOTE: The BGP4 related commands reflect a new API. Older commands, options, and subcommands are deprecated. Although they are deprecated, tests written with the older API will continue to work; refer to the *Ixia TCL Development Guide for Release 3.55* for a description of that API.

STANDARD OPTIONS

The following options apply to all internal neighbors.

enableInternalActive Connect true | false

If enabled, a HELLO message is actively sent when BGP testing starts. Otherwise, the port waits for the DUT to send its HELLO message. *(default = true)*

triggerVplsPwInitiation

Enables the BGP-LDP communication.

internalLocalAsNum

The AS number (an unsigned integer type) for the routers participating in the simulated IBGP group. *(default = 1)*

internalRetries

The number of times to attempt an OPEN connection with the DUT routers before giving up. *(default = 0)*

internalRetryDelay

When retries are necessary, the delay in seconds between retries. *(default = 120)*

The following options apply to all external neighbors:

enableLabelExchangeOverLsp true / false

Enables protocol sessions to run over established LSPs.

If true, when a protocol packet is transmitted by an Ixia port and the IP details match an established LSP, the packet is MPLS encapsulated.

The MPLS label is set to the value learned from the LSP.

enableExternal ActiveConnect

If enabled, a HELLO message is actively sent when BGP testing starts.Otherwise, the port waits for the DUT to send its HELLO message. *(default = true)*

externalRetries

The number of times to attempt an OPEN connection with the DUT routers before giving up. *(default = 3)*

externalRetryDelay

When retries are necessary, the delay in seconds between retries. *(default = 120)*

COMMANDS

The **bgp4Server** command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

bgp4Server addNeighbor *neighborId*

Adds a neighbor to the list of BGP4 neighbors at the current position using the contents of the *NAME - bgp4Neighbor* command. Specific errors are:

- A port has not been selected via the *bgp4serverselect* command.
- The port is owned by another user.
- An object with this ID has already been added.

bgp4Server cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **bgp4Server** command.

- Invalid neighbor configuration.

bgp4Server clearAllNeighbors

Clears all of the neighbors from the neighbor list. Specific errors are:

- A port has not been selected via the *bgp4serverselect* command.
- The port is owned by another user.
- An object with this ID has already been added.

bgp4Server config *option value*

Modify the configuration options of the bgp4Server. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bgp4Server.

bgp4Server delNeighbor *[neighborId]*

Deletes a neighbor from the list of BGP4 neighbors either by matching the *neighborId* or using the neighbor at the current position. Specific errors are:

- A port has not been selected via the *bgp4serverselect* command.
- The port is owned by another user.
- An object with this ID has already been added.
- There is no neighbor with this ID.

bgp4Server generateStreams *chasID cardID portID action*

Generate streams or creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each enabled route range associated with each neighbor router; each stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address control are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

bgp4Server get

Gets the current configuration of the BGP4 server for the port set with the *select* subcommand from IxHal. Call this command before calling *bgp4Server* cget *option value* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.
- A port has not been selected via the *bgp4server select* command.

bgp4Server getFirstNeighbor

Makes the first neighbor in the list the `current' neighbor and retrieves the data so that it can be viewed and modified with the *NAME - bgp4Neighbor* command. Specific errors are:

- A port has not been selected via the *bgp4server select* command.
- The list is empty.

bgp4Server getNeighbor *neighborId*

Finds the neighbor indicated by the *neighborId*, sets it to the `current' neighbor and retrieves the data so that it can be viewed and modified with the *NAME - bgp4Neighbor* command. Specific errors are:

- A port has not been selected via the *bgp4server select* command.
- There is no neighbor with this ID.

bgp4Server getNextNeighbor

Makes the next neighbor in the list the `current' neighbor and retrieves the data so that it can be viewed and modified with the *NAME - bgp4Neighbor* command. Specific errors are:

- A port has not been selected via the *bgp4server select* command.
- There are no more objects in the list.

bgp4Server select *chasID cardID portID*

Accesses the BGP4 component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- Invalid port specified.

bgp4Server set

Sets the configuration of the BGP4 server in IxHAL for the port selected with the *select* subcommand by reading the configuration option values set by the *bgp4Serverconfigoption value* command. Specific errors are:

- No connection to a chassis.
- Invalid port number.

bgp4Server setDefault

Sets default values for all configuration options.

bgp4Server setNeighbor *[neighborId]*

Replaces the data associated with a neighbor, either the neighbor with the indicated *neighborId* or the currently selected neighbor, if the *neighborId* argument is omitted. The data for the neighbor is retrieved from the *NAME - bgp4Neighbor* command. Specific errors are:

- A port has not been selected via the *bgp4server select* command.
- The port is owned by another user.
- Invalid neighbor configuration.
- There is no neighbor with this ID.

bgp4Server write

Writes or commits the changes in IxHAL to hardware for the BGP4 related parameters on the port selected with the *select* subcommand*.* Before using this command, use the *bgp4Server set*. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- Network problem between the client and chassis.

EXAMPLES

```
package req IxTclHal

set hostname loopback

set username user

set card 4

set port 1

set streamId 1

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts “Could not connect to $host”

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {

ixPuts $::ixErrorInfo

return 1

}

# Get the chassis ID to use in port lists

set chassis [ixGetChassisID $host]

set portList [list [list $chassis $card $port]]
```

```
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
# Set up interface table for port
interfaceTable select $chassis $card $port
interfaceTable clearAllInterfaces
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress {192.18.1.1}
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress {192.18.1.2}
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description {1 - 04:01}
interfaceEntry config -macAddress {00 00 09 79 B4 78}
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 0
interfaceTable addInterface
# Select port for remaining bgp4 commands
bgp4Server select $chassis $card $port
bgp4Server clearAllNeighbors
# Set up an AS path sequence to use in a route range
bgp4AsPathItem setDefault
bgp4AsPathItem config -enableAsSegment true
bgp4AsPathItem config -asList {100 200 300 }
bgp4AsPathItem config -asSegmentType bgpSegmentAsSequence
# Add the AS path item to the route range
```

```
bgp4RouteItem addASPathItem
# Configure the route range
bgp4RouteItem setDefault
bgp4RouteItem config -enableRouteRange true
bgp4RouteItem config -networkAddress {10.0.0.0}
bgp4RouteItem config -ipType addressTypeIpV4
bgp4RouteItem config -fromPrefix 16
bgp4RouteItem config -thruPrefix 16
bgp4RouteItem config -numRoutes 1000
bgp4RouteItem config -enableRouteFlap true
bgp4RouteItem config -routeFlapTime 1999
bgp4RouteItem config -routeFlapDropTime 1
bgp4RouteItem config -enableNextHop true
bgp4RouteItem config -nextHopIpAddress {0.0.0.0}
bgp4RouteItem config -nextHopIpType addressTypeIpV4
bgp4RouteItem config -nextHopMode 1
bgp4RouteItem config -nextHopSetMode \
bgpRouteNextHopSetSameAsLocalIp
bgp4RouteItem config -enableOrigin true
bgp4RouteItem config -originProtocol bgpOriginIGP
bgp4RouteItem config -enableLocalPref true
bgp4RouteItem config -localPref 10
bgp4RouteItem config -enableASPath true
bgp4RouteItem config -iterationStep 2
bgp4RouteItem config -asPathSetMode \
bgpRouteAsPathIncludeAsSeq
# Add the route range to the Neighbor
bgp4Neighbor addRouteRange routeRange1
# Configure a similar MPLS Route Range, with labels
bgp4MplsRouteRange setDefault
bgp4MplsRouteRange config -enableRouteRange true
bgp4MplsRouteRange config -networkAddress {10.0.0.0}
bgp4MplsRouteRange config -ipType addressTypeIpV4
bgp4MplsRouteRange config -fromPrefix 16
```

```
bgp4MplsRouteRange config -thruPrefix 16
bgp4MplsRouteRange config -numRoutes 1000
bgp4MplsRouteRange config -enableRouteFlap true
bgp4MplsRouteRange config -routeFlapTime 1999
bgp4MplsRouteRange config -routeFlapDropTime 1
bgp4MplsRouteRange config -enableNextHop true
bgp4MplsRouteRange config -nextHopIpAddress {0.0.0.0}
bgp4MplsRouteRange config -nextHopIpType addressTypeIpV4
bgp4MplsRouteRange config -nextHopMode 1
bgp4MplsRouteRange config -nextHopSetMode
bgpRouteNextHopSetSameAsLocalIp
bgp4MplsRouteRange config -enableOrigin true
bgp4MplsRouteRange config -originProtocol bgpOriginIGP
bgp4MplsRouteRange config -enableLocalPref true
bgp4MplsRouteRange config -localPref 10
bgp4MplsRouteRange config -enableASPath true
bgp4MplsRouteRange config -iterationStep 2
bgp4RouteItem clearASPathList
bgp4MplsRouteRange config -labelMode
bgp4VpnIncrementLabel
bgp4MplsRouteRange config -labelStart 1000
bgp4MplsRouteRange config -labelEnd 1500
# Add the MPLS route range to the neighbor
bgp4Neighbor addMplsRouteRange mplsRouteRange1
# Set up the external neighbor
bgp4Neighbor setDefault
bgp4Neighbor config -type bgp4NeighborExternal
bgp4Neighbor config -enable true
bgp4Neighbor config -localIpAddress {192.18.1.2}
bgp4Neighbor config -rangeCount 1
bgp4Neighbor config -dutIpAddress {192.18.1.1}
bgp4Neighbor config -ipType addressTypeIpV4
bgp4Neighbor config -holdTimer 90
bgp4Neighbor config -updateInterval 0
```

```
bgp4Neighbor config -enableLinkFlap true
bgp4Neighbor config -linkFlapDownTime 89
bgp4Neighbor config -linkFlapUpTime 1
bgp4Neighbor config -localAsNumber 42
bgp4Neighbor config -asNumMode
bgp4AsNumModeIncrement
bgp4Neighbor config -enableBgpId true
bgp4Neighbor config -bgpId {192.18.1.2}
# Set up filter to later retrieve IPv4 Unicast routes
bgp4RouteFilter config -enableIpV4Unicast true
# And add the neighbor to the server
bgp4Server addNeighbor neighbor1
# Enable the BGP protocol
protocolServer setDefault
protocolServer config -enableBgp4Service true
protocolServer set $chassis $card $port
ixWritePortsToHardware portList
# Set up for bgp4StatsQuery
bgp4StatsQuery clearAllNeighbors
bgp4StatsQuery clearAllStats
bgp4StatsQuery addNeighbor 192.168.1.1 192.168.1.2
bgp4StatsQuery addStat bgpExternalConnectsAccepted
bgp4StatsQuery addStat bgpExternalConnectsReceived
# Send the data to the hardware
ixWriteConfigToHardware portList
bgp4StatsQuery get $chassis $card $port
# Need to wait until getStat returns successfully
set timer 10
for {set time 0} {$time < $timer } {incr time} {
if {![bgp4StatsQuery getStat bgpExternalConnectsAccepted \
192.168.1.1 192.168.1.2]} {
set accepted [bgp4StatsQuery cget -statValue]
bgp4StatsQuery setDefault
if [bgp4StatsQuery getStat bgpExternalConnectsReceived \
$localIpAdd $dutIpAdd] {
set received [bgp4StatsQuery cget -statValue]
}
break
}
after 1000
}
# Retrieve IPv4 Unicast routes for the first neighbor
# The filter was set up above before the addNeighbor call
if [bgp4Server getNeighbor neighbor1] {
ixPuts "Could not retrieve neighbor1"
}
# Request the routes
if [bgp4Neighbor requestLearnedRoutes] {
ixPuts "requestLearnedRoutes failed"
}
# Wait until they're ready - this could be improved by
# waiting only for a limited period of time
while {[bgp4Neighbor getLearnedRouteList != $::TCL_OK]}
after 1000
}
# Now get the first two routes learned
if [bgp4LearnedRoute getFirst bgp4FamilyIpV4Unicast] {
ixPuts "No IPV4 Unicast routes learned"
} else {
showCmd bgp4LearnedRoute
if [bgp4LearnedRoute getNext bgp4FamilyIpV4Unicast] {
ixPuts "No second IPV4 Unicast route learned"
} else {
showCmd bgp4LearnedRoute
}
}
# Let go of the ports that we reserved
ixClearOwnership $portList
```

```
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4MplsRouteRange* *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4StatsQuery* , *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4StatsQuery

bgp4StatsQuery — gets the BGP server statistics on a port of a card on a chassis.

bgpStatsQuery — also known by this name, but usage is deprecated.

SYNOPSIS

bgp4StatsQuery *subcommand options*

DESCRIPTION

The **bgp4StatsQuery** command is used to get BGP4 protocol server related statistics. Specific statistics must be requested for a set of <Neighbor IP address, DUT IP address pairs. The pairs are set up using the *addNeighbor* subcommand, while the set of statistics desired are established with the *addStat* subcommand. Statistics are read from the hardware via the *get* subcommand, and specific statistics are made available through the *getStat* subcommand. The optional BGP4 test package must be installed and running in order for this command to operate.

STANDARD OPTIONS

statName

(Read-only.) The name of the statistic retrieved.

statValue

(Read-only.) The value of the statistic, as a string.

COMMANDS

The **bgp4StatsQuery** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4StatsQuery addNeighbor *neighborIPAddress dutIPAddress [ipType]*

Adds a new neighbor-DUT address pair to the table of statistics to be fetched. The *ipType* option indicates the type of addresses for both *neighborAddress* and *dutIPAddress.* One of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address.
addressTypeIpV6	18	An IPv6 address.

Specific errors are:

- Invalid IP address(es)

bgp4StatsQuery addStat *statID*

Adds a new statistic to the table of statistics to be fetched. See the **getStat** command for a list of *statID*s. Specific errors are:

- Invalid *statID*.

bgp4StatsQuery cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **bgp4StatsQuery** command.

bgp4StatsQuery clearAllNeighbors

Deletes all neighbor-DUT address pairs from the table of statistics to be fetched.

bgp4StatsQuery clearAllStats

Deletes all statistics from the table of statistics to be fetched. See the **getStat** command for a list of *statID*s.

bgp4StatsQuery delNeighbor *neighborIPAddress dutIPAddress [ipType]*

Deletes a neighbor-DUT address pair from the table of statistics to be fetched. The *ipType* option indicates the type of addresses for both *neighborAddress* and *dutIPAddress.* One of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address.
addressTypeIpV6	18	An IPv6 address.

Specific errors are:

- The neighbor pair could not be found in the list.

bgp4StatsQuery delStat *statID*

Deletes a statistic from the table of statistics to be fetched. See the **getStat**command for a list of *statID*s.

bgp4StatsQuery get *chasID cardID portID*

This command causes the statistics associated with the indicated port to be fetched from the protocol server. It may be followed by one or more calls to *bgp4StatsQuery getStat.*

To correctly execute this command, the BGP session configuration must be completed. This can be verified by checking that the global *bgpTotalSessions* statistic is equal to the number of configured sessions.

bgp4StatsQuery getStat *statID neighborIPAddress dutIPAddress [ipType]*

Gets the statistics counter of type *statID* which may be one of the items from the table below. This must be preceded by a call to *bgp4StatsQuery get*. The first call to this command following a call to *bgp4StatsQuery get* must wait for a return value before preceding. The actual value of the statistics may be obtained via calls to *bgp4StatsQuery cget statName* and *statValue.* Additional statistics may be obtained by calling *bgp4StatsQuery getStat* again.

The *ipType* option indicates the type of addresses for both *neighborAddress* and *dutIPAddress.* One of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address.
addressTypeIpV6	18	An IPv6 address.

The available statistics are:

statID	Usage
bgpActiveOn	Indicates that *enableActiveConnect* is on. See *NAME - bgp4ExternalTable* and *NAME - bgp4InternalTable*.
bgpCeaseReceived	Ceases Received. Number of Ceases received. The Cease error code is used by a BGP peer in a Notification message to close its BGP connection. Must not be used when a fatal error exists, such as those listed here.
bgpCeaseSent	Ceases Sent. Number of Ceases sent. The Cease error code is used by a BGP peer in a Notification message to close its BGP connection. Must not be used when a fatal error exists, such as those listed here.
bgpHeaderErrorBadMsgLength	Bad Message Length. Multiple causes for this error are if the Length field for the header is less than 19 or greater than 4096, the OPEN message is less than minimum length, the UPDATE message is less than minimum length, the KEEPALIVE message is not equal to 19, or the NOTIFICATION message is less than minimum length.
bgpHeaderErrorBadMsgType	Bad Message Type. When the Type field of the message header is unrecognized.
bgpExternalConnectsAccepted	External Connects Accepted. The total number of attempted BGP connections by remote peers, in which the local system accepted the connection.
bgpExternalConnectsReceived	External Connects Received. The total number of attempted BGP connections by remote peers, including accepted and rejected.
bgpGracefulRestartsAttempted	The number of graceful restarts that were attempted.
bgpGracefulRestartsFailed	The number of graceful restarts that failed.
bgpHeaderErrorConnNotSyncron	Connection Not Synchronized. When the Marker field of the message header is not the one expected.
bgpHeaderErrorReceived	Header Errors Received. Total Number of Header Errors received on this port.
bgpHeaderErrorSent	Header Errors Sent. Total Number of Header

statID	Usage
	Errors sent from this port.
bgpHeaderErrorsSubUnspecified	Invalid Header Suberror Unspecified. If the error subcode is not defined, a zero identifies this message header error as unspecified.
bgpHoldTimeExpiredReceived	Hold Timer Expireds Received. Number of Notification error messages received. For KeepAlive, Update, and/or Notification messages. If not received by the system within the time in the Hold Time field of the OPEN message. After this message is sent, the BGP connection must be closed.
bgpHoldTimeExpiredSend	Hold Timer Expireds Sent. Number of Notification error messages sent. For KeepAlive, Update, and/or Notification messages. If not received by the system within the time in the Hold Time field of the OPEN message. After this message is sent, the BGP connection must be closed.
bgpHoldTimer	Our Hold Timer. A 2-octet unsigned integer indicating the Hold Timer value, in seconds, proposed by the sender.
bgpInvalidOpenAuthenticationFail	Authentication Failures Received. For messages which carry Authentication Information, if the authentication procedure fails.
bgpInvalidOpenBadBGPId	Bad BGP Ids Received. When the BGP Identifier field is not syntactically correct (not a valid IP host address).
bgpInvalidOpenBadPeerAS	Invalid Open with bad peer AS number. An OPEN was receives with an invalid peer AS number.
bgpInvalidOpenReceived	Invalid Opens Received. Total number of Invalid Open error messages received.
bgpInvalidOpenSent	Invalid Opens Sent. Total number of Invalid Open error messages received.
bgpInvalidOpenSubUnspecified	Invalid Open Suberror Unspecified. If the error subcode is not defined, a zero identifies this Open message error as unspecified.
bgpInvalidOpenUnacceptHoldTime	Non Acceptable Hold Times Received. The Hold Time field is not acceptable. Hold time values of 1 or 2 seconds must be rejected. Any Hold Time may be rejected by the implementation.
bgpInvalidOpenUnsupportParm	Unsupported Parameters Received. When one of the optional parameters in the OPEN message is not recognized.
bgpKeepAliveReceived	KeepAlives Received. Total number of KeepAlive messages received.
bgpKeepAliveSent	KeepAlives Sent. Total number of KeepAlive

statID	Usage
	messages sent. They cannot be sent more often than 1 per second, but must be sent often enough to keep the Hold Timer from expiring.
bgpMessageSent	Messages Sent. Total number of Notification messages sent.
bgpMessageReceived	Messages Received. Total number of all types of BGP4 messages received.
bgpNotificationReceived	Notifications received. The number of BGP notification messages received.
bgpNotificationSent	Notifications sent. The number of BGP notification messages sent.
bgpOpenReceived	Opens Received. Total number of Open messages received.
bgpOpenSent	Opens Sent. Total number of Open messages sent.
bgpOurAS	Our AS Number. A 2-octet unsigned integer indicating the Autonomous System number of the sender.
bgpOurId	Our ID. A 4-octet unsigned integer for the sender's BGP Identifier.
bgpOurIp	Our IP. The sender's IP address.
bgpPeerAS	Peer AS Number. A 2-octet unsigned integer indicating the Autonomous System number of the BGP peer.
bgpPeerHoldTime	Peer Hold Timer. A 2-octet unsigned integer indicating the Hold Timer value, in seconds, proposed by the BGP peer.
bgpPeerId	Peer Id. A 4-octet unsigned integer for the peer's BGP Identifier.
bgpPeerIP	Peer IP. The peer's IP address.
bgpRoutesAdvertised	Routes Advertised. Total number of BGP routes advertised.
bgpRoutesAdvertisedReceived	Routes Received. Total number of BGP routes received.
bgpRoutesPerSecondReceived	Routes Received per Second. Number of BGP routes received per second.
bgpRoutesReceivedBeforeStale TimerExpired	The number of routes that were received prior to the stale timer expiring.
bgpRoutesWithdrawn	Routes Withdrawn. Total number of BGP routes withdrawn.
bgpRoutesWithdrawnReceived	Route Withdraws Received. Total number of Update messages received which have a non-empty Withdrawn Routes field.
bgpStartsOccured	Starts Occurred. The number of BGP Start Events which have occurred.

statID	Usage
bgpStateMachineErrorReceived	State Machine Errors Received. Total number of State Machine Errors Received. These are errors detected by the BGP Finite State Machine.
bgpStateMachineErrorSent	State Machine Errors Sent. Total number of State Machine Errors Sent. These are errors detected by the BGP Finite State Machine.
bgpStateMachineState	State Machine State. The current state of the Finite State Machine. One of: Idle Connect Active OpenSent OpenConfirm Established
bgpUnspecifiedErrorReceived	Unspecified Error Received. The number of errors received that are not described in RFC 1771 and amendments.
bgpUnspecifiedErrorSent	Unspecified Error Sent. The number of errors sent that are not described in RFC 1771 and amendments.
bgpUpdateAttribLengthError	Attribute Length Error. For any recognized attribute where the Attribute Length conflicts with the expected length, based on attribute type code.
bgpUpdateErrorAsPathInvalid	Malformed AS_PATH. If the AS_PATH attribute is not correct syntactically.
bgpUpdateErrorASRoutingLoop	AS Routing Loop. The number of AS routing loop errors received.
bgpUpdateErrorAttribFlagError	Attribute Flags Error. For any recognized attribute where the Attribute Flags conflict with the Attribute Type.
bgpUpdateErrorAttribListError	Malformed Attribute List. When the Unfeasible Routes Length or Total Attribute Length is too large. Also, when any attribute appear more than once in the Update message.
bgpUpdateErrorMissingWellKnownAttrib	Missing Well-known Attribute. When any of the mandatory well-known attributes are not present.
bgpUpdateErrorNetworkFieldInvalid	Invalid Network Field. The syntax of the Network Layer Reachability Information (NLRI) field is not correct.
bgpUpdateErrorNextHopAttribInvalid	Invalid NEXT_HOP Attribute. The NEXT_HOP attribute field is not syntactically correct.

statID	Usage
bgpUpdateErrorOptionalAttribError	Optional Attribute Error. The number of optional attribute incorrect values received.
bgpUpdateErrorOriginAttribInvalid	Invalid ORIGIN Attribute. When the ORIGIN attribute has an undefined value.
bgpUpdateErrorReceived	Update Errors Received. Total number of Update errors received.
bgpUpdateErrorSent	Update Errors Sent. Total number of Update errors sent.
bgpUpdateErrorSubUnspecified	Invalid Update Suberror Unspecified. If the error subcode is not defined, a zero identifies this Update message error as unspecified.
bgpUpdateErrorUnknownWellKnownAttrib	Unrecognized Well-known Attribute. If any of the mandatory well-known attributes are not recognized.
bgpUpdateReceived	Updates Received. Total number of BGP route updates received.
bgpUpdateSent	Updates Sent. Total number of BGP route updates sent.
bgpvalidOpenUnsupportVersion	Unsupported Version Received. When the Version Field contains an unsupported version number.

Specific errors include:

- No connection to a chassis.
- Invalid port number.
- A network problem has occurred.

bgp4StatsQuery set *chasID cardID portID*

No function is currently performed.

bgp4StatsQuery setDefault

Sets default values for all configuration options.

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4VpnBgpAdVplsRange

bgp4VpnBgpAdVplsRange — gets the BGP AD VPLS Range on a Neighbor range.

SYNOPSIS

bgp4VpnBgpAdVplsRange *subcommand options*

DESCRIPTION

The **bgp4VpnBgpAdVplsRange** command is used to get VPLS configured for this BGP/BGP+ Neighbor, for use with BGP/BGP+-VPLS.

STANDARD OPTIONS

statName

(Read-only.) The name of the statistic retrieved.

statValue

(Read-only.) The value of the statistic, as a string.

COMMANDS

The **bgp4VpnBgpAdVplsRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4VpnBgpAdVplsRange addBgpAdVplsRange

Adds a BGP Ad VPLS Range to the BGP4 Neighbor.

bgp4VpnBgpAdVplsRange delBgpAdVplsRange

Deletes a BGP Ad VPLS Range to the BGP4 Neighbor.

bgp4VpnBgpAdVplsRange getBgpAdVplsRange

Allows to get a BGP Ad VPLS Range to the BGP4 Neighbor.

bgp4VpnBgpAdVplsRange setBgpAdVplsRange

Allows to set a BGP Ad VPLS Range to the BGP4 Neighbor.

bgp4VpnBgpAdVplsRange getFirstBgpAdVplsRange

Allows to get the first BGP Ad VPLS Range in the BGP4 Neighbor.

bgp4VpnBgpAdVplsRange getNextBgpAdVplsRange

Allows to get the next BGP Ad VPLS Range in the BGP4 Neighbor.

bgp4VpnBgpAdVplsRange clearAllBgpAdVplsRanges

Clears all BGP Ad VPLS Range in the BGP4 Neighbor.

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4McastSenderSite

bgp4McastSenderSite — configures the BGP Multicast Sender Site options.

SYNOPSIS

bgp4McastSenderSite *subcommand options*

DESCRIPTION

The **bgp4McastSenderSite** command is used to configure BGP Multicast Sender Site options for this BGP/BGP+ Neighbor.

STANDARD OPTIONS

enabled

Enables or disables the multicast sender site.

addressFamily

Indicates the IPv4/IPv6 interface id of the router.

startGroupAddress

The first IPv4 or IPv6 Multicast group address in the range of group addresses included in this Register message.

groupMaskWidth

The number of bits in the network mask used with the Group Address.

groupAddressCount

The number of group addresses to be included in the Register message.

sourceGroupMapping

Indicates the source group mapping. One of:

- fullyMeshed
- oneToOne

startSourceAddress

The first IPv4 or IPv6 source address to be included in this Register message. (IPv4 Multicast addresses are not valid for sources.)

sourceMaskWidth

The number of bits in the mask applied to the Source Address. (The masked bits in the Source Address form the address prefix.)

The default value is 32. The valid range is 1 to 128, depending on address family type.

Used for (S,G) Type and (S,G, rpt) only

sourceAddressCount

The number of multicast source addresses to be included. The maximum number of valid possible addresses depends on the values of the Source Address and the Source Mask Width.

The default value is 0.

sPmsiTrafficGroupId

Creates traffic using MPLS Labels of S-PMSI Tunnel and S-PMSI Upstream Assigned Label.

sPmsiRsvpP2mpId

The P2MP Id represented in IP address format.

sPmsiRsvpP2mpIdAsNumber

The P2MP Id represented in integer format.

sPmsiRsvpP2mpIdStep

Indicates the P2MP ID. This accepts only integer values.

sPmsiRsvpTunnelId

The first Tunnel ID value in the range of Tunnel IDs.

sPmsiRsvpTunnelIdStep

Indicates the P2MP ID. This accepts only integer values.

sPmsiTunnelCount

The total count of the S-PMSI RSVP Tunnel Count.

useUpstreamAssignedLabel

Indicates whether upstream label as configured be used or not.

If this field is false, then MPLS Assigned Upstream Label and MPLS Assigned Upstream Label Step fields are disabled.

mplsAssignedUpstreamLabel

S-PMSI A-D route is sent with this Upstream Label. This is applicable only if Use Upstream Assigned Label is true.

mplsAssignedUpstreamLabelStep

This helps to assign unique upstream assigned label for each flow. This is applicable only if Use Upstream Assigned Label is true.

sendTriggeredSourceActiveAdRoute

If true, allows to send the Source Active A-D Route after receiving Source Tree Join C-Multicast route.

setLeafInformationRequiredBit

This is used to send S-PMSI A-D Route with Leaf Information Required bit Set.

COMMANDS

bgp4McastSenderSite addbgp4McastSenderSite

Adds a BGP Multicast Sender Site to the BGP4 Neighbor.

bgp4McastSenderSite delbgp4McastSenderSite

Deletes a BGP Multicast Sender Site from the BGP4 Neighbor.

bgp4McastSenderSite getbgp4McastSenderSite

Allows to get a BGP Multicast Sender Site to the BGP4 Neighbor.

bgp4McastSenderSite setbgp4McastSenderSite

Allows to set a BGP Multicast Sender Site to the BGP4 Neighbor.

bgp4McastSenderSite getbgp4McastSenderSite

Allows to get the first BGP Multicast Sender Site in the BGP4 Neighbor.

bgp4McastSenderSite getbgp4McastSenderSite

Allows to get the next BGP Multicast Sender Site in the BGP4 Neighbor.

bgp4McastSenderSite clearbgp4McastSenderSite

Clears all BGP Multicast Sender Site from the BGP4 Neighbor.

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4Neighbor NAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4McastReceiverSite

bgp4McastReceiverSite — configures the BGP Multicast Receiver Site options.

SYNOPSIS

bgp4McastReceiverSite *subcommand options*

DESCRIPTION

The **bgp4McastReceiverSite** command is used to configure BGP Multicast Receiver Site options for this BGP/BGP+ Neighbor.

STANDARD OPTIONS

enabled

Enables or disables use of the multicast Sender site.

cMastRouteType

The C-Multicast Route Type. One of:

- sourceTreeJoin
- sharedTreeJoin

addressFamily

Indicates the IPv4/IPv6 interface id of the router.

startGroupAddress

The first IPv4 or IPv6 Multicast group address in the range of group addresses included in this Register message.

groupMaskWidth

The number of bits in the network mask used with the Group Address.

groupAddressCount

The number of group addresses to be included in the Register message.

sourceGroupMapping

Indicates the source group mapping. One of:

- fullyMeshed
- oneToOne

startSourceAddress

The first IPv4 or IPv6 source address to be included in this Register message.

(IPv4 Multicast addresses are not valid for sources.)

sourceMaskWidth

The number of bits in the mask applied to the Source Address. (The masked bits in the Source Address form the address prefix.)

The default value is 32. The valid range is 1 to 128, depending on address family type.

Used for (S,G) Type and (S,G, rpt) only.

sourceAddressCount

The number of multicast source addresses to be included. The maximum number of valid possible addresses depends on the values for the Source Address and the Source Mask Width.

The default value is 0.

supportLeafAdRoutesSending

If true, helps IXIA to send Leaf A-D Route on receiving a S-PMSI A-D Route with the Leaf Information Required flag set.

If false, IXIA shall not send the Leaf A-D Route even if such Update message is received.

sendTriggeredCmulticastRoute

This helps to send Source Tree Join C-Multicast route after receiving Source Active A-D route. This is also required by Shared Tree Join C-Multicast route to send Source Tree Join after receiving Source Active A-D Route.

COMMANDS

bgp4McastReceiverSite addbgp4McastReceiverSite

Adds a BGP Multicast Receiver Site to the BGP4 Neighbor.

bgp4McastReceiverSite delbgp4McastReceiverSite

Deletes a BGP Multicast Receiver Site from the BGP4 Neighbor.

bgp4McastReceiverSite getbgp4McastReceiverSite

Allows to get a BGP Multicast Receiver Site to the BGP4 Neighbor.

bgp4McastReceiverSite setbgp4McastReceiverSite

Allows to set a BGP Multicast Receiver Site to the BGP4 Neighbor.

bgp4McastReceiverSite getbgp4McastReceiverSite

Allows to get the first BGP Multicast Receiver Site in the BGP4 Neighbor.

bgp4McastReceiverSite getbgp4McastReceiverSite

Allows to get the next BGP Multicast Receiver Site in the BGP4 Neighbor.

bgp4McastReceiverSite clearbgp4McastReceiverSite

Clears all BGP Multicast Receiver Site from the BGP4 Neighbor.

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4Neighbor NAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4UserDefinedAfiSafi

bgp4UserDefinedAfiSafi — configures the AFI/SAFI route.

SYNOPSIS

bgp4UserDefinedAfiSafiRoute *subcommand options*

DESCRIPTION

The **bgp4UserDefinedAfiSafi** command is used to configure BGP Afi/Safi routes for this BGP/BGP+ Neighbor.

STANDARD OPTIONS

userDefinedAfiSafi Route

The Afi/Safi routes are being added.

COMMANDS

bgp4UserDefinedAfiSafi addBgpUserDefinedAfiSafiRoute

Allows to add the Afi/Safi routes to the BGP4 Neighbor

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4UserDefinedAfiSafiRoute

bgp4UserDefinedAfiSafiRoute — configures the AFI/SAFI route options.

SYNOPSIS

bgp4UserDefinedAfiSafiRoute *subcommand options*

DESCRIPTION

The **bgp4UserDefinedAfiSafiRoute** command is used to configure BGP Afi/Safi route options for this BGP/BGP+ Neighbor.

STANDARD OPTIONS

enabled

Enables or disables use of the afi/safi route options.

length

The data is padded up to length with left alignment otherwise chopped till length.

data

Data to be transmitted for AFI/SAFI, and regular enable-disable.

COMMANDS

NA

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*, *NAME - bgp4VpnTarget*

NAME - bgp4VpnL2Site

bgp4VpnL2Site — configures a BGP VPLS L2 VPN Customer Edge (CE) site.

SYNOPSIS

bgp4VpnL2Site *subcommand options*

DESCRIPTION

The *bgp4VpnL2Site* command holds information about a BGP VPLS Layer 2 VPN site. A site is set of networks connected to an internal neighbor interface. Sites specified with this command are added to internal neighbors using the *NAME - bgp4Neighbor addL2Site* command. A Layer 2 VPN site includes one list:

- A list of Label Blocks.

The optional BGP4 test package must be installed.

STANDARD OPTIONS

clusterList

The list of BGP clusters through which the multicast routes have passed. *(default = {})*

enable *true | false*

Enables the use of this L2 VPN site. *(default = false)*

enableCluster *true | false*

If *true*, enables the use of BGP route reflection clusters for multicast VPN route distribution. *(default = false)*

enableControlWord *true | false*

Enables the use of a control word, as part of the extended community information. (One of the control flags.) *(default = false)*

enableSequencedDelivery *true | false*

Enables the use of sequenced delivery of frames, as part of the extended community information. (One of the control flags.) *(default = false)*

routeDistinguisherAS

If the *routeDistinguisherType* was set to *routeDistinguisherAS*, this is the 2-byte AS unsigned integer in the Administrator subfield of the Value field of the Route Distinguisher (RD). It is the "Global" part of the RD. *(default = 0)*

routeDistinguisherAssigned

The Assigned Number sub-field of the Value field of the Route Distinguisher. It is a number from a "numbering space," which the enterprise administers, for a given IP address or ASN space. It is the "Local" part of the RD. *(default = 0)*

routeDistinguisherIP

If the *routeDistinguisherType* was set to *routeDistinguisherIP*, this is the 4-byte IP address in the Administrator subfield of the Value field of the Route Distinguisher (RD). It is the "Global" part of the RD. *(default = 0.0.0.0)*

noOfL2Site

Signifies the number of L2 sites.

siteIdIncrement

Signifies the increment of site identifier.

targetAssignedNumberIncrement

Signifies the increment of target assigned number.

distinguishIpIncrement

Signifies the increment of IP that is distinguished.

distinguishNumberIncrementAs

Signifies the increment of distinguish number.

distinguishAssignedIncrement

Signifies the increment of assigned distinguished value.

enableL2SiteAsTrafficEndpoint

If true, enables L2 site as traffic endpoint.

routeDistinguisherType

Indicates the type of administrator field used in route distinguisher that will be included in the route announcements. One of:

Option	Value	Usage
routeDistinguisherTypeAS	0	*(default)* The Administrator subfield is 2 bytes in length, and contains an Autonomous System number (ASN).
routeDistinguisherTypeIP	1	The Administrator subfield is 4 bytes in length, and contains an IP address

routeTargetAS

If the *routeTargetType* was set to the value for AS, this is the 2-byte IP AS number in the Local Administrator subfield of the Value field of the Route Target (RT). *(default = 0)*

routeTargetAssigned

The Local Administrator sub-field of the Value field of the Route Target. It is a number from a "numbering space," which the enterprise administers, for a given IP address or ASN space. *(default = 0)*

routeTargetIP

If the *routeTargetType* was set to the value for IP, this is the 4-byte IP address in the Local Administrator subfield of the Value field of the Route Target (RT). *(default = 0.0.0.0)*.

routeTargetType

The type of BGP Route Target community. One of: AS or IP. *(default = AS)*

siteId

The identifier for the BGP L2 Site. *(default = 0)*

COMMANDS

The *bgp4VpnL2Site* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4VpnL2Site addVpnLabelBlock *vpnLabelBlockName*

Adds the label block created with the *NAME - bgp4VpnLabelBlock* command to this router. The label block is tagged with the *vpnLabelBlockName*.

bgp4VpnL2Site cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bgp4VpnL2Site* command.

bgp4VpnL2Site clearAllVpnLabelBlocks

Clears all of the label blocks associated with the command.

bgp4VpnL2Site config *option value*

Modify the configuration options of the *bgp4VpnL2Site*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4VpnL2Site*.

bgp4VpnL2Site delVpnLabelBlock *[vpnLabelBlockName]*

Deletes the label block with the tag *vpnLabelBlockName,* if specified. Otherwise, the currently accessed label block is deleted.

bgp4VpnL2Site getFirstVpnLabelBlock

Gets the first label block from the list as the `current' item. The contents of the item are available in the *NAME - bgp4VpnLabelBlock*command.

bgp4VpnL2Site getLearnedRouteList

This subcommand should be called after *requestLearnedRoutes*. It must be called until it returns TCL_OK, usually with a wait between calls.

bgp4VpnL2Site getNextVpnLabelBlock

Gets the next label block in the list as the `current' item. The contents of the item are available in the *NAME - bgp4VpnLabelBlock* command.

bgp4VpnL2Site getVpnLabelBlock *[vpnLabelBlockName]*

Gets the label block with the tag *vpnLabelBlockName* from the list. The contents of the item are available in the *NAME - bgp4VpnLabelBlock* command.

bgp4VpnL2Site requestLearnedRoutes

Requests that the learned routes for this site be retrieved from the protocol server. The *getLearnedRouteList* subcommand must be called after this subcommand to determine when the complete list of routes has been retrieved.

bgp4VpnL2Site setDefault

Sets default values for all configuration options.

bgp4VpnL2Site setVpnLabelBlock *[vpnLabelBlockName]*

Overwrites the label block with the tag*,* if specified. Otherwise the current route range is overwritten. The data used is from the contents of the label block in the *NAME - bgp4VpnLabelBlock*command.

EXAMPLES

```
# example:

bgp4VpnLabelBlock setDefault

bgp4VpnLabelBlock config -enable true

bgp4VpnLabelBlock config -offset 0

bgp4VpnLabelBlock config -startBlock 100

bgp4VpnLabelBlock config -numberOfVpnLabels 10

bgp4VpnL2Site addVpnLabelBlock LabelBlock1

bgp4VpnL2Site setDefault

bgp4VpnL2Site config -enable true

bgp4VpnL2Site config -siteId 5

bgp4VpnL2Site config -enableCluster false

bgp4VpnL2Site config -clusterList

“1.2.3.4,5.1.2.4”

bgp4VpnL2Site config -enableControlWord false

bgp4VpnL2Site config -enableSequencedDelivery

false

bgp4VpnL2Site config -mtu 0

bgp4VpnL2Site config -routeTargetType 0

bgp4VpnL2Site config -routeTargetAS 2

bgp4VpnL2Site config -routeTargetAssigned 2
```

```
bgp4VpnL2Site config -routeTargetIP
"0.0.0.0"
bgp4VpnL2Site config -routeDistinguisherType 0
bgp4VpnL2Site config -routeDistinguisherAS 2
bgp4VpnL2Site config -routeDistinguisherAssigned 3
bgp4VpnL2Site config -routeDistinguisherIP
"0.0.0.0"
bgp4Neighbor addL2Site l2Site1
# LearnedInfo commandset for L2VPN:
To retrieve LearnedRoute info from under a Neighbor , the
following tcl commands are to be used:
bgp4Neighbor requestLearnedRoutes
bgp4Neighbor getLearnedRouteList
bgp4LearnedRoute getFirst bgp4FamilyIpVpls
showCmd bgp4LearnedRoute
# example:
bgp4LearnedRoute cget -blockOffset: 0
bgp4LearnedRoute cget -blockSize: 300
bgp4LearnedRoute cget -description: RD: 1:6, Site ID: 6, Block
Offset: 0, Block Size: 300, Label Base: 100, CW : Disabled, Seq
Del: Disabled, , NHop: 2.2.2.3
bgp4LearnedRoute cget -enableControlWord: 0
bgp4LearnedRoute cget -enableSequencedDelivery: 0
bgp4LearnedRoute cget -labelBase: 100
bgp4LearnedRoute cget -neighbor: 2.2.2.2
bgp4LearnedRoute cget -nextHop: 2.2.2.3
bgp4LearnedRoute cget -routeDistinguisher: 1:6
bgp4LearnedRoute cget -siteId: 6
# To get LearnedRoute Info for a L2VPN Site, the commandset is as
follows:
bgp4VpnL2Site requestLearnedRoutes
bgp4VpnL2Site getLearnedRouteList
bgp4LearnedRoute getFirst bgp4FamilyIpVpls
showCmd bgp4LearnedRoute
```

```
# example:

bgp4LearnedRoute cget -blockOffset: 300

bgp4LearnedRoute cget -blockSize: 25

bgp4LearnedRoute cget -description: RD: 1:6, Site ID: 6, Block

Offset: 300, Block Size: 25, Label Base: 200, CW : Disabled, Seq

Del: Disabled, , NHop: 2.2.2.3

bgp4LearnedRoute cget -enableControlWord: 0

bgp4LearnedRoute cget -enableSequencedDelivery: 0

bgp4LearnedRoute cget -labelBase: 200

bgp4LearnedRoute cget -neighbor: 2.2.2.2

bgp4LearnedRoute cget -nextHop: 2.2.2.3

bgp4LearnedRoute cget -routeDistinguisher: 1:6

bgp4LearnedRoute cget -siteId: 6
```

SEE ALSO

NAME - bgp4LearnedRoute, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery*

NAME - bgp4VpnL3Site

bgp4VpnL3Site—configures an L3 VPN site.

SYNOPSIS

bgp4VpnL3Site subcommand options

DESCRIPTION

The bgp4VpnL3Site command holds information about a VPN Layer 3 site. A site is set of networks connected to an internal neighbor interface. Sites specified with this command are added to internal neighbors using the bgp4Neighbor addL3Site command. A VPN Layer 3 site includes three lists:

- A list of VPN route ranges. These route ranges are advertised to other routers.
- A list of VPN targets, which will receive routing tables.
- A list of VPN import targets, which are used to filter incoming route tables.

The optional BGP4 test package must be installed.

STANDARD OPTIONS

clusterList

The list of BGP clusters through which the multicast routes have passed. *(default = {})*

distinguisherAssigned- Number

The Assigned Number sub-field of the Value field of the MVPN Route Distinguisher. It is a number from a "numbering space," which the enterprise administers, for a given IP address or ASN space. It is the "Local" part of the RD. *(default = 0)*

distinguisherAsNumber

If the *distinguisherType* was set to *bgp4DistinguisherTypeAS*, this is the 2-byte AS unsigned integer in the Administrator subfield of the Value field of the MVPN Route Distinguisher (RD). It is the "Global" part of the RD. *(default = 0)*

distinguisherIpAddress

If the *distinguisherType* was set to *bgp4DistinguisherTypeIP*, this is the 4-byte IP address in the Administrator subfield of the Value field of the MVPN Route Distinguisher (RD). It is the "Global" part of the RD. *(default = 0.0.0.0)*

distinguisherType

Indicates the type of administrator field used in route distinguisher that will be included in the route announcements. One of:

Option	Value	Usage
bgp4DistinguisherTypeAS	0	(default) The Administrator subfield is 2 bytes in length, and contains an Autonomous System number (ASN).

Option	Value	Usage
bgp4DistinguisherTypeIP	1	The Administrator subfield is 4 bytes in length, and contains an IP address.
bgp4DistinguisherTypeAS4Byte	2	The Administrator subfield is 4 bytes in length, and contains an Autonomous System number (ASN).

enable true | false

Enables the use of this L3 site. *(default = false)*

enableCluster true | false

If true, enables the use of BGP route reflection clusters for multicast VPN route distribution. *(default = false)*

enableVpnMulticast true | false

Enables the use of Multicast VRFs (MVRFs). *(default = false)*

groupAddress

The IP address for the Multicast Group. The default value is the default MDT group address, used as the Multicast Group address used as the destination for the MVPN tunnel. *(default = 239.1.1.1)*

mplsAssignedUpstreamLabel

S-PMSI A-D route is sent with this Upstream Label. This is applicable only if UseUpstreamAssignedLabel is true.

vrfCount

Number of VRFs within the VRF Range.

multicastGroupAddressStep

The increment step to be added to each additional Multicast Group Address.

enableUmhRtSameAsL 3SiteRt

Enables the use of UMH route same as L3 site rate.

enableUmhTargetListSa meAsL3SiteTargetList

Enables the use of UMH target list same as L3 site target list.

enableUmhTargetListSameAsL3SiteTargetList

Enables the use of UMH target list same as L3 site target list.

COMMANDS

The *bgp4VpnL3Site* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4VpnL3Site addImportTarget

Adds the import target site created with the *NAME - bgp4VpnTarget* command to this router. Specific errors are:

- Invalid VPN target parameters.
- The VPN target already exists. (Delete an old entry before adding it again.)

bgp4VpnL3Site addVpnRouteRange vpnRouteRangeLocalId

Adds the route range created with the *NAME - bgp4VpnRouteRange* command to this

router. The route range is tagged with the *vpnRouteRangeLocalId*. Specific errors are:

- Invalid route range parameters.
- The route range already exists. (Delete an old entry before adding it again.)

bgp4VpnL3Site addVpnTarget

Adds the L3 site created with the *NAME - bgp4VpnTarget* command to this router.Specific errors are:

- Invalid VPN target parameters.
- The VPN target already exists. (Delete an old entry before adding it again.)

bgp4VpnL3Site cget option

Returns the current value of the configuration option given by option. Option may have any of the values accepted by the *bgp4VpnL3Site* command.

bgp4VpnL3Site clearAllImportTargets

Clears all of the import targets associated with the command.

bgp4VpnL3Site clearAllVpnRouteRanges

Clears all of the route ranges associated with the command.

bgp4VpnL3Site clearAllVpnTargets

Clears all of the VPN targets associated with the command.

bgp4VpnL3Site config option value

Modify the configuration options of the *bgp4VpnL3Site*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4VpnL3Site*.

bgp4VpnL3Site delImportTarget

Deletes the currently accessed import target.

bgp4VpnL3Site delVpnRouteRange [vpnRouteRangeLocalId]

Deletes the route range with the tag *vpnRouteRangeLocalId*, if specified. Otherwise the currently accessed route range is deleted.

bgp4VpnL3Site delVpnTarget

Deletes the currently accessed VPN target.

bgp4VpnL3Site getFirstImportTarget

Gets the first import target from the list as the 'current' item. The contents of the item are available in the *NAME - bgp4VpnTarget* command. Specific errors are:

- The neighbor does not have any import targets.

bgp4VpnL3Site getFirstVpnRouteRange

Gets the first route range from the list as the 'current' item. The contents of the item are available in the NAME - bgp4VpnRouteRange command. Specific errors are:

- The site does not have any route ranges.

bgp4VpnL3Site getFirstVpnTarget

Gets the first VPN target from the list as the 'current' item. The contents of the item are available in the *NAME - bgp4VpnTarget* command. Specific errors are:

- The site does not have any VPN targets.

bgp4VpnL3Site getLearnedRouteList

This subcommand should be called after *requestLearnedRoutes*. It must be called until it returns TCL_OK, usually with a wait between calls. Specific errors are:

- The list has not been completely retrieved yet.

bgp4VpnL3Site getNextVpnRouteRange

Gets the next route range in the list as the 'current' item. The contents of the item are available in the *NAME - bgp4VpnRouteRange* command. Specific errors are:

- The neighbor does not have any more route ranges.

bgp4VpnL3Site getNextImportTarget

Gets the next import target in the list as the 'current' item. The contents of the item are available in the *NAME - bgp4VpnTarget* command. Specific errors are:

- The site does not have any more VPN targets.

bgp4VpnL3Site getVpnRouteRange vpnRouteRangeLocalId

Gets the route range with the tag *vpnRouteRangeLocalId* from the list. The contents of the item are available in the *NAME - bgp4VpnRouteRange* command. Specific errors are:

- The site does not have any route ranges.

bgp4VpnL3Site getNextVpnTarget

Gets the next VPN target in the list as the 'current' item. The contents of the item are available in the *NAME - bgp4VpnTarget* command. Specific errors are:

- The site does not have any more VPN targets.

bgp4VpnL3Site requestLearnedRoutes

Requests that the learned routes for this site be retrieved from the protocol server. The *getLearnedRouteList* subcommand must be called after this subcommand to determine when the complete list of routes has been retrieved.

bgp4VpnL3Site setDefault

Sets default values for all configuration options.

bgp4VpnL3Site setVpnRouteRange [vpnRouteRangeLocalId]

Overwrites the route range with the tag *vpnRouteRangeLocalId*, if specified. Otherwise the current route range is overwritten. The data used is from the contents of the route range in the *NAME - bgp4VpnRouteRange* command. Specific errors are:

- The neighbor does not have any route ranges.

bgp4VpnL3Site addMcastSenderSite mcastSenderSite

Adds the multicast sender site values for the bgp4 neighbor.

bgp4VpnL3Site addMcastReceiverSite mcastSenderSite

Adds the multicast receiver site values for the bgp4 neighbor.

bgp4VpnL3Site addUmhSelectionRouteRange

Adds the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site delUmhSelectionRouteRange

Deletes the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site getUmhSelectionRouteRange

Gets the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site setUmhSelectionRouteRange

Sets the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site getFirstUmhSelectionRouteRange

Gets the first UMH selection route range value for the bgp4 neighbor.

bgp4VpnL3Site getNextUmhSelectionRouteRange

Gets the next UMH selection route range value for the bgp4 neighbor.

bgp4VpnL3Site clearAllUmhSelectionRouteRanges

Clears all the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site addUmhSelectionRouteRange

Adds the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site delUmhSelectionRouteRange

Deletes the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site getUmhSelectionRouteRange

Gets the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site setUmhSelectionRouteRange

Sets the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site getFirstUmhSelectionRouteRange

Gets the first UMH selection route range value for the bgp4 neighbor.

bgp4VpnL3Site getNextUmhSelectionRouteRange

Gets the next UMH selection route range value for the bgp4 neighbor.

bgp4VpnL3Site clearAllUmhSelectionRouteRanges

Clears all the UMH selection route range values for the bgp4 neighbor.

bgp4VpnL3Site addUmhTarget

Adds the UMH target value for the bgp4 neighbor.

bgp4VpnL3Site delUmhTarget

Deletes the UMH target value for the bgp4 neighbor.

bgp4VpnL3Site getFirstUmhTarget

Gets the first UMH target value for the bgp4 neighbor.

bgp4VpnL3Site getNextUmhTarget

Gets the next UMH target value for the bgp4 neighbor.

bgp4VpnL3Site clearAllUmhTargets

Clears all the UMH target values for the bgp4 neighbor.

bgp4VpnL3Site addUmhImportTarget

Adds UMH import targets for the bgp4 neighbor.

bgp4VpnL3Site delUmhImportTarget

Deletes UMH import targets for the bgp4 neighbor.

bgp4VpnL3Site getFirstUmhImportTarget

Gets first UMH import target value for the bgp4 neighbor.

bgp4VpnL3Site getNextUmhImportTarget

Gets next UMH import target value for the bgp4 neighbor.

bgp4VpnL3Site clearAllUmhImportTargets

Clears all UMH import target values for the bgp4 neighbor.

EXAMPLES

See examples under *NAME - bgp4VpnTarget*

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery*, *NAME - bgp4VpnTarget*, *bgp4UmhTarget*

NAME - bgp4UmhTarget

bgp4UmhTarget—signifies the UMH target information.

SYNOPSIS

bgp4UmhTarget *subcommand options*

DESCRIPTION

The *bgp4UmhTarget* command helps to import large amount of UMH target information from a text file.

STANDARD OPTIONS

type

Indicates the type of administrator field used in the target that will be included in route announcements.

ipAddress

If *type* is set to *bgp4TargetTypeIP*, this is the 4-byte IP address in the administrator subfield of the value field of the target. It is the global part of the target. *(default = 0.0.0.0)*

asNumber

If *type* is set to *bgp4TargetTypeAS*, this is the 2-byte AS number in the administrator subfield of the value field of the target. It is the global part of the target. *(default = 0)*

assignedNumber

The assigned number subfield of the value field of the target. It is a number from a numbering space, which is maintained by the enterprise administers for a given IP address or ASN space. It is the local part of the target. *(default = 0)*

ipAddressStep

The increment value for the IP address.

asNumberStep

The increment step for a number.

assignedNumberStep

The increment step for the assigned number.

COMMANDS

The umhTarget command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4UmhTarget cget option

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bgp4UmhTarget* command.

bgp4UmhTarget configure option value

Modify the configuration options of the *bgp4UmhTarget*. If no option is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4UmhTarget*.

bgp4UmhTarget setDefault

Sets default values for all configuration options.

SEE ALSO

bgp4UmhSelectionRouteRange, *bgp4UmhImportTarget*

NAME - bgp4UmhImportTarget

bgp4UmhTarget—signifies the UMH target information.

SYNOPSIS

bgp4UmhTarget *subcommand options*.

DESCRIPTION

The *bgp4UmhImportTarget* command helps to import large amount of UMH target information from a text file.

STANDARD OPTIONS

type

Indicates the type of administrator field used in the target that will be included in route announcements.

ipAddress

If type is set to *bgp4TargetTypeIP*, this is the 4-byte IP address in the administrator subfield of the value field of the target. It is the global part of the target. *(default = 0.0.0.0)*

asNumber

If type is set to *bgp4TargetTypeAS*, this is the 2-byte AS number in the administrator subfield of the value field of the target. It is the global part of the target. *(default = 0)*

assignedNumber

The assigned number subfield of the value field of the target. It is a number from a numbering space, which is maintained by the enterprise administers for a given IP address or ASN space. It is the local part of the target. *(default = 0)*

ipAddressStep

The increment value for the IP address.

asNumberStep

The increment step for a number.

assignedNumberStep

The increment step for the assigned number.

COMMANDS

The umhTarget command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4UmhTarget cget option

Returns the current value of the configuration option given by *option*. Option may have any of the values accepted by the *bgp4UmhTarget* command.

bgp4UmhTarget configure option value

Modify the configuration options of the *bgp4UmhTarget*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4UmhTarget*.

bgp4UmhTarget setDefault

Sets default values for all configuration options.

SEE ALSO

bgp4UmhTarget, *bgp4UmhSelectionRouteRange*

Name - bgp4UmhSelectionRouteRange

bgp4UmhSelectionRouteRange—imports the UMH selection route range information.

SYNOPSIS

bgp4UmhSelectionRouteRange *subcommand options*

DESCRIPTION

The *bgp4UmhSelectionRouteRange* command helps to import large amount of UMH target information from a text file.

STANDARD OPTIONS

enable true | false

Enables the use of this route range as an advertised range. *(default = false)*

labelSpaceId

The label space to which the label associated with advertised routes is associated. *(default = 0)*

labelStart

The first label value available in the label space (range). *(default = 16)*

labelEnd

The last label value available in the label space (range). *(default = 1,048,575)*

labelStep

For Increment label mode. The increment step used when assigning successive label values. *(default = 1)*

labelMode

The mode for assigning labels to routes. One of:

Option	Value	Usage
bgp4VpnFixedLabel	0	the same label value is used for all packets.
bgp4VpnIncrementLabel	1	*(default)* the label value is incremented by the labelStep value for each successive packet

distinguisherType

Indicates the type of administrator field used in route distinguisher that will be included in the route announcements. One of:

Option	Value	Usage
bgp4DistinguisherTypeAS	0	*(default)* The Administrator subfield is 2 bytes in length, and contains an Autonomous System number (ASN)
bgp4DistinguisherTypeIP	1	The Administrator subfield is 4 bytes in length, and contains an IP address

Option	Value	Usage
bgp4DistinguisherTypeAS4Byte		The Administrator subfield is 4 bytes in length, and contains an Autonomous System number (ASN), which is 4 bytes in length.

distinguisherIpAddress

If *distinguisherType* is set to *bgp4DistinguisherTypeIP*, this is the 4-byte IP address in the administrator subfield of the value field of the VPN Route Distinguisher. It is the global part of the route distinguisher. *(default = 0.0.0.0)*

distinguisherAsNumber

If *distinguisherType* is set to *bgp4DistinguisherTypeAS*, this is the 2-byte AS number in the administrator sub-field of the value field of the VPN Route Distinguisher. It is the global part of the route distinguisher. *(default = 0)*

distinguisherAsNumberStep

The increment step for for the distinguisher AS number.

distinguisherAsNumberStepAcrossVrfs

he increment step for per VRF distinguisher AS number within the VRF Range

distinguisherAssigned Number

The assigned number subfield of the value field of the VPN route distinguisher. It is a number from a numbering space which is maintained by the enterprise administers for a given IP address or ASN space. It is the local part of the route distinguisher. *(default = 0)*

distinguisherAssigned-NumberStep

The increment step for for the distinguisher assigned number.

distinguisherAssigned-NumberStepAcrossVrfs

The increment step for per VRF distinguisher assigned number within the VRF Range.

distinguisherCount

The number of times that the increment step will be used. *(default = 1)*

distinguisherMode

Specifies which part of the route distinguisher you want to increment. One of:

Option	Value	Usage
vpnDistinguisher IncrementGlobalPart	0	the administrative part of the route distinguisher.
vpnDistinguisher IncrementLocalPart	1	*(default)* the assigned part of the route distinguisher

distinguisherStep

The size of the increment step to be used with the part of the route distinguisher which will be incremented. *(default = 1)*

distinguisherIpAddressStep

The increment step for for the distinguisher IP address.

distinguisherIpAddressStepAcrossVrfs

The increment step for per VRF distinguisher IP address within the VRF Range.

firstRoute

The first route of the route range, in IPv4/IPv6 format.

(default = 0.0.0.0 for IPv4)

(default = 0:0:0:0:0:0:0:0 for IPv6)

ipType

The IP addressing type of ipAddress, one of:

Option	Value	Usage
addressTypeIpV4	17	(default) An IPv4 address.
addressTypeIpV6	18	An IPv6 address.

maskWidth

The network mask width for the route range (in bits). The valid range is from 0 to 32 bits. *(default = 24)*

maskWidthTo

The network mask width to the particular route range.

routeCountPerVrfs

Count of route ranges per VRF.

fromPacking

The minimum number of routes to pack into an UPDATE message. Random numbers are chosen from the range *fromPacking* to *toPacking*. *(default = 0)*

The following chart indicates the range of packing depending on the setting of this and the toPacking values:

fromPacking	toPacking	Packing Ranges
0	0	As many as possible
0	b	Random from 1 to b
a	0	Always a routes
a	b	Random from a to b

toPacking

The maximum number of routes to pack into an UPDATE message. Random numbers are chosen from the range *fromPacking* to *toPacking*. See the discussion under fromPacking above. *(default = 0)*.

enableRouteFlap true | false

Enables the flapping functions described by *routeFlapTime*, *routFlapDropTime*, *routesToFlapFrom*, and *routesToFlapTo*. *(default = false)*

routeFlapTime

During flapping operation, the time between flap cycles, expressed in seconds. *(default = 0)*

routeFlapDropTime

During flapping operation, the period expressed in seconds during which the route will be withdrawn from its neighbors. *(default = 0)*

routeStepAcrossVrfs

The route step across VRFs.

distinguisherCountPerVrf

The number of times that the increment step is used per VRF.

enableNextHop true | false

This is used for IPv4 traffic and enables the use of the NEXT_HOP attributes. *(default = false)*

enableAdvertiseNextHopAsV4

This option is exposed an option to advertise v4 next-hop address as v4 in case of IPv6 route range. It should be enabled if Route Range type is IPv6 and Enable NextHop is selected. An IPv4 next address should be set, either manually or by same as local IP as IPv4.

nextHopIpAddress

The IP address, in either IPv4 or IPv6 format of the next hop associated with the NEXT HOP attribute. *(default = 0.0.0.0)*

nextHopIpType

The type of IP address in *nextHopIpAddress*. One of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

nextHopMode

Indicates that the *nextHopIpAddress* may be incremented for each neighbor session generated for the range of neighbor addresses. Three types of increment are available:

Option	Value	Usage
bgpRouteNextHopFixed	0	Do not change next Hop values.
bgpRouteNextHopIncrement	1	*(default)* Increment entire address.
bgpRouteNextHopIncrementPerPrefix	2	The next hop will be increment for each

Option	Value	Usage
		advertised prefix.

nextHopSetMode

Indicates now to set the next hop IP address. One of:

Option	Value	Usage
bgpRouteNextHopSetManually	0	The value is read from *nextHopIpAddress*.
bgpRouteNextHopSetSameAsLocalIp	1	*(default)* The value is the same as the local IP address.

enableOrigin true | false

Enables the generation of an ORIGIN attribute, based on information in *originProtocol. (default = true)*

originProtocol

An indication of where the route entry originated. One of:

Option	Value	Usage
bgpOriginIGP	0	*(default)* Interior Gateway Protocol
bgpOriginEGP	1	Exterior Gateway Protocol
bgpOriginIncomplete	2	learned by some other means

localPref

The local preference value for the routes with the LOCAL PREF attribute. *(default = 0)*

med

The multi-exit discriminator value in the MULTI EXIT DISCRIMINATOR attribute. *(default = 0)*

enableLocalPref true | false

ables the generation of a LOCAL PREF attribute based on the information in *localPref*. This value should be set to true only for EBGP. *(default = false)*

enableMED true | false

Enables the generation of a MULTI EXIT DISCRIMINATOR attribute, based on the information in *MED. (default = false)*

enableCluster true | false

Enables the generation of the CLUSTER attribute list based on information in *clusterList*. *(default = false)*

enableCommunity true | false

Enables the generation of a COMMUNITY attribute list based on information in *communityList. (default = false)*

communityList

A list of communities associated with the route entry; associated with the COMMUNITY attribute. *(default = { })* The following values may also be used within the community list:

Option	Value	Usage
bgpCommunityNoExport	0xFFFFFF01	
bgpCommunityNoAdvertise	0xFFFFFF02	
bgpCommunityExportSubconfed	0xFFFFFF03	

aggregatorASNum

The AS associated with the aggregator router ID in the AGGREGATOR attribute. *(default = 0)*. Returns the specified dimensions of the aggregator router as an unsigned integer type.

aggregatorIDMode

Causes the AS field to be incremented for each neighbor session generated for the range of neighbor addresses in the AGGREGATOR attribute. *(default = 1)*

aggregatorIpAddress

The IP address of the router that aggregated two or more routes in the AGGREGATOR attribute. *(default = 0.0.0.0)*

asPathSetMode

Determines the handling of the Local AS# in generated AS sequences and sets and AS Confederation Sequences and Sets. The options are:

Option	Value	Usage
bgpRouteAsPath NoInclude	0	Do not include the Local AS#. When the route item is associated with an Internal router, this is only option available—and must be set in a bgp4RouteItem config -asPathSetMode 0.
bgpRouteAsPathIncludeAsSeq	1	(default) Include the Local AS# in the AS Path Sequence only.
bgpRouteAsPathIncludeAsSet	2	Include the Local AS# in the AS Path Set only.
bgpRouteAsPathIncludeAsSeqConf	3	Include the Local AS# in the AS Path Confederation Sequence only.
bgpRouteAsPathIncludeAsSetConf	4	Include the Local AS# in the AS Path Confederation Sequence only.

clusterList A

list of clusters that a particular route has passed through; associated with the CLUSTER attribute. Expressed as a TCL list {...}. *(default = { })*

enableASPath true | false

Enables the generation of AS Path related items (AsSet, AsSequence, AsConfedSet, and AsConfedSequence) based on information added from the *bgp4AsPathItem. (default = true)*

enableGenerate UniqueRoutes true | false

When set to 1, each router generates a different IP address range. A *bgp4InternalNeighborItem* or *bgp4ExternalNeighborItem* item must be configured with rangeCount > 1. When not enabled, each router will advertise the route range as is. When enabled, the first router advertises *numRoutes* routes starting at *networkAddress*, the next router advertises *numRoutes* routes starting at (*networkAddress* + *numRoutes*), and so on. *(default = false)*

enableIncludeLoopback true | false

If *true*, will include the loopback address (127.0.0.1) if it is in the generated network range. *(default = false)*

enableIncludeMulticast true | false

If *true*, will include multicast addresses if they are in the generated network range. The SAFI used for multicast addresses is dictated by the setting of the enableProperSafi option. *(default = false)*

enableOriginator true | false

Enables the generation of an ORIGINATOR attribute, based on information in originator. *(default = false)*

enableProperSafi true | false

If *false*, generated multicast addresses will use a unicast SAFI. If true, generated multicast addresses will use a proper multicast SAFI. *(default = false)*

originatorId

The router that originated a particular route; associated with the ORIGINATORID attribute. *(default = 0.0.0.0)*

enablePartialFlap

If true, enables partial flap.

routesToFlapFrom

During flapping operation, the first route prefix in a range to flap. *(default = 0)*

routesToFlapTo

During flapping operation, the last route prefix in a range to flap. *(default = 0)*

step

The step value.

enableUseTraditionalNlri

If true, enables use of traditional NLRI value.

enableIncludeSourceAsExtendedCommunity

If true, enables included source as extended community.

enableIncludeVrfRouteImportExtendedCommunity

If true, enables included VRF route import value as extended community.

delay

The total of delays on the path to the route/network, in microseconds. The valid range is 0 to 4294967295. *(default = 0)*

COMMANDS

The umhTarget command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4UmhSelectionRouteRange cget option

Returns the current value of the configuration option given by option. Option may have any of the values accepted by the *bgp4UmhSelectionRouteRange* command.

bgp4UmhSelectionRouteRange configure option value

Modify the configuration options of the *bgp4UmhSelectionRouteRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4UmhSelectionRouteRange*.

bgp4UmhSelectionRouteRange setDefault

Sets default values for all configuration options.

bgp4UmhSelectionRouteRange addASPathItem

Adds the AS Path items specified via *config* option calls to the *bgp4AsPathItem* list. The AS path item must have been previously configured through the use of the *bgp4AsPathItem* command. Specific errors are:

- Invalid AS path parameters

bgp4UmhSelectionRouteRange addExtendedCommunity

Adds an extended community attribute built with the *bgp4ExtendedCommunity* command.

bgp4UmhSelectionRouteRange clearASPathList

Clears all of the AS Path items associated with the command.

bgp4UmhSelectionRouteRange clearExtendedCommunityList

Clears all of the extended community items associated with the command.

bgp4UmhSelectionRouteRange getFirstASPathItem

Gets the first AS path item from the list. The contents of the item are available in the *bgp4AsPathItem* command. This and *getNextASPathItem* command may only be used if the route item was retrieved with the *bgp4Server* *getFirst/GetNext- Neighbor* command. Specific errors are:

- There are no items in the list.

bgp4UmhSelectionRouteRange getNextASPathItem

Gets the next AS path item in the list. The contents of the item are available in the *bgp4AsPathItem* command. Specific errors are:

- There are no more items in the list

bgp4UmhSelectionRouteRange getNextExtendedCommunity

Gets the next extended community item in the list. The contents of the item are available in the *bgp4ExtendedCommunity* command. Specific errors are:

- There are no more items in the list

bgp4UmhSelectionRouteRange delExtendedCommunity

Deletes an extended community attribute built with the *bgp4ExtendedCommunity* command.

bgp4UmhSelectionRouteRange getFirstExtendedCommunity

Gets the first extended community item in the list. The contents of the item are available in the *bgp4ExtendedCommunity* command. Specific errors are:

- There are no more items in the list

SEE ALSO

bgp4UmhTarget, *bgp4UmhImportTarget*

NAME - bgp4OpaqueRouteRange

bgp4OpaqueRouteRange — imports the opaque route range information.

SYNOPSIS

bgp4OpaqueRouteRange *subcommand options*

DESCRIPTION

The *bgpImportRouteRange* command helps to import large amount of route information from a text file. The route information in these files are generally real life information collected from the internet by the vendors.The imported route information is treated as opaque data and managed separately from the manually configured route ranges.

STANDARD OPTIONS

enabled ***true/false***

If true, enables the particular opaque route range.

importedFile

Indicates the location of the imported file.

interpretAsPath

The AS-Path present in the file is sent as AS-Sent in the Update messages.

sendMultiExitDiscovery

numberOfRoutes

The total number of opaque routes.

status

Indicates the status of the imported file.

COMMANDS

The opaqueRouteRange command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4Neighbor applyOpaqueRouteRange

Applies the concerned opaque route range to the specified route information.

SEE ALSO

NAME - bgp4Neighbor

NAME - bgp4RouteImportOptions

bgp4RouteImportOptions — configures the route import.

SYNOPSIS

bgp4RouteImportOptions *subcommand options*

DESCRIPTION

The *bgpImportRouteRange* command helps to import large amount of route information from a text file. The route information in these files are generally real life information collected from the internet by the vendors.The routeImportOptions is generally executed considering the arguments of the ImportRouteRange command.

STANDARD OPTIONS

advertise best routes

If checked, only the best routes are imbibed and advertised. The sub-optimal routes are ignored.

configureAsPath

If checked, the AS Path as present in the file is sent in the Update message as AS-SET.

numberOfRoutesPerBlock

Represents the maximum number of routes that can be forwarded in a block.

sendMultiExitDiscValue ***true/false***

If enabled, the BGP router sends the MED value of the attribute.

routeFileType

The file format of the import file.

COMMANDS

The opaqueRouteRange command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4Neighbor import

Imports the route ranges from the file specified as argument.

bgp4Neighbor getSupportedBGPRouteFileTypes

Displays the list of available route files formats.

SEE ALSO

NAME - bgp4Neighbor

NAME - bgp4VpnLabelBlock

bgp4VpnLabelBlock — configures an L2 VPN label block.

SYNOPSIS

bgp4VpnLabelBlock *subcommand options*

DESCRIPTION

The *bgp4VpnLabelBlock* command holds label information concerning a Layer 2 VPN site. Label blocks specified with this command are added to L2 VPN sites using the *NAME - bgp4VpnL2Site addVpnLabelBlock* command.

The optional BGP4 test package must be installed.

STANDARD OPTIONS

enable *true* | *false*

Enables the use of this L2 label block. *(default = false)*

numberOfVpnLabels

The number of L2 VPN labels contained in this label block.

offset

The offset in the packet — the location of the label block in the packet.

startBlock

The starting (first) label in the label block.

labelBlockOffsetIncrementAcrossL2Site

Signifies the label block offset increment across L2 site. Default is 1.

labelStartIncrementAcrossL2Site

It signifies the start of increment of label across L2 site. Default is 1.

totalLabelCount(RO)

Signifies the total label count. Default is 1.

COMMANDS

The *bgp4VpnLabelBlock* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4VpnL2Site cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bgp4VpnLabelBlock* command.

bgp4VpnL2Site config *option value*

Modify the configuration options of the *bgp4VpnLabelBlock*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4VpnLabelBlock*.

bgp4VpnL2Site setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *bgp4VpnL2Site*.

SEE ALSO

NAME - bgp4LearnedRoute, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQuery*, *NAME - bgp4VpnL2Site*

NAME - bgp4VpnRouteRange

bgp4VpnRouteRange — configures a route range, with attributes, to be associated with an L3 VPN site.

SYNOPSIS

bgp4VpnRouteRange *subcommand options*

DESCRIPTION

The *bgp4VpnRouteRange* command holds a route range that is associated with a *NAME - bgp4VpnL3Site*.This command includes all of the options and commands of the *NAME - bgp4RouteItem*command and defines a set of routes and associat ed attributes.

Refer to *bgp4VpnRouteRange* for an overview of this command. The optional BGP4 test package must be installed in order for this command to operate.

NOTE: Only the additional label related options are described for this command. Refer to *NAME - bgp4RouteItem* for the remainder of the options. Note that the following options in this command should be used in lieu of those in the *NAME - bgp4RouteItem* command: *enableversus enableRouteRange, networkIpAddressversusnetworkAddress*and to*Prefixversus thruPrefix*.

STANDARD OPTIONS

distinguisherAsNumber

If *distinguisherType* is set to *bgp4DistinguisherTypeAS*, this is the 2-byte AS number in the administrator sub-field of the value field of the VPN Route Distinguisher. It is the global part of the route distinguisher. *(default = 0)*

distinguisherAsNumberStep

The increment step for for the distinguisher AS number.

distinguisherAsNumberStepAcrossVrfs

he increment step for per VRF distinguisher AS number within the VRF Range.

distinguisherAssigned Number

The assigned number subfield of the value field of the VPN route distinguisher. It is a number from a numbering space which is maintained by the enterprise administers for a given IP address or ASN space. It is the local part of the route distinguisher. *(default = 0)*

distinguisherAssignedNumberStep

The increment step for for the distinguisher assigned number.

distinguisherAssignedNumberStepAcrossVrfs

The increment step for per VRF distinguisher assigned number within the VRF Range.

distinguisherCount

The number of times that the increment step will be used. *(default = 1)*

distinguisherIpAddress

If *distinguisherType* is set to *bgp4DistinguisherTypeIP*, this is the 4-byte IP address in the administrator subfield of the value field of the VPN Route Distinguisher. It is the global part of the route distinguisher. *(default = 0.0.0.0)*

distinguisherIpAddressStep

The increment step for for the distinguisher IP address.

distinguisherIpAddressStepAcrossVrfs

The increment step for per VRF distinguisher IP address within the VRF Range.

routeStepAcrossVrfs

The route step across VRFs.

distinguisherCountPerVrf

The number of times that the increment step is used per VRF.

distinguisherMode

Specifies which part of the route distinguisher you want to increment. One of:

Option	Value	Usage
vpnDistinguisher IncrementGlobalPart	0	the administrative part of the route distinguisher.
vpnDistinguisher IncrementLocalPart	1	*(default)* the assigned part of the route distinguisher

distinguisherStep

The size of the increment step to be used with the part of the route distinguisher which will be incremented. *(default = 1)*

distinguisherType

Indicates the type of administrator field used in route distinguisher that will be included in the route announcements. One of:

Option	Value	Usage
bgp4Distinguisher TypeAS	0	*(default)* The Administrator subfield is 2 bytes in length, and contains an Autonomous System number (ASN).
bgp4Distinguisher TypeIP	1	The Administrator subfield is 4 bytes in length, and contains an IP address
bgp4Distinguisher TypeAS4Byte		The Administrator subfield is 4 bytes in length, and contains an Autonomous System number (ASN), which is 4 bytes

Option	Value	Usage
		in length.

enable *true* | *false*

Enables the use of this route range as an advertised range. *(default = false)*

labelEnd

The first label value available in the label space (range). *(default = 1,046,400)*

labelMode

The mode for assigning labels to routes. One of:

Option	Value	Usage
bgp4VpnFixedLabel	0	the same label value is used for all packets.
bgp4VpnIncrementLabel	1	*(default)* the label value is incremented by the *labelStep* value for each successive packet

labelSpaceId

The label space to which the label associated with advertised routes is associated. *(default = 0)*

labelStart

The first label value available in the label space (range). *(default = 16)*

labelStep

For Increment label mode. The increment step used when assigning successive label values. *(default = 1)*

networkIpAddress

The network address used for the generated prefixes. *(default = 0.0.0.0)*

toPrefix

The last prefix length to generate based on the *networkIpAddress* and *numRanges.(default = 24)*

enableAdvertiseNextHopAsV4

This option is exposed an option to advertise v4 next-hop address as v4 in case of IPv6 route range. It should be enabled if Route Range type is IPv6 and Enable NextHop is selected. An IPv4 next address should be set, either manually or by same as local IP as IPv4.

COMMANDS

The **bgp4VpnRouteRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4VpnRouteRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *bgp4VpnRouteRange* command.

bgp4VpnRouteRange config *option value*

Modify the configuration options of the *bgp4VpnRouteRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *bgp4VpnRouteRange*.

bgp4VpnRouteRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *NAME - bgp4VpnTarget*.

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQueryNAME - bgp4VpnL3Site*, *NAME - bgp4VpnTarget*

NAME - bgp4VpnTarget

bgp4VpnTarget — configures a target attribute to be associated with advertised L3 VPN route ranges.

SYNOPSIS

bgp4VpnTarget *subcommand options*

DESCRIPTION

The *bgp4VpnTarget* command holds information about a target attribute to be associated with VPN route ranges advertised by an L3 site. VPN targets are added to a *NAME - bgp4VpnL3Site* using the *NAME - bgp4VpnL3Site addVpnTarget* subcommand.

It also holds information on import targets used to filter received routes by an L3 site.Import targets are added to a *NAME - bgp4VpnL3Site* using the *NAME - bgp4VpnL3SiteaddImportTarget* subcommand.

The optional BGP4 test package must be installed in order for this command to operate.

STANDARD OPTIONS

asNumber

If *type* is set to *bgp4TargetTypeAS*, this is the 2-byte AS number in the administrator subfield of the value field of the target. It is the global part of the target. *(default = 0)*

assignedNumber

The assigned number subfield of the value field of the target. It is a number from a numbering space, which is maintained by the enterprise administers for a given IP address or ASN space. It is the local part of the target. *(default = 0)*

ipAddress

If *type* is set to *bgp4TargetTypeIP*, this is the 4-byte IP address in the administrator subfield of the value field of the target. It is the global part of the target. *(default = 0.0.0.0)*

type

Indicates the type of administrator field used in the target that will be included in route announcements. One of:

Option	Value	Usage
bgp4TargetTypeAS	0	*(default)* The Administrator subfield is 2 bytes in length, and contains an Autonomous System number (ASN).
bgp4TargetTypeIP	1	The Administrator subfield is 4 bytes in length, and contains an IP address

COMMANDS

The **bgp4VpnTarget** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4VpnTarget cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **bgp4VpnTarget** command.

bgp4VpnTarget config *option value*

Modify the configuration options of the bgp4VpnTarget. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bgp4VpnTarget.

bgp4VpnTarget setDefault

Sets default values for all configuration options.

EXAMPLES

```
package req IxTclHal

set hostname loopback

set username user

set card 4

set port 1

set streamId 1

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {

ixPuts $::ixErrorInfo

return 1

}

# Get the chassis ID to use in port lists

set chassis [ixGetChassisID $host]

set portList [list [list $chassis $card $port]]
```

```
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
# Set up the interface table for the port
interfaceTable select $chassis $card $port
interfaceTable clearAllInterfaces
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress {192.168.18.1}
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress {192.168.18.2}
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable false
interfaceEntry config -description {1 - 04:01
ProtocolInterface}
interfaceEntry config -macAddress {00 00 00 93 BE 34}
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 0
interfaceTable addInterface
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
# Select the port that the following commands will work with
bgp4Server select $chassis $card $port
bgp4Server clearAllNeighbors
# Set up an AS path for 100, 200, 300 to be used in a VPN route
rante
bgp4AsPathItem setDefault
```

```
bgp4AsPathItem config -enableAsSegment true

bgp4AsPathItem config -asList {100 200 300 }

bgp4AsPathItem config -asSegmentType bgpSegmentAsSet

# Add the path item into the VPN route range

bgp4VpnRouteRange addASPathItem

# Set remaining VPN route range characteristics

bgp4VpnRouteRange setDefault

bgp4VpnRouteRange config -enable true

bgp4VpnRouteRange config -labelSpaceId 0

bgp4VpnRouteRange config -labelStart 42

bgp4VpnRouteRange config -labelEnd 44

bgp4VpnRouteRange config -labelStep 1

bgp4VpnRouteRange config -labelMode bgp4VpnIncrementLabel

bgp4VpnRouteRange config -distinguisherType

bgp4DistinguisherTypeAS

bgp4VpnRouteRange config -distinguisherAsNumber 1234

bgp4VpnRouteRange config -distinguisherCount 1

bgp4VpnRouteRange config -distinguisherStep 1

bgp4VpnRouteRange config -distinguisherMode

vpnDistinguisherIncrementLocalPart

bgp4VpnRouteRange config -distinguisherAssignedNumber 1

bgp4VpnRouteRange config -networkIpAddress {10.1.0.0}

bgp4VpnRouteRange config -ipType addressTypeIpV4

bgp4VpnRouteRange config -fromPrefix 16

bgp4VpnRouteRange config -toPrefix 16

bgp4VpnRouteRange config -numRoutes 16

bgp4VpnRouteRange config -fromPacking 0

bgp4VpnRouteRange config -thruPacking 0

bgp4VpnRouteRange config -enableRouteFlap true

bgp4VpnRouteRange config -routeFlapTime 30

bgp4VpnRouteRange config -routeFlapDropTime 20

bgp4VpnRouteRange config -enableNextHop true

bgp4VpnRouteRange config -nextHopIpAddress {10.3.0.0}

bgp4VpnRouteRange config -nextHopMode 1
```

```
bgp4VpnRouteRange config -enableOrigin true
bgp4VpnRouteRange config -originProtocol bgpOriginIGP
bgp4VpnRouteRange config -enableLocalPref false
bgp4VpnRouteRange config -localPref 10
bgp4VpnRouteRange config -enableASPath true
bgp4VpnRouteRange config -iterationStep 1
# Add the route range into the VPN site
bgp4VpnL3Site addVpnRouteRange vpnRouteRange1
bgp4VpnRouteRange clearASPathList
# Set up the VPN target
bgp4VpnTarget setDefault
bgp4VpnTarget config -type bgp4TargetTypeAS
bgp4VpnTarget config -asNumber 1234
bgp4VpnTarget config -assignedNumber 1
# Add the VPN target into the VPN site
bgp4VpnL3Site addVpnTarget
bgp4VpnL3Site setDefault
bgp4VpnL3Site config -enable true
# And add the site into the BGP4 internal neighbor
bgp4Neighbor addL3Site l3Site1
# Repeat for the second site
bgp4VpnL3Site clearAllVpnTargets
bgp4VpnRouteRange setDefault
bgp4VpnRouteRange config -enable true
bgp4VpnRouteRange config -labelSpaceId 0
bgp4VpnRouteRange config -labelStart 50
bgp4VpnRouteRange config -labelEnd 60
bgp4VpnRouteRange config -labelStep 1
bgp4VpnRouteRange config -labelMode
bgp4VpnIncrementLabel
bgp4VpnRouteRange config -distinguisherType
bgp4DistinguisherTypeAS
bgp4VpnRouteRange config -distinguisherAsNumber 1234
bgp4VpnRouteRange config -distinguisherCount 1
```

```
bgp4VpnRouteRange config -distinguisherStep 1

bgp4VpnRouteRange config -distinguisherMode

vpnDistinguisherIncrementLocalPart

bgp4VpnRouteRange config -distinguisherAssignedNumber 2

bgp4VpnRouteRange config -networkIpAddress {10.2.0.0}

bgp4VpnRouteRange config -ipType addressTypeIpV4

bgp4VpnRouteRange config -fromPrefix 16

bgp4VpnRouteRange config -toPrefix 16

bgp4VpnRouteRange config -numRoutes 32

bgp4VpnRouteRange config -fromPacking 0

bgp4VpnRouteRange config -thruPacking 0

bgp4VpnRouteRange config -enableRouteFlap false

bgp4VpnRouteRange config -routeFlapTime 0

bgp4VpnRouteRange config -routeFlapDropTime 0

bgp4VpnRouteRange config -enableNextHop true

bgp4VpnRouteRange config -nextHopIpAddress {0.0.0.0}

bgp4VpnRouteRange config -nextHopMode 1

bgp4VpnRouteRange config -enableOrigin true

bgp4VpnRouteRange config -originProtocol bgpOriginIGP

bgp4VpnRouteRange config -enableLocalPref true

bgp4VpnRouteRange config -localPref 5

bgp4VpnRouteRange config -enableASPath true

bgp4VpnRouteRange config -iterationStep 2

bgp4VpnL3Site addVpnRouteRange vpnRouteRange1

bgp4VpnRouteRange clearASPathList

bgp4VpnTarget setDefault

bgp4VpnTarget config -type bgp4TargetTypeAS

bgp4VpnTarget config -ipAddress {0.0.0.0}

bgp4VpnTarget config -asNumber 1234

bgp4VpnTarget config -assignedNumber 2

bgp4VpnL3Site addVpnTarget

bgp4VpnL3Site setDefault

bgp4VpnL3Site config -enable true

bgp4Neighbor addL3Site l3Site2
```

```
bgp4VpnL3Site clearAllVpnTargets

bgp4Neighbor setDefault

bgp4Neighbor config -type bgp4NeighborInternal

bgp4Neighbor config -enable true

bgp4Neighbor config -localIpAddress {192.168.18.2}

bgp4Neighbor config -rangeCount 1

bgp4Neighbor config -dutIpAddress {192.18.1.1}

bgp4Neighbor config -ipType addressTypeIpV4

bgp4Neighbor config -holdTimer 90

bgp4Neighbor config -updateInterval 0

bgp4Neighbor config -externalNeighborASNum 0

bgp4Neighbor config -asNumMode bgp4AsNumModeFixed

bgp4Neighbor config -numUpdatesPerIteration 1

bgp4Server addNeighbor neighbor1

protocolServer setDefault

protocolServer config -enableArpResponse true

protocolServer config -enableBgp4Service true

protocolServer set $chassis $card $port

ixWritePortsToHardware portList

# Let go of the ports that we reserved

ixClearOwnership $portList

# Disconnect from the chassis we're using

ixDisconnectFromChassis $host

# If we're running on UNIX, disconnect from the TCL Server

if [isUNIX] {

ixDisconnectTclServer $host

}
```

SEE ALSO

NAME - bgp4AsPathItem, *NAME - bgp4LearnedRoute*, *NAME - bgp4NeighborNAME - bgp4RouteFilter*, *NAME - bgp4RouteItem*, *NAME - bgp4Server*, *NAME - bgp4StatsQueryNAME - bgp4VpnL3Site*, *NAME - bgp4VpnRouteRange*

NAME - cfmBridge

cfmBridge — configures a simulated CFM Bridge.

SYNOPSIS

cfmBridge *subcommand options*

DESCRIPTION

The *cfmBridge* command holds the information related to a single simulated CFM Bridge. Interfaces are added into the cfmBridge interface list using the cfmBridge *addInterface* command. Refer to *CFM* for an overview.

The following lists are maintained for each bridge:

- Interfaces — a list of CFM interfaces associated with this bridge.
- MD Levels — a list of Maintenance Domain Levels.
- MPs — a list of Maintenance Points.
- Links — a list of links between CFM Maintenance Points.
- VLANs — a list of associated VLANs.
- Trunks — CFM Point-to-Point Trunks, used with Provider Backbone Bridging Traffic Engineering (PBB-TE).
- CCM Learned Info — Information learned from the CFM Continuity Check Messages (CCMs).
- LB Learned Info — Information learned from the CFM Loopback (LB)/Ping messages.
- LT Learned Info — Information learned from the CFM Link Trace (LT) messages.
- LT Learned Hop — Link Trace Learned Hop information.
- ITU (Delay) Learned Info — Learned information concerning frame delays. (Applies to: Y.1731 (ITU) and PBB-TE.)
- PBB-TE CCM Learned Info — Information learned from the PBB-TE CCMs.
- PBB-TE LB Learned Info — Information learned from the PBB-TE Loopback (LB)/Ping messages.

STANDARD OPTIONS

aisInterval

The type of interval between AIS messages. One of:

Option	Value	Usage
oneSec	0	One second (default).
oneMin	1	One minute.

bridgeId

The 6-octet MAC Address for this CFM bridge.

enableAis true |false

If set to *True*, enables the use of Alarm Indication Signal (AIS) messages. *(Default = disabled)*

enabled true |false

If set to *True*, enables the use of this emulated CFM Bridge.

encapsulation

The encapsulation type of the bridge. One of:

Option	Value	Usage
ethernet	0	Ethernet encapsulation.
llcSnap	1	LLC Snap encapsulation.

etherType

The EtherType for the CFM Bridge. (Integer)

function

The CFM function for the bridge. One of:

Option	Value	Usage
faultManagement	0	Configuration Fault Management (per IEEE 802.1ag).
performanceMeasurement	1	Available for ITU y.1731.

name

(Read-only) Indicates the name of the bridge which will be used as a unique key to retrieve the object. (String)

operationMode

The operation mode of the CFM bridge. One of:

Option	Value	Usage
cfm	0	(Default) Per IEEE 802.1ag.
y1731	1	Per ITU Y.1731.
pbbTe	2	Provider BackBone - Traffic Engineering.

User Input Required for Learned Info

userBvlan

For use with PBB-TE. User selection for the B-VLAN to filter on. One of:

Option	Value	Usage
noVlanId	0	No VLAN ID to filter on.

Option	Value	Usage
vlanId	1	Filter on a VLAN ID.
allVlanId	2	Filter on all VLAN IDs.

userBvlanId

For use with PBB-TE. User entry for the B-VLAN identifier to filter on. (Integer)

userBvlanPriority

For use with PBB-TE. User selection for the B-VLAN priority to filter on. Range: 0 through 7. *(default = 0)*

userBvlanTpId

For use with PBB-TE. User entry for the B-VLAN Tag Protocol ID to filter on. (Integer)

userCvlan

User selection for the C-VLAN to filter on. One of:

Option	Value	Usage
noVlanId	0	No VLAN ID to filter on.
vlanId	1	Filter on a VLAN ID.
allVlanId	2	Filter on all VLAN IDs.

userCvlanId

User entry for the C-VLAN identifier to filter on. (Integer)

userCvlanPriority

User selection for the C-VLAN priority to filter on. Range: 0 through 7. *(Default = 0)*

userCvlanTpId

User entry for the C-VLAN Tag Protocol ID to filter on. (Integer)

userDelayMethod

The type of delay mesurement method to filter on. One of:

- 1-oneWay
- 0-twoWay

userDelayType

User selection for the Delay Type to filter on. One of:

Option	Value	Usage
dm	0	Delay Measurement
dvm	1	Delay Variation Measurement

userDstMacAddress

The 6-octet MAC Address for the Destination MAC Address filter.

userDstMepId

Value for the Destination MEP ID filter. (Integer)

userLearnedInfoTimeOut

The interval, in milliseconds (ms), for the learned record to timeout. (Integer) *(default = 5000)*

userMdLevel

The Maintenance Domain (MD) Level to filter on. One of:

Option	Value	Usage
zeroMd	0	MD Level 0.
oneMd	1	MD Level 1.
twoMd	2	MD Level 2.
threeMd	3	MD Level 3.
fourMd	4	MD Level 4.
fiveMd	5	MD Level 5.
sixMd	6	MD Level 6.
sevenMd	7	MD Level 7.
allMd	8	All MD Levels.

userPbbTeDelayMethod

The type of delay mesurement method to filter on. One of:

- 1- oneWay
- 0- twoWay

userPbbTeDelayType

User selection for the PBB-TE Delay Type to filter on. One of:

Option	Value	Usage
dm		Delay Measurement
dvm		Delay Variation Measurement

userSelectDstMepById true / false

If set to *True*, enables the Select by ID option for using the Destination MEP ID to filter on, instead of the Destination MEP MAC Address.

userSelectSrcMepById true / false

If set to *True*, enables the Select by ID option for using the Source MEP ID to filter on, instead of the Source MEP MAC Address.

userSendType

User selection for the type of loopback. One of:

Option	Value	Usage
unicast	0	Only Unicast is available for CFM.
multicast	1	Available for Y.1731 only.

mepId

The MEP identifier of the CCM message.

mipId

The MIP identifier.

mpType

Sets the MP type. Possible values include:

Option	Value	Usage
mip	1	Maintenance Intermediate Point.
mep	5	Maintenance Point.

macAddress

The MAC address of the MP.

cciInterval

Sets the Continuity Check Interval (CCI). Possible values include:

- 3.33msec
- 10msec
- 100msec
- 1sec
- 10sec
- 1min
- 10min

userShortMaName

The Short MA Name to filter on. (String)

userShortMaNameFormat

User selection for the Short MA Name Format to filter on. One of:

Option	Value	Usage
allFormats	0	Filters on all Short MA Name formats.
primaryVid	1	Uses the Primary VLAN ID as the name to filter on.

Option	Value	Usage
characterString	2	Uses a simple character string to filter on.
twoOctetInteger	3	Uses a two-octet integer as the base name to filter on.
ref2685VpnId	4	Uses the VPN ID as the name to filter on.

rdi

The Remote Defect Identification. Possible values include:

- auto
- on
- off

userSrcMacAddress

The 6-octet MAC Address for the Source MAC Address filter.

userSrcMepId

Value for the Source MEP ID filter. (Integer)

userSvlan

User selection for the S-VLAN to filter on. One of:

Option	Value	Usage
noVlanId	0	No VLAN ID to filter on.
vlanId	1	Filter on a VLAN ID.
allVlanId	2	Filter on all VLAN IDs.

userSvlanId

User entry for the C-VLAN identifier to filter on. Integer.

userSvlanPriority

User selection for the C-VLAN priority to filter on. Range: 0 through 7. *(Default = 0)*

userSvlanTpId

User entry for the S-VLAN Tag Protocol ID to filter on. (Integer)

userTransactionId

User entry for the transaction identifier, if the configured MEP is not found. (Integer) *(Default = 1)*

userTtlInterval

The Time To Live value in the LTM, in hops. (Integer) *(Default =64)*

Read-only Learned Info

delayLearnedErrorString

(Read-only) Delay Learned Error String.

isDelayLearnedConfigMep true / false

(Read-only)True means that a configured MEP was found.

isDelayLearnedPacketSent true / false

(Read-only)True means that a packet was sent.

isLbLearnedConfigMep true / false

(Read-only)True means that a configured MEP was found.

isLbLearnedPacketSent true / false

(Read-only)True means that a packet was sent.

isLtLearnedConfigMep true / false

(Read-only)True means that a configured MEP was found.

isLtLearnedPacketSent true / false

(Read-only)True means that a packet was sent.

isPbbTeDelayLearnedConfigMep true / false

(Read-only)True means that a configured MEP was found.

isPbbTeDelayLearnedPacketSent true / false

(Read-only)True means that a packet was sent.

isPbbTeLbLearnedConfigMep true / false

(Read-only) True means that a configured MEP was found.

isPbtLbLearnedPacketSent true / false

(Read-only)True means that a packet was sent.

isPeriodicOamLearned InfoRefreshed true / false

(Read-only) True means that the periodic OAM information is up to date.

lbLearnedErrorString

(Read-only) LoopBack (LB)/Ping Learned Error String.

ltLearnedErrorString

(Read-only) Link Trace (LT) Learned Error String.

pbbTeDelayLearnedErrorString

(Read-only) PBB-TE Delay Learned Error String.

pbbTeLbLearnedErrorString

(Read-only) PBB-TE LoopBack (LB)/Ping Learned Error String.

COMMANDS

The **cfmBridge** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmBridge addInterface *interfaceName*

Adds the bridge interface described in the NAME - cfmInterface command to the list of interfaces associated with the bridge. The interface's entry in the list is given an identifier of *interfaceName*.

NOTE: If an interface is added to an existing bridge and is selected by a *get* command before calling *add*, *cfmServer write* can be called immediately without calling *setBridge* command. It will behave as *addOnFly*.

cfmBridge addCustomTlv *customTlv*

Adds a preconfigured custom TLV to the bridge, as defined in *cfmCustomTlv*.

cfmBridge delCustomTlv *customTlv*

Deletes a preconfigured custom TLV to the bridge, as defined in *cfmCustomTlv*.

cfmBridge getCustomTlv *customTlv*

Gets the specified custom TLV from the learned information for the bridge.

cfmBridge setCustomTlv *customTlv*

Edit on-the-fly "customTLV" name on the selected Bridge. If no customTLV name is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *NAME - cfmServer* write command in order to send these changes to the protocol server.

cfmBridge getFirstCustomTlv

Gets the first custom TLV from the learned information. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getNextCustomTlv

Gets the next custom TLV from the learned information. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge clearAllCustomTlvs

Clears all the configured custom TLVs from this bridge.

cfmBridge addLink *linkName*

Adds a link to this CFM Bridge.

cfmBridge addMdLevel *mdLevelName*

Adds an MD Level to this CFM Bridge.

cfmBridge addMp *mpName*

Adds an MP to this CFM Bridge.

cfmBridge addTrunk *mpName*

Adds a Trunk to this CFM Bridge.

cfmBridge addVlan *vlanName*

Adds a VLAN to this CFM Bridge.

cfmBridge clearAllInterfaces

Deletes all of the Interfaces in the Interface list for the bridge.

cfmBridge clearAllLinks

Deletes all of the Links in the Link list for the bridge.

cfmBridge clearAllMdLevels

Deletes all of the MD Levels in the MD Level list for the bridge.

cfmBridge clearAllMps

Deletes all of the MPs in the MP list for the bridge.

cfmBridge clearAllTrunks

Deletes all of the Trunks in the Trunk list for the bridge.

cfmBridge clearAllVlans

Deletes all of the VLANs in the VLAN list for the bridge.

cfmBridge config *option value*

Modify the configuration options of the cfmBridge. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for cfmBridge.

cfmBridge delInterface *[interfaceName]*

Deletes the bridge interface with an identifier of *interfaceName* from the interfaces list of a selected bridge. If no interfaceName is specified, it deletes the current one.

NOTE:*cfmServer write* can be called immediately without calling *setBridge* command. It will behave as *delOnFly*.

cfmBridge delLink *[linkName]*

Deletes the Link with an identifier of *linkName* from the Link list of a selected bridge. If no Link Name is specified, it deletes the current one.

cfmBridge delMdLevel *[mdLevelName]*

Deletes the MD Level with an identifier of *mdLevelName* from the MD Level list of a selected bridge. If no MdLevel Name is specified, it deletes the current one.

cfmBridge delMp *[mPName]*

Deletes the MP with an identifier of *mpName* from the MP list of a selected bridge. If no MP Name is specified, it deletes the current one.

cfmBridge delTrunk *[trunkName]*

Deletes the Trunk with an identifier of *trunkName* from the Trunk list of a selected bridge. If no Trunk Name is specified, it deletes the current one.

cfmBridge delVlan *[vlanName]*

Deletes the VLAN with an identifier of *vlanName* from the VLAN list for the bridge. If no vlanName is specified, it deletes the current one.

cfmBridge getCcmLearnedInfoList

Populates the CCM learned info list in the bridge. When it returns TCL_OK, it means learned info is returned. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getDelayMeasurementLearnedInfo

Gets the Delay Measurement learned info in the bridge. When it returns TCL_OK, it means learned info is returned. (Common to ITU (Y.1731) and PBB-TE.)

cfmBridge getDelayMeasurementLearnedInfoList

Populates the delay measlearned info list in the bridge. When it returns TCL_OK, it means learned info is returned. (Common to ITU (Y.1731), and PBB-TE.)

cfmBridge getFirstCcmLearnedInfo

Gets the first entry of CCM learned info from the list. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getFirstDelayMeasurementLearnedInfo

Gets the first entry of Delay Measurement learned info from the list. (Common to ITU (Y.1731), and PBB-TE.)

cfmBridge getFirstInterface

Gets the first interface in the list for the bridge*.* The results may be accessed using the *NAME - cfmInterface* command.

cfmBridge getFirstLink

Gets the first Link in the Link list for the bridge and refreshes the options.

cfmBridge getFirstLinkTraceLearnedInfo

Gets the first entry of LT learned info from the list. (Common to CFM and ITU (Y.1731).)

cfmBridge getFirstLoopbackLearnedInfo

Gets the first entry of Ping/Loopback (LB) learned info from the list. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getFirstMdLevel

Gets the first MD level from the MD Level list for the bridge and refreshes the options.

cfmBridge getFirstMp

Gets the first MP from the MP list for the bridge and refreshes the options.

cfmBridge getFirstPeriodicOamLearnedInfo

Gets the first entry of period OAM learned info from the list. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getFirstTrunk

Gets the first Trunk from the Trunk list for the bridge and refreshes the options.

cfmBridge getFirstVlan

Gets the first VLAN from the VLAN list for the bridge and refreshes the options.

cfmBridge getInterface *interfaceName*

Gets the interface's entry in the list with an identifier of *interfaceName* and refreshes the options. The bridge interface is accessed in the *NAME - cfmInterface*command.

cfmBridge getLearnedInfo

Gets learned info list on the bridge. When it returns TCL_OK, it means learned info is returned.

cfmBridge getLink *linkName*

Gets the Link entry in the list with an identifier of *linkName* and refreshes the options.

cfmBridge getLinkTraceLearnedInfoList

Populates the Link Trace learned info list in the bridge. When it returns TCL_OK, it means learned info is returned. (Common to CFM and ITU (Y.1731).)

cfmBridge getLoopbackLearnedInfoList

Populates the Ping/Loopback (LB) learned info list in the bridge. When it returns TCL_OK, it means learned info is returned. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getMdLevel *mdLevelName*

Gets the MD Level entry in the list with an identifier of *mdLevelName* and refreshes the options.

cfmBridge getMp *mpName*

Gets the MP entry in the list with an identifier of *mpName* and refreshes the options.

cfmBridge getNextCcmLearnedInfo

Gets the next entry of CCM learned info from the list. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getNextDelayMeasurementLearnedInfo

Gets the next entry of Delay Measurement learned info from the list. (Common to ITU (Y.1731), and PBB-TE.)

cfmBridge getNextInterface

Gets the next interface in the list of interfaces for the bridge and refreshes the options. The results may be accessed using the *NAME - cfmInterface* command.

cfmBridge getNextLoopbackLearnedInfo

Gets the next entry of Ping/Loopback (LB) learned info from the list. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getNextLink

Gets the next Link from the Link list for the bridge and refreshes the options.

cfmBridge getNextLinkTraceLearnedInfo

Gets the next entry of Link Trace learned info from the list. (Common to CFM and ITU (Y.1731).

cfmBridge getNextMdLevel

Gets the next MD Level from the MD Level list for the bridge and refreshes the options.

cfmBridge getNextMP

Gets the next MP from the MP list for the bridge and refreshes the options.

cfmBridge getNextPeriodicOamLearnedInfo

Gets the next entry of periodic OAM learned info from the list. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getNextTrunk

Gets the next Trunk from the Trunk list for the bridge and refreshes the options.

cfmBridge getNextVlan

Gets the next VLAN from the VLAN list for the bridge and refreshes the options.

cfmBridge getNextVlanLearnedInfo

Gets the learned info list for the next VLAN in the list. When it returns TCL_OK, it means learned info is returned.

cfmBridge getPeriodicOamLearnedInfoList

Populates the Delay Measurement learned info list in the bridge. When it returns TCL_OK, it means learned info is returned. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge getTrunk *trunkName*

Gets the Trunk entry in the list with an identifier of *trunkName,* and refreshes the options.

cfmBridge getVlan *vlanName*

Gets the VLAN entry in the list with an identifier of *vlanName,* and refreshes the options.

cfmBridge requestCcmLearnedInfo

Requests the CCM database for the respective bridge, with the provided user inputs for the learned info. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge requestPeriodicOamLearnedInfo

Requests the periodic OAM database for the respective bridge, with the provided user inputs for the learned info. (Common to CFM, ITU (Y.1731), and PBB-TE.)

cfmBridge setDefault

Sets default values for all configuration options.

cfmBridge setInterface *[interfaceName]*

Edit on-the-fly "interfaceName" on the selected Bridge. If no *interfaceName* is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *NAME - cfmServer write* command in order to send these changes to the protocol server.

cfmBridge setLink *[linkName]*

Edit on-the-fly "linkName" on the selected Bridge. If no linkName is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *NAME - cfmServer write* command in order to send these changes to the protocol server.

cfmBridge setMdLevel *[mdLevelName]*

Edit on-the-fly "mdLevelName" on the selected Bridge. If no mdLevelName is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *NAME - cfmServer write* command in order to send these changes to the protocol server.

cfmBridge setMp *[mpName]*

Edit on-the-fly "mpName" on the selected Bridge. If no mpName is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *NAME - cfmServerwrite* command in order to send these changes to the protocol server.

cfmBridge setTrunk *[trunkid]*

Edit on the fly "trunkName" on the selected Bridge. If no trunkName is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *NAME - cfmServer write* command in order to send these changes to the protocol server.

cfmBridge setVlan *[vlanid]*

Edit on the fly "vlanName" on the selected Bridge. If no vlanName is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *NAME - cfmServerwrite* command in order to send these changes to the protocol server.

cfmBridge showInterfaceNames

Returns the names of interfaces in the Interface list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.

cfmBridge showLinkNames

Returns the names of links in the Link list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.

cfmBridge showMdLevelNames

Returns the names of MD levels in the MD Level list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.

cfmBridge showMpNames

Returns the names of MPs in the MP List on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.

cfmBridge showTrunkNames

Returns the names of Trunks in the Trunk List on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.

cfmBridge showVlanNames

Returns the names of VLANs in the VLAN list on the selected bridge. Calling the *select* command is recommended before calling this command.

cfmBridge startDelayMeasurement

Starts the Delay Measurement learned information for the respective bridge, with the provided user inputs for the learned info. (Common to ITU (Y.1731) and PBB-TE.)

cfmBridge startLinkTrace

Starts the Link Trace learned information for the respective bridge, with the provided user inputs for the learned info. (Common to CFM and ITU (Y.1731).)

cfmBridge startLoopback

Starts the Ping/Loopback (LB) learned information for the respective bridge, with the provided user inputs for the learned info. (Common to CFM, ITU (Y.1731), and PBB-TE.)

EXAMPLES

See examples under *NAME - cfmServer*

SEE ALSO

NAME - cfmServer, *NAME - cfmInterface*, *NAME - cfmMdLevel*, *NAME - cfmMp*, *NAME - cfmLink*, *NAME - cfmVlan*, *NAME - cfmTrunk*, *NAME - cfmLbLearnedInfo*, *NAME - cfmLtLearnedInfo*, *NAME - cfmLtLearnedHop*, *NAME - cfmItuLearnedInfo*, *NAME - cfmPbtCcmLearnedInfo*, *NAME - cfmPbtLbLearnedInfo*

NAME - cfmCcmLearnedInfo

cfmCcmLearnedInfo — views retrieved learned CFM Continuity Check Messages (CCMs).

SYNOPSIS

cfmCcmLearnedInfo *subcommand options*

DESCRIPTION

The *cfmCcmLearnedInfo* command is used to look at the information retrieved when using the *requestCcmLearnedInfo* and *getCcmLearnedInfoList* subcommands of the *NAME - cfmBridge* command.

Refer to *cfmCcmLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

allRmepDead ***true / false***

(Read-only.) True indicates that this MEP is receiving none of the remote MEP's CCMs.

cciInterval

(Read-only.) The Continuity Check Message interval. (Integer)

cVlan

(Read-only.) The Stacked VLAN identifier. (String)

errCcmDefect true / false

(Read-only.) True indicates that the received CCM from this MEP has some incorrect value.

mdLevel

(Read-only.) The Maintenance Domain Level. (Integer)

mdName

(Read-only.) The Maintenance Domain Name. (String)

mdNameFormat

(Read-only.) The Maintenance Domain Name format. (Integer)

mepId

(Read-only.) The MEP identifier. Integer.

mepMacAddress

(Read-only.) The 6-byte MAC address for the MEP.

rdiRxCount

The number of rdi received.

rdiRxState

Indicates the state of the RDI whether it is *Receiving* or *Idle*.

rdiTxCount

The number of rdi transmitted.

rdiTxState

Indicates the state of the RDI whether it is *Transmitting* or *Idle*.

outOfSequenceCcmCount

(Read-only.) The number of Out of Sequence Continuity Check Messages. (Integer)

receivedAis true / false

(Read-only.) True indicates that an AIS message has been received from this MEP.

receivedIfaceTlvDefect true / false

(Read-only.) Indicates the interface status of the remote MEP.

receivedPortTlvDefect true / false

(Read-only.) Indicates the port status of the remote MEP.

receivedRdi true / false

(Read-only.) Indicates the status of the RDI bit in the last received CCM. *True*, if RDI = 1; *False*, if none has been received.

rmepCcmDefect true / false

(Read-only.) True indicates that no CCM is being received from this MEP.

shortMaName

(Read-only.) The Short Maintenance Association Name. (String)

shortMaNameFormat

(Read-only.) The short Maintenance Association Name format. (Integer)

someRmepDefect true / false

(Read-only.) Indicates the state of the Remote MEP State Machines. *True*, if at least one of the Remote MEP State Machines is not receiving valid CCM from its remote MEPs. *False*, if all Remote MEP State Machines are receiving valid CCMs.

sVlan

(Read-only.) The Single VLAN identifier. (String)

COMMANDS

The *cfmCcmLearnedInfo* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmCcmLearnedInfo setDefault

Sets default values for all of the options.

EXAMPLES

See examples under *NAME - cfmServer*.

SEE ALSO

NAME - cfmServer, *NAME - cfmBridge*, *NAME - cfmInterface*, *NAME - cfmMdLevel*, *NAME - cfmMp*, *NAME - cfmLink*, *NAME - cfmVlan*, *NAME - cfmTrunk*, *NAME - cfmLbLearnedInfo*, *NAME - cfmLtLearnedInfo*, *NAME - cfmLtLearnedHop*, *NAME - cfmItuLearnedInfo*, *NAME - cfmPbtCcmLearnedInfo*, *NAME - cfmPbtLbLearnedInfo*

NAME - cfmAisLearnedInfo

cfmAisLearnedInfo — views retrieved learned CFM AIS.

SYNOPSIS

cfmAisLearnedInfo *subcommand options*

DESCRIPTION

The *cfmAisLearnedInfo* command is used to look at the information retrieved when using the *requestAisLearnedInfo* and *getAisLearnedInfoList* subcommands of the *NAME - cfmBridge* command.

Refer to *cfmAisLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

sVlan

*(Read-only.)*The outer vlan in the received AIS packet.

cVlan

(Read-only.) The inner vlan in the received AIS packet.

remoteMacAddress

(Read-only.) MAC address of remote peer mep.

localMacAddress

(Read-only.) MAC address of local peer mep

txState

Indicates whether AIS is being transmitted from this local peer mep or not (string {Idle, Transmitting})

txCount

The number of Tx (integer)

rxState

Indicates whether AIS is being received from remote peer mep or not (string {Idle, Transmitting})

rxCount

The number of Rx (integer)

rxInterval

Rx interval of the received AIS from remote mep. It will be shown in string format

PBB-TE bVlan

(Read-only) The outer vlan in the received AIS packet. (String)

PBB-TE remoteMacAddress

(Read-only) MAC address of remote peer mep.

PBB-TE localMacAddress

(Read-only) MAC address of local peer mep.

PBB-TE txState

Indicates whether AIS is being transmitted from this local peer mep or not (string {Idle, Transmitting})

PBB-TE txCount

The number of Tx (integer).

PBB-TE rxState

Indicates whether AIS is being received from remote peer mep or not (string {Idle, Transmitting}).

PBB-TE rxCount

The number of Rx (integer)

PBB-TE rxInterval

Rx interval of the received AIS from remote mep. It will be shown in string format.

COMMANDS

EXAMPLES

See examples under *NAME - cfmServer*.

SEE ALSO

NAME - cfmServer, *NAME - cfmBridge*, *NAME - cfmInterface*, *NAME - cfmMdLevel*, *NAME - cfmMp*, *NAME - cfmLink*, *NAME - cfmVlan*, *NAME - cfmTrunk*, *NAME - cfmLbLearnedInfo*, *NAME - cfmLtLearnedInfo*, *NAME - cfmLtLearnedHop*, *NAME - cfmItuLearnedInfo*, *NAME - cfmPbtCcmLearnedInfo*, *NAME - cfmPbtLbLearnedInfo*

NAME - cfmLckLearnedInfo

cfmLckLearnedInfo — views retrieved learned CFM LCK.

SYNOPSIS

cfmLckLearnedInfo *subcommand options*

DESCRIPTION

The *cfmLckLearnedInfo* command is used to look at the information retrieved when using the *requestLckLearnedInfo* and *getLckLearnedInfoList* subcommands of the *cfmBridge* command.

Refer to *cfmLckLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

sVlan

*(Read-only.)*The outer vlan in the received LCK packet.

cVlan

(Read-only.) The inner vlan in the received LCK packet.

remoteMacAddress

(Read-only.) MAC address of remote peer mep.

localMacAddress

(Read-only.) MAC address of local peer mep

txState

Indicates whether LCK is being transmitted from this local peer mep or not (string {Idle, Transmitting})

txCount

The number of Tx (integer)

rxState

Indicates whether LCK is being received from remote peer mep or not (string {Idle, Transmitting})

rxCount

The number of Rx (integer)

rxInterval

Rx interval of the received LCK from remote mep. It will be shown in string format

PBB-TE bVlan

(Read-only) The outer vlan in the received LCK packet. (String)

PBB-TE remoteMacAddress

(Read-only) MAC address of remote peer mep.

PBB-TE localMacAddress

(Read-only) MAC address of local peer mep.

PBB-TE txState

Indicates whether LCK is being transmitted from this local peer mep or not (string {Idle, Transmitting})

PBB-TE txCount

The number of Tx (integer).

PBB-TE rxState

Indicates whether LCK is being received from remote peer mep or not (string {Idle, Transmitting}).

PBB-TE rxCount

The number of Rx (integer)

PBB-TE rxInterval

Rx interval of the received LCK from remote mep. It will be shown in string format.

COMMANDS

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmTstLearnedInfo

cfmTstLearnedInfo — views retrieved learned CFM TST.

SYNOPSIS

cfmTstLearnedInfo *subcommand options*

DESCRIPTION

The *cfmTstLearnedInfo* command is used to look at the information retrieved when using the *requestTstLearnedInfo* and *getTstLearnedInfoList* subcommands of the *cfmBridge* command.

Refer to *cfmTstLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

sVlan

*(Read-only.)*The outer vlan in the received TST packet.

cVlan

(Read-only.) The inner vlan in the received TST packet.

remoteMacAddress

(Read-only.) MAC address of remote peer mep.

localMacAddress

(Read-only.) MAC address of local peer mep

txState

Indicates whether TST is being transmitted from this local peer mep or not (string {Idle, Transmitting})

txCount

The number of Tx (integer)

rxCount

The number of Rx (integer)

rxOutofSequenceTSTCount)

The number of out of sequence TST (integer)

PRBSBitErrorsCount

The number of TST packets received with RPBS bit error (integer)

PBB-TE bVlan

(Read-only) The outer vlan in the received LCK packet. (String)

PBB-TE remoteMacAddress

(Read-only) MAC address of remote peer mep.

PBB-TE localMacAddress

(Read-only) MAC address of local peer mep.

PBB-TE txState

Indicates whether TST is being transmitted from this local peer mep or not (string {Idle, Transmitting})

PBB-TE txCount

The number of Tx (integer).

PBB-TE rxCount

The number of Rx (integer)

PBB-TE rxOutofSequenceTSTCount

The number of the out of sequence TST (integer).

PBB-TE PRBSBitErrorsCount

The number of TST packets received with RPBS bit error (integer)

COMMANDS

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmCustomTlv

cfmCustomTlv — configures a custom TLV for a CFM bridge.

SYNOPSIS

cfmCustomTlv *subcommand options*

DESCRIPTION

The *cfmCustomTlv* holds the information related to a single custom TLV on the simulated bridge. Custom TLVs are added into the *cfmBridge* interface list using the *cfmBridge addCustomTlv* command.

STANDARD OPTIONS

type

Sets the type value for the TLV.

length

Sets the length value for the TLV.

value

Sets the data value for the TLV.

includeInCcm true / false

If true, includes the custom TLV in Continuity Check Messages.

includeInLtm true / false

If true, includes the custom TLV in Link Trace Messages.

includeInLtr true / false

If true, includes the custom TLV in Link Trace Responses.

includeInLbm true / false

If true, includes the custom TLV in Loopback Messages.

includeInLbr true / false

If true, includes the custom TLV in Loopback Responses.

name

(Read-only.) The name of the custom TLV which will be used as a unique key to retrieve the object.

COMMANDS

The *cfmCustomTlv* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmInterface setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmItuLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmInterface

cfmInterface — configures an interface for a CFM bridge.

SYNOPSIS

cfmInterface *subcommand options*

DESCRIPTION

The *cfm Interface* holds the information related to a single interface on the simulated bridge. Interfaces are added into the *cfmBridge* interface list using the cfmBridge addInterface command. Refer to cfmInterface for an overview.

STANDARD OPTIONS

enabled true / false

If set, enables this CFM interface.

interfaceId

The local ID for the interface, unique per bridge. (This attribute references an object of *Interface*.)

name

(Read-only.) The name of the interface which will be used as a unique key to retrieve the object.

protocolInterfaceDescription

(Read-only.) The descriptive identifier of the protocol interface.

COMMANDS

The *cfmInterface* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmInterface setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmItuLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmItuLearnedInfo

cfmItuLearnedInfo — views retrieved learned Delay information. Applies to Y.1731 (ITU) and PBB-TE.

SYNOPSIS

cfmDelayLearnedInfo *subcommand options*

DESCRIPTION

The *cfmItuLearnedInfo* command is used to look at the information retrieved when using the *startDelayMeasurement* and *getDelayMeasurementLearnedInfo* subcommands of the *cfmBridge* command.

Refer to *cfmItuLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

srcMacAddress

The source MAC address of the CFM bridge.

dstMacAddress

The destination MAC address of the CFM bridge.

mdLevel

The Maintenance Domain level for the CFM bridge.

sVlan

The S-VLAN identifier of the CFM bridge.

cVlan

The C-VLAN identifier of the CFM bridge.

valueInNanoSec

(Read-only.) The measured amount of delay, in nanoseconds. (Integer)

valueInSec

(Read-only.) The measured amount of delay, in seconds. (Integer)

COMMANDS

The *cfmItuLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmItuLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmLbLearnedInfo

cfmLbLearnedInfo — views retrieved learned CFM Loopback (LB)/Ping information.

SYNOPSIS

cfmLbLearnedInfo *subcommand options*

DESCRIPTION

The *cfmLbLearnedInfo* command is used to look at the information retrieved when using the *startLoopback* and *getLoopbackLearnedInfoList* subcommands of the *cfmBridge* command.

Refer to *cfmLbLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

cVlan

(Read-only.) The Stacked VLAN Identifier. (String)

dstMacAddress

(Read-only.) The 6-octet destination MAC Address from the loopback.

mdLevel

(Read-only.) The MD/MEG Level for the loopback. (Integer)

reachability true / false

(Read-only.) Indicates the status of the Ping.

srcMacAddress

(Read-only.) The 6-octet source MAC Address from the loopback.

sVlan

(Read-only.) The Single VLAN identifier. (String)

transactionId

(Read-only.) The identifier the LBM was sent with. (Integer)

COMMANDS

The *cfmLbLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmLbLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmItuLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmLink

cfmLink — configures a Link for a CFM bridge.

SYNOPSIS

cfmLink *subcommand options*

DESCRIPTION

The *cfm Link* holds the information related to a single link on the simulated bridge. Interfaces are added into the *cfmBridge* link list using the *cfmBridge addLink* command. Refer to *cfmLink* for an overview.

STANDARD OPTIONS

enabled true / false

If set to *True*, enables this simulated Link on the bridge.

leftMpLocalId

(Read-only.) The MP ID corresponding to *mpTowardsIxia*. (String)

linkType

Determines the type of link. One of:

Option	Value	Usage
pointToPoint		A Point-to-Point link.
broadcast		A Broadcast link.

moreMps

This option is only available for broadcast links. Broadcast links can have multiple MPs. (This attribute references an object of *chmMp.*) (List of objRef of *cfmMp*)

mpOutwardsIxia

Specifies the MP(s) for this link that is outbound from the chassis:

- For the point-to-point *cfmLink*, corresponds to *cfmMp* outbound from the chassis (for the connected MP). This attribute references an object of *chmMp*. (objRef of *cfmMp*)

- For shared (broadcast) links, corresponds to *moreMps* for shared links outbound from the chassis to additional connected MPs. This attribute references a list of objects of *chmMp*. (List of objRef of *cfmMp*)

mpTowardsIxia

Specifies the Maintenance Point (MP) for this link that is facing the chassis. (This attribute references an object of *cfmMp*.)

name

(Read-only.) The name of the link which will be used as a unique key to retrieve the object.

rightMpLocalId

(Read-only.) The MP ID corresponding to *mpOutwardsIxia*. (String)

COMMANDS

The *cfmLink* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmLink setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmLtLearnedHop

cfmLtLearnedHop — views retrieved learned CFM Link Trace (LT) Learned Hop information.

SYNOPSIS

cfmLtLearnedHop *subcommand options*

DESCRIPTION

The *cfmLtLearnedHop* command is used to look at the information retrieved when using the *getLtLearnedHopList* subcommand of the *cfmLtLearnedInfo* command.

Refer to *cfmLtLearnedHop* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

replyTtl

(Read-only.) The Time to Live/TTL (number of hops) in the Link Trace Reply (LTR) message from the MEP or MIP. The TTL is one less than the TTL in the Link Trace Message (LTM) that was received by that network element. (Integer)

self true / false

(Read-only.) If set to Yes/True (1), indicates that the MAC Address of the MIP is configured on the same bridge port as the MEP transmitting the LTM, and belongs to the path to reach the target.

If set to No/False (0), indicates that the MAC address of the MIP (not configured on the same bridge port) is on the other end of the transmitting MEP. This MIP belongs to the path to reach the target and sends the LTR.

egressMac

(Read-only.) The MAC Address of the MP, if any, residing on the egress port of the bridge. The LTM responder resides on this bridge. (String)

ingressMac

(Read-only.) The MAC Address of the MP, if any, residing on the ingress port of the bridge. The LTM responder resides on this bridge. (String)

COMMANDS

The *cfmLtLearnedHop* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmLtLearnedHop setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmItuLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmLtLearnedInfo

cfmLtLearnedInfo — views retrieved learned CFM LinkTrace (LT) information.

SYNOPSIS

cfmLtLearnedInfo *subcommand options*

DESCRIPTION

The *cfmLtLearnedInfo* command is used to look at the information retrieved when using the *startLinkTrace* and *getLinkTraceLearnedInfoList* subcommands of the *cfmBridge* command. It contains one list — of learned hops. See *cfmLtLearnedHop*

Refer to *cfmLtLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

cVlan

(Read-only.) The link stacked VLAN identifier. (String)

dstMacAddress

(Read-only.) The 6-octet MEP destination MAC Address. (String)

hopCount

(Read-only.) The number of hops in the link. (Integer)

hops

(Read-only.) List of hops to reach the particular MEP (MAC Address). (String, 6-octet MAC Addresses)

mdLevel

(Read-only.) The Maintenance Domain or Maintenance Entity Group identifier for the link. (Integer)

replyStatus

(Read-only.) Indicates the status of the reply. One of:

Option	Value	Usage
completeReply		The reply was complete.
partialReply		The reply was partially complete.
noReply		No reply was received.

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the link.

sVlan

(Read-only.) The link single VLAN identifier. (String)

transactionId

(Read-only.) Identifier sent with the Link Trace Message (LTM). (Integer)

COMMANDS

The *cfmLtLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmLtLearnedInfo getFirstLtLearnedHop

Gets the first entry of Link Trace Learned Hop info from the list.

cfmLtLearnedInfo getLtLearnedHopList

Populates the Link Trace Learned Hop list. When it returns TCL_OK, it means learned info is retrieved.

cfmLtLearnedInfo getNextLtLearnedHop

Gets the next entry of Link Trace Learned Hop info from the list

cfmLtLearnedInfo setDefault

Sets default values for all configuration options.

EXAMPLE

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmItuLearnedInfo*, *cfmLtLearnedHop*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmMdLevel

cfmMdLevel — configures a Maintenance Domain (MD) Level for a CFM bridge.

SYNOPSIS

cfmMdLevel *subcommand options*

DESCRIPTION

The *cfm MdLevel* holds the information related to a single link on the simulated bridge. MD Levels are added into the *cfmBridge* MD Level list using the *cfmBridge addMdLevel* command. Refer to *cfmMdLevel* for an overview.

STANDARD OPTIONS

enabled true / false

If set to *True*, enables the simulated MD Level on the bridge.

mdLevelId

(Read-only.) The Level available on the bridge. Depending on the configuration, this can be a number from 0 to 7. (Integer)

mdName

The name of the MD, based on the selection for the *mdNameFormat* (below). (String)

mdNameFormat

The naming format for each level instance. Options include the following:

- set enumList [list cfmNoNamePresent cfmDomainNameString cfmMACAddressPlus2OctetInt cfmMANNameCharString]
- setEnumValList $enumList enumValList
- set enumsArray(cfmMdLevel,mdNameFormat) $enumValList

name

(Read-only.) The name of the MD Level which will be used as a unique key to retrieve the object.

COMMANDS

The *cfmMdLevel* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmMdLevel setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmItuLearnedInfo*, *cfmLtLearnedHop*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmMp

cfmMp — configures a Maintenance Point (MP) on a CFM bridge.

SYNOPSIS

cfmMp *subcommand options*

DESCRIPTION

The *cfm Mp* holds the information related to a single MP on the simulated bridge. MPs are added into the *cfmBridge* MP list using the *cfmBridge addMp* command. Refer to *cfmMp* for an overview.

STANDARD OPTIONS

addCcmCustomTlvs true / false

If true, adds a custom CCM TLV to messages.

addLbmCustomTlvs true / false

If true, adds a custom LBM TLV to messages.

addLbrCustomTlvs true / false

If true, adds a custom LBR TLV to messages.

addLtmCustomTlvs true / false

If true, adds a custom LTM TLV to messages.

addLtrCustomTlvs true / false

If true, adds a custom LTR TLV to messages.

addDataTlv true / false

This adds a data TLV to messages. This TLV is applicable for LBM/LBR.

addInterfaceStatusTlv true / false

If true adds an interface status TLV to messages.

We do not allow user to change value of Interface Status TLV from here. However the user can always add Interface Status TLV as an optional TLV in Bridge and edit value.

This TLV is applicable for CCM.

Default is true.

addOrganization SpecificTlv true / false

If true, adds an organization specific TLV to messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.

addPortStatusTlv true / false

If true adds an interface status TLV to messages.

We do not allow user to change value of Interface Status TLV from here. However the user can always add Interface Status TLV as an optional TLV in Bridge and edit value.

This TLV is applicable for CCM.

Default is true.

addSenderIdTlv

If true, adds a Sender TLV to PBB-TE messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.

autoDmIteration

The count for how many times DMMs will be transmitted. Default is 0 (no limit).

Min: 0

Max: 2^32

autoDmTimeout

The timeout period in seconds to wait for a response to DMMs. This value should be less than the Auto LB Timer. Default is 30.

Min: 1

Max: 65535

autoDmTimer

The time period in seconds between DMMs. Default is 60.

Min: 1

Max: 65535

autoLbIteration

The count for how many times LBM will be transmitted. Default is 0 (no limit).

Min: 0

Max: 2^32

autoLbTimeout

The timeout period in seconds to wait for a response to LBMs. This value should be less than the Auto LB Timer. Default is 30.

Min: 1

Max: 65535

autoLbTimer

The time period in seconds between LBMs. Default is 60.

Min: 1

Max: 65535

autoLtIteration

The count for how many times LTM will be transmitted. Default is 0 (no limit).

Min: 0

Max: 2^32

autoLtTimeout

The timeout period in seconds to wait for a response to LTMs. This value should be less than the Auto LT Timer. Default is 30.

Min: 1

Max: 65535

autoLtTimer

The time period in seconds between LTMs. Default is 60.

Min: 1

Max: 65535

cciInterval

The configured time between CCM transmissions. One of:

Option	Value	Usage
cci3msec	1	CCI Interval is 3 milliseconds.
cci10msec	2	CCI Interval is 10 milliseconds.
cci100msec	3	CCI Interval is 100 milliseconds.
cci1sec	4	(Default) CCI Interval is 1 second.
cci10sec	5	CCI Interval is 10 seconds.
cci1min	6	CCI Interval is 1 minute.
cci10min	7	CCI Interval is 10 minutes.

rdi

The Remote Defect Identification. Possible values include:

- auto
- on
- off

ccmPriority

Sets the priority for Continuity Check Messages. The default is 0.

Min: 0

Max: 7

chassisId

Sets the Chassis identifier. Default is 00 00 00 00 00 00.

This will take Hex value as input (0-255 byte).

chassisIdLength

Sets the length of the chassis identifier. Default is 6.

Min: 0

Max: 255.

chassisIdSubType

Sets the chassis identifier sub-type for the optional TLV messages. One of:

Option	Value	Usage
chassisComponen	1	Chassis component
interfaceAlias	2	Interface alias
portComponent	3	Port component
chassisMacAddress	4	MAC Address
networkAddress	5	Network Address
interfaceName	6	Interface name
locallyAssigned	7	Locally assigned

dataTlvLength

Sets the length of the Data TLV. Default is 4.

Min: 0

Max: 1500.

dataTlvValue

This column will take Hex value of data. This data TLV will be added both for periodic LBM and requested LBM transmit.

Default is 44 61 74 61.

dmMethod

Sets the delay mesaurement method. The available options are:

- oneWay
- twoWay

dmPriority

Sets the priority for DM Messages. This priority will be used only for periodic DMMs. The default is 0.

Min: 0

Max: 7

Note: Backward compatibility is maintained for the legacy `dmmPriority' attribute.

enabled true / false

If set to *True*, enables the simulated MP on this bridge.

enableAutoDm true / false

If true, enables the automatic sending to DM Messages.

enableAutoLb true / false

If true, enables the automatic sending to Loopback Messages.

enableAutoLt true / false

If true, enables the automatic sending of Link Trace Messages.

lbmPriority

Sets the priority for Loopback Messages. This priority will be used only for periodic LBMs. The default is 0.

Min: 0

Max: 7

ltmPriority

Sets the priority for Link Trace Messages. This priority will be used only for periodic LTMs. The default is 0.

Min: 0

Max: 7

macAddress

The 6-octet MAC Address of the MP.

managementAddress

Sets the Managment Address. Input type is HEX (0-255 byte).

Default is 01 02 03 03 04 05.

managementAddress Domain

Sets the Management Address Domain.

This will take HEX input (0-255 byte). Default is 4d 61 6e 61 67 65 6d 65 6e 74 20 41 64 64 72 20 44 6f 6d 61 69 6e ("Management Addr Domain").

managementAddress DomainLength

Sets the length of the Management Address domain. Default is 22.

Min: 0

Max: 255.

management AddressLength

Sets the length of the Managment Address.

Default is 6.

Min: 0

Max: 255.

mdLevel

The MD or MEG Level assigned to the MP. (This attribute references an object of *cfmMdLevel*.) (String)

mdLevelLocalId

(Read-only.) The MD Level Local ID. (Integer)

megId

The identifier of the Maintenance Entity Group (MEG). (For use with Y.1731.) The base of this depends on the *megIdFormat* selection (below). (String)

megIdFormat

Sets the format for the megId (for use with Y.1731). The only option is *iccBasedFormat*.

mepId

The number that is used to identify the Maintenance End Point (MEP). (Integer)

mpType

The type of Maintenance Point. One of:

Option	Value	Usage
cfmMIP	0	Maintenance Intermediate Point
cfmMEP	1	Maintenance End Point.

name

(Read-only.) The name of the MP which will be used as a unique key to retrieve the object. (String)

organization SpecificTlvLength

Sets the length for the Organizational TLV.

Default is 4.

Min: 4

Max: 1500

organization SpecificTlvValue

Sets the value for the Organizational TLV. Default is NULL.

overrideVlanPriority true / false

If true, overrides the set VLAN priority for this bridge, and uses the advanced settings instead.

shortMaName

The short name of the MA. The base of this name depends on the selection for the *shortMaNameFormat* (below). (String)

shortMaNameFormat

Sets the format of the MA Name. One of:

Option	Value	Usage
primaryVid	1	Uses the Primary VLAN ID as the name.
characterString	2	Uses a simple character string.
twoOctetInteger	3	Uses a two-octet integer as the base name.
rfc2685VpnId	4	Uses the VPN ID as the name.

ttl

Sets the Time To Live for the period OAM. Default is 64.

Min: 1

Max: 255

vlan

The VLAN assigned to the MP. (This attribute references an object of *cfmVlan*.) (String)

vlanLocalId

(Read-only) The VLAN Local ID. (Integer).

aisEnableUnicastMac

If true, enables Ais unicast MAC address.

aisInterval

Sets the Ais interval.

aisMode

Indicates the Ais configuration mode.

aisPriority

Indicates the Ais priority value.

aisUnicastMac

Indicates ihe Ais unicast MAC address.

enableAisRx

If true, enables the Ais receiver port.

enableLckRx

If true, enables the Lck receiver port.

enableTstRx

If true, enables the Tst receiver port.

lckEnableUnicastMac

If true, enables Lck unicast MAC address.

lckInterval

Sets the Lck interval.

lckMode

Indicates the Lck configuration mode.

lckPriority

Indicates the Lck priority value.

lckSupportAisGeneration

Indicates Lck support for Ais generation.

lckUnicastMac

Indicates the Lck unicast MAC address.

tstEnableUnicastMac

If true, enables Tst unicast MAC address.

tstIncrPacketLength

Increments the Tst packet size, including the padding length.

tstIncrPacketLengthStep

Increments the Tst packet size, including the padding length by step.

tstInitialPatternValue

Indicates the initial value of Tst pattern.

tstInterval

Sets the Tst interval.

tstMode

Indicates the Tst configuration mode.

tstOverwriteSequenceNumber

Overwrites the Tst sequence number.

tstPacketLength

Indicates the Tst packet size, including the padding length.

tstPatternType

Indicates the type of Tst data pattern.

tstPriority

Indicates theTst priority value.

tstSequenceNumber

(read only) Indicates the sequence number of Tst.

tstTestType

Indicates the type of Tst test.

tstUnicastMac

Indicates the Ais unicast MAC address.

COMMANDS

The *cfmMp* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmMp setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmPbtCcmLearnedInfo

cfmPbtCcmLearnedInfo — views retrieved learned CFM Provider Backbone Bridge Traffic Engineering (PBB-TE) Continuity Check Message (CCM) information.

SYNOPSIS

cfmPbteCcmLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPbtCcmLearnedInfo* command is used to look at the information retrieved for a particular trunk when using the *requestCcmLearnedInfo* and *getCcmLearnedInfoList* subcommands of the *cfmBridge* command.

Refer to *cfmPbtCcmLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

bVlan

(Read-only.) The VLAN identifier. (String)

cciInterval

(Read-only.) The Continuity Check Message interval. (Integer)

errCcmDefect true / false

(Read-only.) True indicates that the received CCM from the remote MEP has some incorrect value.

errCcmDefectCount

(Read-only.) The number of error defects in CCMs received. (Integer)

ifaceTlvDefectCount

(Read-only.) The number of Interface TLV defects received. (Integer)

mdLevel

(Read-only.) The Maintenance Domain Level. (Integer)

mdName

(Read-only.) The Maintenance Domain Name. (String)

mdNameFormat

(Read-only.) The Maintenance Domain Name format. (Integer)

outOfSequenceCcm Count

(Read-only.) The number of Out of Sequence CCMs. (Integer)

portTlvDefectCount

(Read-only.) The number of Port TLV defects received. (Integer)

receivedIfaceTlvDefect true / false

(Read-only.) Indicates the Interface status of the remote MEP.

receivedPortTlvDefect true / false

(Read-only.) Indicates the Port status of the remote MEP.

receivedRdi true / false

(Read-only.) A value indicating the state of the RDI bit in the last received CCM. *True*, if an RDI was received by the transmitting MEP, or *False*, if no RDI was received.

remoteMacAddress

(Read-only.) The 6-octet remote MAC Address of the trunk.

remoteMepDefectCount

(Read-only.) The number of Remote MEP defects received. (Integer)

remoteMepId

(Read-only.) The remote Maintenance Endpoint identifier. (Integer)

rmepCcmDefect true / false

(Read-only.) Indicates the state of the Remote MEP State Machines. *True* indicates that the Remote MEP State Machine is not receiving valid CCMs from its Remote MEP. *False* indicates that the Remote MEP State Machine is receiving valid CCMs.

srcMacAddress

(Read-only.) The 6-octet source MAC Address of the trunk.

srcMepId

(Read-only.) The source Maintenance Endpoint identifier. (Integer)

shortMaName

(Read-only.) Short Maintenance Association (MA) name. (String)

shortMaNameFormat

(Read-only.) Short Maintenance Association (MA) name format. (Integer

COMMANDS

The *cfmPbtCcmLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmPbtCcmLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*,*cfmMP*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*,, *cfmLtLearnedHop*, *cfmItuLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmPbtDmLearnedInfo

cfmPbtDmLearnedInfo — views retrieved learned CFM Provider Backbone Bridge Traffic Engineering (PBB-TE) Domain Maintenance (DM) information.

SYNOPSIS

cfmPbtDmLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPbtDmLearnedInfo* command is used to look at the Loopback (Ping) information retrieved when using the *startLoopback* and *getLoopbackLearnedInfoList* subcommands of the *cfmBridge* command.

STANDARD OPTIONS

bVlan

(Read-only.) The trunk VLAN identifier. (String)

dstMacAddress

(Read-only.) The 6-octet destination MAC Address for the trunk.

mdLevel

(Read-only.) The MD level for the trunk. (Integer.

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the trunk.

valueInNanoSec

(Read-only.) The measured amount of delay, in nanoseconds. (Integer)

valueInSec

(Read-only.) The measured amount of delay, in seconds. (Integer)

COMMANDS

The *cfmPbtDmLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmPbtDmLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*,*cfmMdLevel* *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*, *cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*

NAME - cfmPbtLbLearnedInfo

cfmPbtLbLearnedInfo — views retrieved learned CFM Provider Backbone Bridge Traffic Engineering (PBB-TE) Loopback (LB)/Ping information.

SYNOPSIS

cfmPbtLbLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPbtLbLearnedInfo* command is used to look at the Loopback (Ping) information retrieved when using the *startLoopback* and *getLoopbackLearnedInfoList* subcommands of the *cfmBridge* command.

Refer to *cfmPbtLbLearnedInfo* for an overview of this command. The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

bVlan

(Read-only.) The trunk VLAN identifier. (String)

dstMacAddress

(Read-only.) The 6-octet destination MAC Address for the trunk.

mdLevel

(Read-only.) The MD level for the trunk. (Integer.

reachability true / false

(Read-only.) Indicates the status of the Ping.

rtt

(Read-only.) The time difference between the Ping request and the Ping reply, in micro-seconds (us). (Integer)

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the trunk.

transactionId

(Read-only.) The transaction identifier the LBM was sent with. (Integer)

COMMANDS

The *cfmPbtLbLearnedInfo* command is invoked with the following subcommand. If no sub-command is specified, returns a list of all subcommands available.

cfmPbtLbLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo* *cfmPbtCcmLearnedInfo*

NAME - cfmPbtLtLearnedInfo

cfmPbtLtLearnedInfo — views retrieved learned CFM Provider Backbone Bridge Traffic Engineering (PBB-TE) Link Trace information.

SYNOPSIS

cfmPbtLtLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPbtLtLearnedInfo* command is used to look at the Link trace information retrieved.

The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

bVlan

(Read-only.) The trunk VLAN identifier. (String)

dstMacAddress

(Read-only.) The 6-octet destination MAC Address for the trunk.

hopCount

(Read-only.) The number of hops in the link.

hops

(Read-only.) List of hops to reach the particular MEP (MAC address).

mdLevel

(Read-only.) The MD level for the trunk. (Integer.

replyStatus

(Read-only.) Indicates the status of the reply.

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the trunk.

transactionId

(Read-only.) The transaction identifier the LBM was sent with. (Integer)

COMMANDS

The *cfmPbtLtLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmPbtLtLearnedInfo getLtLearnedHopList

Retrieves the learned hops for the link.

cfmPbtLtLearnedInfo getFirstLtLearnedHop

Retrieves the first hop in the link list.

cfmPbtLtLearnedInfo getNextLtLearnedHop

Retrieves the next hop in the link list.

cfmPbtLtLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo* *cfmPbtCcmLearnedInfo*

NAME - cfmPbtPeriodicOamDmLearnedInfo

cfmPbtPeriodicOamDmLearnedInfo — views retrieved learned CFM Provider Backbone Bridge Traffic Engineering (PBB-TE) Domain Maintenance information.

SYNOPSIS

cfmPbtPeriodicOamDmLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPbtPeriodicOamDmLearnedInfo* command is used to look at the Domain Maintenance information retrieved.

The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

averageDelayNanoSec

(Read-only.) Retrieves the average delay for the Period OAM DM in nanoseconds.

averageDelaySec

(Read-only.) Retrieves the average delay for the Period OAM DM in seconds

bVlan

(Read-only.) The trunk VLAN identifier. (String)

dmmCountSent

(Read-only.) Retrieves the sent DM message count.

dstMacAddress

(Read-only.) The 6-octet destination MAC Address for the trunk.

mdLevel

(Read-only.) The MD level for the trunk. (Integer.

noReplyCount

(Read-only.) The number of no replies for the Periodic OAM messages.

recentDelayNanoSec

(Read-only.) Retrieves the the most recent delay for the Period OAM DM in nanoseconds.

recentDelaySec

(Read-only.) Retrieves the most recent delay for the Period OAM DM in seconds

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the trunk.

oneDMReceivedCount

The number of 1DM packets received.

dmmCountSent

The number of DMM packets sent.

averageDelayVariationNanoSec

(Read-only) Retrieves the avrerage delay variation for the Period OAM DM in nanoseconds.

recentDelayVariationNanoSec

(Read-only) Retrieves the most recent delay variation for the Period OAM DM in nano seconds.

COMMANDS

The *cfmPbtPeriodicOamDmLearnedInfo* command is invoked with the following sub-command. If no subcommand is specified, returns a list of all subcommands available.

cfmPbtPeriodicOamDmLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo* *cfmPbtCcmLearnedInfo*

NAME - cfmPbtPeriodicOamLbLearnedInfo

cfmPbtPeriodicOamLbLearnedInfo — views retrieved learned CFM Provider Backbone Bridge Traffic Engineering (PBB-TE) Loopback (LB)/Ping information.

SYNOPSIS

cfmPbtPeriodicOamLbLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPbtPeriodicOamLbLearnedInfo* command is used to look at the Loopback (Ping) information retrieved when using the *startLoopback* and *getLoopbackLearnedInfoList* sub-commands of the *cfmBridge* command.

The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

averageRtt

(Read-only.) The average time difference between the t Ping requests and the Ping replies, in microseconds (us). (Integer)

bVlan

(Read-only.) The trunk VLAN identifier. (String)

dstMacAddress

(Read-only.) The 6-octet destination MAC Address for the trunk.

lbmSentCount

(Read-only.) The number of Loopback messages sent.

mdLevel

(Read-only.) The MD level for the trunk. (Integer.

noReplyCount

(Read-only.) The number of loopback messages that haven't been replied to.

recentRtt

(Read-only.) The time difference between the most recent Ping request and the Ping reply, in microseconds (us). (Integer)

recentReachability true / false

(Read-only.) Indicates the status of the Ping.

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the trunk.

transactionId

(Read-only.) The transaction identifier the LBM was sent with. (Integer)

COMMANDS

The *cfmPbtPeriodicOamLbLearnedInfo* command is invoked with the following sub-command. If no subcommand is specified, returns a list of all subcommands available.

cfmPbtPeriodicOamLbLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo* *cfmPbtCcmLearnedInfo*

NAME - cfmPbtPeriodicOamLtLearnedInfo

cfmPbtPeriodicOamLtLearnedInfo — views retrieved learned CFM Provider Backbone Bridge Traffic Engineering (PBB-TE) periodic OAM Link Trace information.

SYNOPSIS

cfmPbtPeriodicOamLtLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPbtPeriodicOamLtLearnedInfo* command is used to look at the periodic OAM Link trace information retrieved.

The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

averageHopCount

(Read-only.) The average hop count for the Link trace.

bVlan

(Read-only.) The trunk VLAN identifier. (String)

completeReplyCount

(Read-only.) The number of replies to send Link trace messages.

dstMacAddress

(Read-only.) The 6-octet destination MAC Address for the trunk.

ltmSentCount

(Read-only.) The number of link trace messages sent.

mdLevel

(Read-only.) The MD level for the trunk. (Integer.

noReplyCount

(Read-only.) The number of link trace messages sent that were not replied to

partialReplyCount

(Read-only.) The number of link trace messages sent that received some sort of reply.

recentHopCount

(Read-only.) The number of hops in the link.

recentHops

(Read-only.) List of hops to reach the particular MEP (MAC address).

recentReplyStatus

(Read-only.) Indicates the status of the reply.

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the trunk.

transactionId

(Read-only.) The transaction identifier the LBM was sent with. (Integer)

COMMANDS

The *cfmPbtPeriodicOamLtLearnedInfo* command is invoked with the following sub-command. If no subcommand is specified, returns a list of all subcommands available.

cfmPbtPeriodicOamLtLearnedInfo getLtLearnedHopList

Retrieves the learned hops for the link.

cfmPbtPeriodicOamLtLearnedInfo getFirstLtLearnedHop

Retrieves the first hop in the link list.

cfmPbtPeriodicOamLtLearnedInfo getNextLtLearnedHop

Retrieves the next hop in the link list.

cfmPbtPeriodicOamLtLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo*, *cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*

NAME - cfmPeriodicOamDmLearnedInfo

cfmPeriodicOamDmLearnedInfo — views retrieved learned CFM Domain Maintenance information.

SYNOPSIS

cfmPeriodicOamDmLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPeriodicOamDmLearnedInfo* command is used to look at the Domain Maintenance information retrieved.

The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

averageDelayNanoSec

(Read-only.) Retrieves the average delay for the Period OAM DM in nanoseconds.

averageDelaySec

(Read-only.) Retrieves the average delay for the Period OAM DM in seconds

cVlan

(Read-only.) The C-VLAN identifier. (String)

dmmCountSent

(Read-only.) Retrieves the sent DM message count.

dstMacAddress

(Read-only.) The 6-octet destination MAC Address.

mdLevel

(Read-only.) The MD level. (Integer.

noReplyCount

(Read-only.) The number of no replies for the Periodic OAM messages.

recentDelayNanoSec

(Read-only.) Retrieves the most recent delay for the Period OAM DM in nanoseconds

recentDelaySec

(Read-only.) Retrieves the most recent delay for the Period OAM DM in seconds

oneDMReceivedCount	The number of 1DM packets received.

dmmCountSent	The number of DMM packets sent.
averageDelayVariationNanoSec	*(Read-only)* Retrieves the avrerage delay variation for the Period OAM DM in nanoseconds.
recentDelayVariationNanoSec	*(Read-only)* Retrieves the most recent delay variation for the Period OAM DM in nano seconds

oneDMReceivedCount

The number of 1DM packets received.

dmmCountSent

The number of DMM packets sent.

averageDelayVariationNanoSec

(Read-only) Retrieves the avrerage delay variation for the Period OAM DM in nanoseconds.

recentDelayVariationNanoSec

(Read-only) Retrieves the most recent delay variation for the Period OAM DM in nano seconds.

sVlan

(Read-only.) The S-VLAN identifier. (String)

srcMacAddress

(Read-only.) The 6-octet source MAC Address.

COMMANDS

The *cfmPbtPeriodicOamDmLearnedInfo* command is invoked with the following sub-command. If no subcommand is specified, returns a list of all subcommands available.

cfmPeriodicOamDmLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo* *cfmPbtCcmLearnedInfo*

NAME - cfmPeriodicOamLbLearnedInfo

cfmPeriodicOamLbLearnedInfo — views retrieved learned CFM Loopback (LB)/Ping information.

SYNOPSIS

cfmPeriodicOamLbLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPeriodicOamLbLearnedInfo* command is used to look at the Loopback (Ping) information retrieved when using the *startLoopback* and *getLoopbackLearnedInfoList* subcommands of the *cfmBridge* command.

The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

averageRtt

(Read-only.) The average time difference between the Ping requests and the Ping replies, in microseconds (us). (Integer)

cVlan

(Read-only.) The C-VLAN identifier. (String)

dstMacAddress

(Read-only.) The 6-octet destination MAC Address.

lbmSentCount

(Read-only.) The number of Loopback messages sent.

mdLevel

(Read-only.) The MD level (Integer.)

noReplyCount

(Read-only.) The number of loopback messages that haven't been replied to.

recentRtt

(Read-only.) The time difference between the most recent Ping request and the Ping reply, in microseconds (us). (Integer)

recentReachability true / false

(Read-only.) Indicates the status of the Ping.

srcMacAddress

(Read-only.) The 6-octet source MAC Address for the trunk.

sVlan

(Read-only.) The S-VLAN identifier. (String)

transactionId

(Read-only.) The transaction identifier the LBM was sent with. (Integer)

COMMANDS

The *cfmPeriodicOamLbLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmtPeriodicOamLbLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo* *cfmPbtCcmLearnedInfo*

NAME - cfmPeriodicOamLtLearnedInfo

cfmPeriodicOamLtLearnedInfo — views retrieved learned CFM periodic OAM Link Trace information.

SYNOPSIS

cfmPeriodicOamLtLearnedInfo *subcommand options*

DESCRIPTION

The *cfmPeriodicOamLtLearnedInfo* command is used to look at the periodic OAM Link trace information retrieved.

The optional CFM test package must be installed in order for this command to operate.

STANDARD OPTIONS

averageHopCount

(Read-only.) The average hop count for the Link trace.

cVlan

(Read-only.) The C-VLAN identifier. (String)

completeReplyCount

(Read-only.) The number of replies to send Link trace messages.

dstMacAddress

(Read-only.) The 6-octet destination MAC Address.

ltmSentCount

(Read-only.) The number of link trace messages sent.

mdLevel

(Read-only.) The MD level (Integer.)

noReplyCount

(Read-only.) The number of link trace messages sent that were not replied to

partialReplyCount

(Read-only.) The number of link trace messages sent that received some sort of reply.

recentHopCount

(Read-only.) The number of hops in the link.

recentHops

(Read-only.) List of hops to reach the particular MEP (MAC address).

recentReplyStatus

(Read-only.) Indicates the status of the reply.

srcMacAddress

(Read-only.) The 6-octet source MAC Address.

sVlan

(Read-only.) The S-VLAN identifier. (String)

transactionId

(Read-only.) The transaction identifier the LBM was sent with. (Integer)

COMMANDS

The *cfmPeriodicOamLtLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

cfmPeriodicOamLtLearnedInfo getLtLearnedHopList

Retrieves the learned hops for the link.

cfmPeriodicOamLtLearnedInfo getFirstLtLearnedHop

Retrieves the first hop in the link list.

cfmPeriodicOamLtLearnedInfo getNextLtLearnedHop

Retrieves the next hop in the link list.

cfmPeriodicOamLtLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo* *cfmPbtCcmLearnedInfo*

NAME - cfmServer

cfmServer — accesses the CFM component of the protocol server for a particular port.

SYNOPSIS

cfmServer *subcommand options*

DESCRIPTION

The *cfmServer* command is necessary in order to access the CFM protocol server for a particular port. The *select* subcommand **must** be used before all other CFM commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on CFM server testing with Ixia equipment. Refer to *cfmServer* for an overview of this command.

STANDARD OPTIONS

enableOptionalTlvValidation

If set to yes (1), enables the validation of optional TLVs.

receiveCcm

If set to yes (1), enables the receipt of Continuity Check Messages (CCMs).

sendCcm

If set to yes (1), enables the transmission of Continuity Check Messages (CCMs).

COMMANDS

The **cfmServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmServer addBridge *bridgeName*

Adds a bridge to the bridge list.

cfmServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the cfmServer command.

cfmServer clearAllBridges

Clears the bridge list.

cfmServer config *option value*

Modifies the configuration options of the bfdServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for bfdServer.

cfmServer delBridge *[bridgeName]*

Adds a bridge to the bridge list.

cfmServer get

Gets the current configuration of the protocol server for the last selected port from its hardware. Call this command before calling *cfmServer* cget *option value* to get the value of the configuration option.

cfmServer getFirstBridge

Gets the first bridge from the bridge list and refreshes the options.

cfmServer getNextBridge

Gets the next bridge from the bridge list and refreshes the options.

cfmServer getBridge *bridgeName*

Gets bridgeName and refreshes the options.

cfmServer select *chasID cardID portID*

Selects the port.

cfmServer set

Sets the current configuration of the protocol server on the most recently selected port to its hardware. Call this command before calling *cfmServer* cget *option value* to get the value of the configuration option.

cfmServer setBridge *bridgeName*

Edits bridgeName on-the-fly. If no bridgeName is specified, the current one will be modified. Saves the configuration of the bridge.

cfmServer showBridgeNames

Returns names of bridges in the list on the selected port. Calling the *select* command is recommended before calling this command.

cfmServer write

Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card, and port. Before using this command, use the *cfmServer select* command to select the port.

EXAMPLE

```
ixInitialize $hostname

set chassis [chassis cget -id]

set portList [list [list $chassis $card $port]]

port setFactoryDefaults $chassis $card $port

port config -speed 1000

port config -duplex full

port config -flowControl false
port config -directedAddress "01 80 C2 00 00 01"
port config -multicastPauseAddress "01 80 C2 00 00
01"
port config -loopback false
port config -transmitMode portTxPacketStreams
port config -receiveMode [expr
$::portRxFirstTimeStamp|$::portRxModeWidePacketGroup]
port config -autonegotiate true
port config -advertise100FullDuplex true
port config -advertise100HalfDuplex true
port config -advertise10FullDuplex true
port config -advertise10HalfDuplex true
port config -advertise1000FullDuplex true
port config -portMode portEthernetMode
port config -rxTxMode gigNormal
port config -ignoreLink false
port config -advertiseAbilities portAdvertiseNone
port config -timeoutEnable true
port config -negotiateMasterSlave 1
port config -masterSlave portSlave
port config -enableSimulateCableDisconnect false
port config -enableAutoDetectInstrumentation false
port config -enableRepeatableLastRandomPattern false
port config -transmitClockDeviation 0
port config -preEmphasis preEmphasis0
port config -MacAddress "00 de bb 00 00 01"
port config -DestMacAddress "00 de bb 00 00 02"
port config -name ""
port config -numAddresses 1
port config -enableManualAutoNegotiate false
port config -enablePhyPolling true
port set $chassis $card $port
stat setDefault
stat config -mode statNormal
stat config -enableProtocolServerStats true
stat config -enableArpStats true
stat config -enablePosExtendedStats true
stat config -enableTemperatureSensorsStats true
stat config -enableAtmOamStats false
stat config -enableDhcpStats false
stat config -enableDhcpV6Stats false
stat config -includeRprPayloadFcsInCrc true
stat config -enableValidStats false
stat config -enableBgpStats false
stat config -enableIcmpStats true
stat config -enableOspfStats false
stat config -enableIsisStats false
stat config -enableRsvpStats false
stat config -enableLdpStats false
stat config -enableIgmpStats false
stat config -enableOspfV3Stats false
stat config -enablePimsmStats false
stat config -enableMldStats false
stat config -enableStpStats false
stat config -enableEigrpStats false
stat config -enableBfdStats false
stat set $chassis $card $port
packetGroup setDefault
packetGroup config -signatureOffset 48
packetGroup config -signature "08 71 18 05"
packetGroup config -insertSignature false
packetGroup config -ignoreSignature false
packetGroup config -groupId 0
packetGroup config -groupIdOffset 52
packetGroup config -enableGroupIdMask false
packetGroup config -enableInsertPgid true
packetGroup config -groupIdMask 0
packetGroup config -latencyControl cutThrough
packetGroup config -preambleSize 8
packetGroup config -sequenceNumberOffset 44
packetGroup config -sequenceErrorThreshold 2
packetGroup config -insertSequenceSignature false
packetGroup config -allocateUdf true
packetGroup config -enableSignatureMask false
packetGroup config -signatureMask "00 00 00 00"
packetGroup config -enableRxFilter false
packetGroup config -headerFilter "00 00 00 00 00
00 00 00 00 00 00 00 00 00 00
00"
packetGroup config -headerFilterMask "00 00 00 00 00
00 00 00 00 00 00 00 00 00 00
00"
packetGroup config -enable128kBinMode false
packetGroup config -enableTimeBins false
packetGroup config -numPgidPerTimeBin 32
packetGroup config -numTimeBins 10
packetGroup config -timeBinDuration 1000000
packetGroup config -enableLatencyBins false
packetGroup config -latencyBinList ""
packetGroup config -groupIdMode packetGroupCustom
packetGroup config -sequenceCheckingMode seqThreshold
packetGroup config -multiSwitchedPathMode
seqSwitchedPathPGID
packetGroup setRx $chassis $card $port
ipAddressTable setDefault
ipAddressTable config -defaultGateway "0.0.0.0"
ipAddressTable set $chassis $card $port
arpServer setDefault
arpServer config -retries 3
arpServer config -mode arpGatewayOnly
arpServer config -rate 2083333
arpServer config -requestRepeatCount 3
arpServer set $chassis $card $port
interfaceTable select $chassis $card $port
interfaceTable setDefault
interfaceTable config -dhcpV4RequestRate 0
interfaceTable config -dhcpV6RequestRate 0
interfaceTable config -dhcpV4MaximumOutstandingRequests 100
interfaceTable config -dhcpV6MaximumOutstandingRequests 100
interfaceTable set
interfaceTable clearAllInterfaces
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
dhcpV4Properties removeAllTlvs
dhcpV4Properties setDefault
dhcpV4Properties config -clientId ""
dhcpV4Properties config -serverId "0.0.0.0"
dhcpV4Properties config -vendorId ""
dhcpV4Properties config -renewTimer 0
dhcpV4Properties config -relayAgentAddress "0.0.0.0"
dhcpV4Properties config -relayDestinationAddress
"255.255.255.255"
dhcpV6Properties removeAllTlvs
dhcpV6Properties setDefault
dhcpV6Properties config -iaType
dhcpV6IaTypePermanent
dhcpV6Properties config -iaId 0
dhcpV6Properties config -renewTimer 0
dhcpV6Properties config -relayLinkAddress
"0:0:0:0:0:0:0:0"
dhcpV6Properties config -relayDestinationAddress
"FF05:0:0:0:0:0:1:3"
interfaceEntry config -enable true
interfaceEntry config -description {Connected -
ProtocolInterface - 100:01 - 3}
```

```
interfaceEntry config -macAddress {00 00 00 39 51 76}
interfaceEntry config -eui64Id {02 00 00 FF FE 39
51 76}
interfaceEntry config -atmEncapsulation
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmMode -1
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableDhcpV6 false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 1
interfaceEntry config -vlanPriority 0
interfaceTable addInterface 0
cfmServer select $chassis $card $port
cfmServer clearAllBridges
cfmInterface setDefault
cfmInterface config -enabled true
cfmInterface config -interfaceId 1
cfmInterface config -protocolInterfaceDescription
"Connected - ProtocolInterface -
100:01 - 3"
cfmBridge addInterface Interface1
cfmMdLevel setDefault
cfmMdLevel config -enabled true
cfmMdLevel config -mdLevelId 1
cfmMdLevel config -mdNameFormat cfmMANNameCharString
cfmMdLevel config -mdName "Ixiacom-0"
cfmBridge addMdLevel MDLevel1
cfmMp setDefault
cfmMp config -enabled true
cfmMp config -mepId 1
cfmMp config -mpType cfmMEP
cfmMp config -macAddress "00:00:00:00:00:01"
cfmMp config -mdLevel "MDLevel1"
cfmMp config -cciInterval cci1sec
cfmMp config -shortMaName "Ixia-0"
cfmMp config -shortMaNameFormat cfmCharacterString
cfmMp config -vlan "Unassigned"
cfmMp config -megId "Ixia - 00001"
cfmMp config -megIdFormat cfmIccBasedFormat
cfmBridge addMp MP1
cfmLink setDefault
cfmLink config -enabled true
cfmLink config -linkType cfmPointToPointLink
cfmLink config -mpOutwardsIxia "MP1"
cfmLink config -mpTowardsIxia "Ixia Port"
cfmLink config -moreMps ""
cfmBridge addLink Link1
cfmBridge setDefault
cfmBridge config -enabled true
cfmBridge config -bridgeId "00:00:c1:77:00:01"
cfmBridge config -operationMode cfm
cfmBridge config -function faultManagement
cfmBridge config -aisInterval 0
cfmBridge config -encapsulation ethernet
cfmBridge config -enableAis false
cfmBridge config -userCvlanTpId 0
cfmBridge config -userShortMaNameFormat allFormats
cfmBridge config -userSvlanId -1
cfmBridge config -userCvlanId -1
cfmBridge config -userSvlanTpId 0
cfmBridge config -userMdLevel 8
cfmBridge config -userDelayType dm
cfmBridge config -userShortMaName ""
cfmBridge config -userTtlInterval 64
cfmBridge config -userLearnedInfoTimeOut 5000
cfmBridge config -userSendType unicast
cfmBridge config -userSvlan noVlanId
cfmBridge config -userCvlan noVlanId
cfmBridge config -userSrcMacAddress
"00:00:00:00:00:01"
cfmBridge config -userDstMacAddress
"FF:FF:FF:FF:FF:FF"
cfmBridge config -userTransactionId 1
cfmBridge config -userSelectSrcMepById 0
cfmBridge config -userSrcMepId 1
cfmBridge config -userSelectDstMepById 0
cfmBridge config -userDstMepId 65535
cfmBridge config -userSvlanPriority 0
cfmBridge config -userCvlanPriority 0
cfmServer addBridge Bridge1
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse false
protocolServer config -enableIgmpQueryResponse false
protocolServer config -enableOspfService false
protocolServer config -enableBgp4Service false
protocolServer config -enableIsisService false
protocolServer config -enableRsvpService false
protocolServer config -enableRipService false
protocolServer config -enableLdpService false
protocolServer config -enableRipngService false
protocolServer config -enableMldService false
protocolServer config -enableOspfV3Service false
protocolServer config -enablePimsmService false
protocolServer config -enableStpService false
protocolServer config -enableEigrpService false
protocolServer config -enableBfdService false
protocolServer config -enableCfmService true
protocolServer config -enableBgp4CreateInterface false
protocolServer config -enableIsisCreateInterface false
protocolServer config -enableOspfCreateInterface false
protocolServer config -enableRipCreateInterface false
protocolServer config -enableRsvpCreateInterface false
protocolServer config -enableIgmpCreateInterface false
protocolServer set $chassis $card $port
flexibleTimestamp setDefault
flexibleTimestamp config -type timestampBeforeCrc
flexibleTimestamp config -offset 23
flexibleTimestamp set $chassis $card $port
capture setDefault
capture config -fullAction lock
capture config -sliceSize 8191
capture config -sliceOffset 0
capture config -trigger -1
capture config -filter -1
capture config -captureMode captureTriggerMode
capture config -continuousFilter 0
capture config -beforeTriggerFilter
captureBeforeTriggerNone
capture config -afterTriggerFilter
captureAfterTriggerFilter
capture config -triggerPosition 1.0
capture config -enableSmallPacketCapture false
capture set $chassis $card $port
filter setDefault
filter config -captureTriggerDA anyAddr
filter config -captureTriggerSA anyAddr
filter config -captureTriggerPattern anyPattern
filter config -captureTriggerError errAnyFrame
filter config -captureTriggerFrameSizeEnable false
filter config -captureTriggerFrameSizeFrom 12
filter config -captureTriggerFrameSizeTo 12
filter config -captureFilterDA anyAddr
filter config -captureFilterSA anyAddr
filter config -captureFilterPattern anyPattern
filter config -captureFilterError errAnyFrame
filter config -captureFilterFrameSizeEnable false
filter config -captureFilterFrameSizeFrom 12
filter config -captureFilterFrameSizeTo 12
filter config -userDefinedStat1DA anyAddr
filter config -userDefinedStat1SA anyAddr
filter config -userDefinedStat1Pattern anyPattern
filter config -userDefinedStat1Error errAnyFrame
filter config -userDefinedStat1FrameSizeEnable false
filter config -userDefinedStat1FrameSizeFrom 12
filter config -userDefinedStat1FrameSizeTo 12
filter config -userDefinedStat2DA anyAddr
filter config -userDefinedStat2SA anyAddr
filter config -userDefinedStat2Pattern anyPattern
filter config -userDefinedStat2Error errAnyFrame
filter config -userDefinedStat2FrameSizeEnable 0
filter config -userDefinedStat2FrameSizeFrom 12
filter config -userDefinedStat2FrameSizeTo 12
filter config -asyncTrigger1DA anyAddr
filter config -asyncTrigger1SA anyAddr
filter config -asyncTrigger1Pattern anyPattern
filter config -asyncTrigger1Error errAnyFrame
filter config -asyncTrigger1FrameSizeEnable false
filter config -asyncTrigger1FrameSizeFrom 12
filter config -asyncTrigger1FrameSizeTo 12
filter config -asyncTrigger2DA anyAddr
filter config -asyncTrigger2SA anyAddr
filter config -asyncTrigger2Pattern anyPattern
filter config -asyncTrigger2Error errAnyFrame
filter config -asyncTrigger2FrameSizeEnable false
filter config -asyncTrigger2FrameSizeFrom 12
filter config -asyncTrigger2FrameSizeTo 12
filter config -captureTriggerEnable true
filter config -captureFilterEnable true
filter config -userDefinedStat1Enable false
filter config -userDefinedStat2Enable false
filter config -asyncTrigger1Enable false
filter config -asyncTrigger2Enable false
filter set $chassis $card $port
filterPallette setDefault
filterPallette config -DA1 "00 00 00 00 00 00"
filterPallette config -DAMask1 "00 00 00 00 00 00"
filterPallette config -DA2 "00 00 00 00 00 00"
filterPallette config -DAMask2 "00 00 00 00 00 00"
filterPallette config -SA1 "00 00 00 00 00 00"
filterPallette config -SAMask1 "00 00 00 00 00 00"
filterPallette config -SA2 "00 00 00 00 00 00"
filterPallette config -SAMask2 "00 00 00 00 00 00"
filterPallette config -pattern1 "DE ED EF FE AC CA"
filterPallette config -patternMask1 "00 00 00 00 00 00"
filterPallette config -pattern2 00
filterPallette config -patternMask2 00
filterPallette config -patternOffset1 12
filterPallette config -patternOffset2 12
filterPallette config -matchType1 matchUser
filterPallette config -matchType2 matchUser
filterPallette config -patternOffsetType1
filterPalletteOffsetStartOfFrame
filterPallette config -patternOffsetType2
filterPalletteOffsetStartOfFrame
filterPallette config -gfpErrorCondition gfpErrorsOr
filterPallette config -enableGfptHecError true
filterPallette config -enableGfpeHecError true
filterPallette config -enableGfpPayloadCrcError true
filterPallette config -enableGfpBadFcsError true
filterPallette set $chassis $card $port
lappend portList [list $chassis $card $port]
```

```
ixWritePortsToHardware portList

ixCheckLinkState portList

################################################################
##

######### Generating streams for all the ports from above
#########

################################################################
##

port reset $chassis $card $port

ixWriteConfigToHardware portList -noProtocolServer
```

SEE ALSO

cfmBridge, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmTrunk

cfmTrunk — configures a Trunk for a CFM PBB-TE bridge.

SYNOPSIS

cfmTrunk *subcommand options*

DESCRIPTION

The *cfm Trunk* holds the information related to a single trunk on the simulated bridge. Trunks are added into the *cfmBridge* link list using the *cfmBridge addTrunk* command. Refer to *cfmTrunk* for an overview.

STANDARD OPTIONS

addCcmCustomTlvs true / false

If true, adds a custom CCM TLV to messages.

addLbmCustomTlvs true / false

If true, adds a custom LBM TLV to messages.

addLbrCustomTlvs true / false

If true, adds a custom LBR TLV to messages.

addLtmCustomTlvs true / false

If true, adds a custom LTM TLV to messages.

addLtrCustomTlvs true / false

If true, adds a custom LTR TLV to messages.

addDataTlv true / false

This adds a data TLV to messages. This TLV is applicable for LBM/LBR.

addInterfaceStatusTlv true / false

If true adds an interface status TLV to messages.

We do not allow user to change value of Interface Status TLV from here. However the user can always add Interface Status TLV as an optional TLV in Bridge and edit value.

This TLV is applicable for CCM.

Default is true.

addOrganization SpecificTlv true / false

If true, adds an organization specific TLV to messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.

addPortStatusTlv true / false

If true adds an interface status TLV to messages.

We do not allow user to change value of Interface Status TLV from here. However the user can always add Interface Status TLV as an optional TLV in Bridge and edit value.

This TLV is applicable for CCM.

Default is true.

addSenderIdTlv

If true, adds a Sender TLV to PBB-TE messages. This TLV is applicable for CCM, LTM/LTR, and LBM/LBR.

autoDmIteration

The count for how many times DMMs will be transmitted. Default is 0 (no limit).

Min: 0

Max: 2^32

autoDmTimeout

The timeout period in seconds to wait for a response to DMMs. This value should be less than the Auto LB Timer. Default is 30.

Min: 1

Max: 65535

autoDmTimer

The time period in seconds between DMMs. Default is 60.

Min: 1

Max: 65535

autoLbIteration

The count for how many times LBM will be transmitted. Default is 0 (no limit).

Min: 0

Max: 2^32

autoLbTimeout

The timeout period in seconds to wait for a response to LBMs. This value should be less than the Auto LB Timer. Default is 30.

Min: 1

Max: 65535

autoLbTimer

The time period in seconds between LBMs. Default is 60.

Min: 1

Max: 65535

autoLtIteration

The count for how many times LTM will be transmitted. Default is 0 (no limit).

Min: 0

Max: 2^32

autoLtTimeout

The timeout period in seconds to wait for a response to LTMs. This value should be less than the Auto LT Timer. Default is 30.

Min: 1

Max: 65535

autoLtTimer

The time period in seconds between LTMs. Default is 60.

Min: 1

Max: 65535

cciInterval

The configured time between CCM transmissions. One of:

Option	Value	Usage
cci3msec	1	CCI Interval is 3 milliseconds.
cci10msec	2	CCI Interval is 10 milliseconds.
cci100msec	3	CCI Interval is 100 milliseconds.
cci1sec	4	(Default) CCI Interval is 1 second.
cci10sec	5	CCI Interval is 10 seconds.
cci1min	6	CCI Interval is 1 minute.
cci10min	7	CCI Interval is 10 minutes.

ccmPriority

Sets the priority for Continuity Check Messages. The default is 0.

Min: 0

Max: 7

chassisId

Sets the Chassis identifier. Default is 00 00 00 00 00 00.

This will take Hex value as input (0-255 byte).

chassisIdLength

Sets the length of the chassis identifier. Default is 6.

Min: 0

Max: 255.

chassisIdSubType

Sets the chassis identifier sub-type for the optional TLV messages. One of:

Option	Value	Usage
chassisComponent	1	Chassis component
interfaceAlias	2	Interface alias
portComponent	3	Port component
chassisMacAddress	4	MAC Address
networkAddress	5	Network Address
interfaceName	6	Interface name
locallyAssigned	7	Locally assigned

dataTlvLength

Sets the length of the Data TLV. Default is 4.

Min: 0

Max: 1500.

dataTlvValue

This column will take Hex value of data. This data TLV will be added both for periodic LBM and requested LBM transmit.

Default is 44 61 74 61.

dmMethod

Sets the delay mesaurement method. The available options are:

- oneWay
- twoWay

dmPriority

Sets the priority for DM Messages. This priority will be used only for periodic DMMs. The default is 0.

- Min: 0
- Max: 7

Note: Backward compatibility is maintained for the legacy `dmmPriority' attribute.

enabled true / false

If set to *True*, enables the simulated MP on this bridge.

enableAutoDm true / false

If true, enables the automatic sending to DM Messages.

enableAutoLb true / false

If true, enables the automatic sending to Loopback Messages.

enableAutoLt true / false

If true, enables the automatic sending of Link Trace Messages.

lbmPriority

Sets the priority for Loopback Messages. This priority will be used only for periodic LBMs. The default is 0.

Min: 0

Max: 7

ltmPriority

Sets the priority for Link Trace Messages. This priority will be used only for periodic LTMs. The default is 0.

Min: 0

Max: 7

macAddress

The 6-octet MAC Address of the MP.

managementAddress

Sets the Management Address. Input type is HEX (0-255 byte).

Default is 01 02 03 03 04 05.

managementAddress Domain

Sets the Management Address Domain.

This will take HEX input (0-255 byte). Default is 4d 61 6e 61 67 65 6d 65 6e 74 20 41 64 64 72 20 44 6f 6d 61 69 6e ("Management Addr Domain").

managementAddress DomainLength

Sets the length of the Management Address domain. Default is 22.

Min: 0

Max: 255.

management AddressLength

Sets the length of the Management Address.

Default is 6.

Min: 0

Max: 255.

mdLevel

The MD or MEG Level assigned to the MP. (This attribute references an object of *cfmMdLevel*.) (String)

mdLevelLocalId

(Read-only.) The MD Level Local ID. (Integer)

megId

The identifier of the Maintenance Entity Group (MEG). (For use with Y.1731.) The base of this depends on the *megIdFormat* selection (below). (String)

megIdFormat

Sets the format for the megId (for use with Y.1731). The only option is *iccBasedFormat*.

mepId

The number that is used to identify the Maintenance End Point (MEP). (Integer)

mpType

The type of Maintenance Point. One of:

Option	Value	Usage
cfmMIP	0	Maintenance Intermediate Point
cfmMEP	1	Maintenance End Point.

rdi

The Remote Defect Identification. Possible values include:

- auto
- on
- off

name

(Read-only.) The name of the MP which will be used as a unique key to retrieve the object. (String)

organization SpecificTlvLength

Sets the length for the Organizational TLV.

Default is 4.

Min: 4

Max: 1500

organization SpecificTlvValue

Sets the value for the Organizational TLV. Default is NULL.

overrideVlanPriority true / false

If true, overrides the set VLAN priority for this bridge, and uses the advanced settings instead.

shortMaName

The short name of the MA. The base of this name depends on the selection for the *shortMaNameFormat* (below). (String)

shortMaNameFormat

Sets the format of the MA Name. One of:

Option	Value	Usage
primaryVid	1	Uses the Primary VLAN ID as the name.
characterString	2	Uses a simple character string.
twoOctetInteger	3	Uses a two-octet integer as the base name.
rfc2685VpnId	4	Uses the VPN ID as the name.

ttl

Sets the Time To Live for the period OAM. Default is 64.

Min: 1

Max: 255

vlan

The VLAN assigned to the MP. (This attribute references an object of *cfmVlan*.) (String)

vlanLocalId

(Read-only.) The VLAN Local ID. (Integer)

COMMANDS

The *cfmTrunk* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmTrunk setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmVlan*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - cfmVlan

cfmVlan — configures a VLAN for a CFM bridge.

SYNOPSIS

cfmVlan *subcommand options*

DESCRIPTION

The *cfm Vlan* holds the information related to a single VLAN on the simulated bridge. VLANs are added into the *cfmBridge* VLAN list using the *cfmBridge addVlan* command. Refer to *cfmVlan*43 for an overview.

STANDARD OPTIONS

cVlanId

A unique, 12-bit VLAN identifier which specifies the VLAN with which this frame is associated. (Integer)

cVlanPriority

The user priority of the tab: a value from 0 through 7. The use and interpretation of this field is defined in ISO/IEC 15802-3. (Integer)

cVlanTpId

The Tag Protocol ID. EtherTypes identify the protocol that follows the VLAN header. (String)

enabled true / false

If set to *True*, enables the CFM VLAN on this bridge.

name

(Read-only.) The name of the VLAN which will be used as a unique key to retrieve the object. (String)

sVlanId

A unique, 12-bit VLAN identifier which specifies the VLAN with which this frame is associated. (Integer)

sVlanPriority

The user priority of the tab: a value from 0 through 7. The use and interpretation of this field is defined in ISO/IEC 15802-3. (Integer)

sVlanTpId

The Tag Protocol ID. EtherTypes identify the protocol that follows the VLAN header. (String)

type

The VLAN type. One of:

Option	Value	Usage
singleVlan	0	Uses a single VLAN.
stackedVlan	1	Uses a stacked VLAN.

vlanLocalId

(Read-only.) The VLAN Local identifier. String.

COMMANDS

The *cfmVlan* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

cfmVlan setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *cfmServer*.

SEE ALSO

cfmServer, *cfmBridge*, *cfmInterface*, *cfmMdLevel*, *cfmMp*, *cfmLink*, *cfmTrunk*, *cfmCcmLearnedInfo*, *cfmLbLearnedInfo*, *cfmLtLearnedInfo*, *cfmLtLearnedHop*,*cfmItuLearnedInfo*, *cfmPbtCcmLearnedInfo*, *cfmPbtLbLearnedInfo*

NAME - eigrpInterface

eigrpInterface — configures an interface for an EIGRP router.

SYNOPSIS

eigrpInterface *subcommand options*

DESCRIPTION

The *eigrpInterface* holds the information related to a single interface on the simulated router. Interfaces are added into the *eigrpRouter* interface list using the *eigrpRouter addInterface* command. Refer to *eigrpInterface* overview.

STANDARD OPTIONS

bandwidth

The amount of bandwidth available on this link, in Kbps. The valid range is 1-4294967295. *(default = 10,000)*

delay

The total of delays on the path to the route/network, in microseconds. The valid range is 0 to 4294967295. *(default = 0)*

enable true / false

Enables the EIGRP interface. *(default = false)*

enableBfdRegistration true / false

Indicates if a BFD session is to be created to the EIGRP peer IP address once the EIGRP session is established. This allows EIGRP to use BFD to maintain IPv4 connectivity the EIGRP peer.

helloInterval

The time interval between Hello packets sent over the interface, in seconds. *(default = 5 seconds)*

holdTime

The amount of time starting from the reception of a HELLO from a neighbor until the moment when the neighbor is to be dropped if no further HELLO is received from it, in seconds. *(default = 15 seconds)*

InterfaceId

The local ID for the interface, unique per router.

load

The amount of load on the link. The valid range is 0 to 255. *(default = 0)*

maxTlvPerPacket

The maximum number of TLVs that will be packed into a single Update packet, taking MTU into consideration. The valid range is 0-255. A value of 0 means that maximum possible packing will be used, which depends on the MTU of the link. *(default = 30)*

name

(Read-only.) The name of the interface which will be used as a unique key to retrieve the object.

poisonedReverse true / false

Enables Poisoned Reverse. If enabled, it lets the router learn a route through a particular interface and then advertise the route through the same interface, but with an infinite metric. *(default = true)*

protocolInterfaceDescription

(Read-only.) The descriptive identifier of the protocol interface.

reliability

The reliability factor. The valid range is 0 to 255. *(default =255, which means 100% reliable)*

COMMANDS

The *eigrpInterface* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

eigrpInterface setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *eigrpServer*.

SEE ALSO

eigrpRouteRange, *eigrpRouter*, *eigrpServer*, *eigrpRouteLearnedInfo*.

NAME - eigrpRouteLearnedInfo

eigrpRouteLearnedInfo — views retrieved learned EIGRP routes.

SYNOPSIS

eigrpLearnedRoute *subcommand options*

DESCRIPTION

The *eigrpLearnedRoute* command is used to look at the routes retrieved when using the *requestLearnedRoutes* and *getLearnedRoutesList* subcommands of the *eigrpRouter* command.

Refer to *eigrpRouteLearnedInfo* for an overview of this command. The optional EIGRP test package must be installed in order for this command to operate.

STANDARD OPTIONS

destination

(Read-only.) The IPv4/IPv6 destination network that was advertised in the learned route.

FD

(Read-only.) The feasible distance; the RD added to the link cost of the interface.

hop_count

(Read-only.) The hop count for the route learned from the neighbor.

neighbor

(Read-only.) The neighbor from which the route was learned.

next_hop

(Read-only.) The IPv4/IPv6 next hop on the path to the destination network in the learned route.

prefixLength

(Read-only.) The prefix length of the route.

RD

(Read-only.) It is the reported distance of the route advertised by the neighbor. It is calculated based on bandwidth, load, delay, and reliability.

type

(Read-only.) The type of route that was learned: internal or external.

COMMANDS

The *eigrpRouteLearnedInfo* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

eigrpRouteLearnedInfo setDefault

Sets default values for all of the configuration options.

EXAMPLES

See examples under *eigrpServer*.

SEE ALSO

eigrpInterface, *eigrpRouteRange*, *eigrpRouter*, *eigrpServer*

NAME - eigrpRouter

eigrpRouter — configures an EIGRP Router.

SYNOPSIS

eigrpRouter *subcommand options*

DESCRIPTION

The *eigrpRouter* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *eigrpRouteRange* command.
- Interface — one router interface, constructed in the *eigrpInterface* command.
- Route Learned Info — routes learned by the simulated router via the *eigrpRouteLearnedInfo* command.

Routers defined in this command are added to an *eigrpServer* using the *eigrpServer addRouter* command. Refer to *eigrpRouter* for an overview of this command.

STANDARD OPTIONS

ASNumber

The identifier of the Autonomous System (AS) where this emulated EIGRP router is located. The valid range is 1 to 4294967295. *(default = 1)*

eigrpMajorVersion

The major version level of the EIGRP software. The valid range is 0 to 255. *(default = 1)*

eigrpMinorVersion

The minor version level of the EIGRP software. The valid range is 0 to 255. *(default = 2)*.

enable true / false

Enables the router. *(default = false)*

enableDiscardLearnedRoutes true /false

If enabled, the router will not store learned routes; it will discard the routes. *(default = false)*

addressFamily

Denotes IP address type, one of ipv4 or ipv6. *(default = ipv4)*

enablePiggyBack true /false

If enabled, EIGRP will piggyback an acknowledgement for the initial update with any unicast packet sent to the neighbor, instead of directly sending a separate acknowledgement packet to the neighbor. *(default = false)*

iosMajorVersion

The major version level of the referenced software. The valid range is 0 to 255. *(default = 12)*

iosMinorVersion

The major version level of the referenced software. The valid range is 0 to 255. *(default = 3)*

k1

Advanced parameter, only used in condition checking for establishing the neighbor relationship. The valid range is 0 to 255. *(default = 1)*

k2

Advanced parameter, only used in condition checking for establishing the neighbor relationship. The valid range is 0 to 255. *(default = 0)*

k3

Advanced parameter, only used in condition checking for establishing the neighbor relationship. The valid range is 0 to 255. *(default = 1)*

k4

Advanced parameter, only used in condition checking for establishing the neighbor relationship. The valid range is 0 to 255. *(default = 0)*

k5

Advanced parameter, only used in condition checking for establishing the neighbor relationship. The valid range is 0 to 255. *(default = 0)*

name

READ-ONLY. The name of the router which will be used as a unique key to retrieve the object.

routeActiveTime

It determines the maximum time (in minutes) for which a route learned from a neighbor will be active in the topology table, if the neighbor stops sending Hellos. The valid range is 1 to 4294967295. (*default = 3 minutes)*

routerId

An IPv4-formatted identifier for this emulated EIGRP router. Its default value is dependent on the card/port type.

COMMANDS

The *eigrpRouter* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

eigrpRouter addInterface *interfaceName*

Adds an interface to a router. Currently, one interface is supported per emulated EIGRP router.

NOTE: If an interface is added to an existing router and then before the router is selected by a get command again, *eigrpServer select* should be called. *eigrpServer write* can be called immediately without calling the *setRouter* command. It will behave as add-on-fly.

eigrpRouter addRouteRange *routeRangeName*

Adds a route range to the route ranges list of a router.

NOTE: If a route range is added to an existing router and then before the route is selected by a get command again, *eigrpServer select* should be called. *eigrpServer write* can be called immediately without calling the *setRouter* command. It will behave as add-on-fly.

eigrpRouter clearInterfaces

Clears the interface list on the selected router.

eigrpRouter clearAllRouteRanges

Clears the route range list on the selected router.

eigrpRouter deleteInterface *[interfaceName]*

Deletes the interface from the interface list of a selected router. If no interfaceName is specified, it deletes the current one.

NOTE:*eigrpServer write* can be called immediately, without calling the *setRouter* command. It will behave as add-on-fly.

eigrpRouter delRouteRange *[routeRangeName]*

Deletes a route range from the route ranges list of a selected router. If no routeRangeName is specified, it deletes the current one.

NOTE: *eigrpServer write* can be called immediately without calling the *setRouter* command. It will behave as add-on-fly.

eigrpRouter getFirstInterface

Gets the first interface from interface list on the selected router and refreshes the options.

eigrpRouter getFirstLearnedRoute

Gets the first learned route. It should be called after *getLearnedRouteList* is successful.

eigrpRouter getFirstRouteRange

Gets the first route range from the route range list on the selected router and refreshes the options.

eigrpRouter getInterface *interfaceName*

Gets the "interfaceName" on the selected router and refreshes the options.

eigrpRouter getLearnedRouteList

Gets the learned route list. It should be called after the *requestLearnedInfo* command.

eigrpRouter getNextLearnedRoute

Gets next learned route.

eigrpRouter getNextRouteRange

Gets the next route range from the route range list on the selected router and refreshes the options.

eigrpRouter getRouteRange

Gets the routeRangeName on the selected router and refreshes the options.

eigrpRouter populateFirstRouteRange

Fetches and populates the first route range from the list.

eigrpRouter requestLearnedInfo

Requests the route learned info.

eigrpRouter setDefault

Sets default values for all configuration options.

eigrpRouter setInterface *[interfaceName]*

Edits on the fly the interfaceName on the selected router. If no interfaceName is specified, the current one is modified.

eigrpRouter setRouteRange *[routeRangeName]*

Edits on the fly the routeRangeName on the selected router. If no routeRangeName is specified, the current one is modified.

eigrpRouter showInterfaceNames

Returns the names of interfaces in the list on the selected router.

eigrpRouter showRouteRangeNames

Returns the names of route ranges in the list on the selected router.

EXAMPLES

See examples under *eigrpServer*.

SEE ALSO

eigrpInterface, *eigrpRouteRange*, *eigrpServer*, *eigrpRouteLearnedInfo*.

NAME - eigrpRouteRange

eigrpRouteRange — sets up the parameters associated with an EIGRP route range.

SYNOPSIS

eigrpRouteRange *subcommand options*

DESCRIPTION

The *eigrpRouteRange* command describes an individual set of routes. Route ranges are added into *eigrpRouter* lists using the *eigrpRouter addRouteRange* command.

STANDARD OPTIONS

addressFamily

Denotes IP address type, one of IPv4 or IPv6. *(default = ipv4)*

arbitraryTag

(Available only for External route ranges.) An administrative tag applied to the route when it is redistributed between EIGRP and an external protocol to prevent routing loops. Used as a route mapping filter. The valid range is 0 to 4294967295. *(default = 0)*

bandwidth

The minimum amount of bandwidth available on this link, in Kbps. The valid range is 1 to 4294967295. *(default = 10,000 Kbps)*

delay

The total of delays on the path to the route/network, in microseconds. The valid range is 0 to 4294967295. *(default = 0)*

destCount

(Available only if Packing is enabled.) If packing is enabled, it indicates the maximum number of destinations that can be packed into a single Internal/External TLV. A value of 0 means that maximum possible packing will be used, which depends on the MTU of the link. The valid range is 0 to 255. (*default = 90)*

enable true / false

Enables the route range. *(default = false)*

enablePacking true /false

Enables packing of multiple destinations into a single Internal/External TLV. If disabled, only one destination will be packed into a single Internal/External TLV. *(default = true)*

firstRoute

The first route of the route range, in IPv4/IPv6 format.

(default = 0.0.0.0 for IPv4)

(default = 0:0:0:0:0:0:0:0 for IPv6)

externalMetric

(Available only for External route ranges.) The EIGRP vector metric for the cost of the path to this route/network. The valid range is 1 to 4294967295.

(default = 1)

flag

(Available only for External route ranges.) The origin of the advertised route. One of:

Option	Value	Usage
eigrpExternalRoute	1	*(default)* The route was originated outside this EIGRP AS.
eigrpCandidateDefault	2	Candidate default

hopCount

The number of hops on the way to the destination address. The valid range is 0 to 255. *(default = 0)*

load

The amount of load on the link. The valid range is 0 to 255. *(default = 0)*

maskWidth

The network mask width for the route range (in bits). The valid range is from 0 to 32 bits. *(default = 24)*

mtu

The Maximum Transmission Unit (MTU) allowed on this link, in bytes. The valid range is 0 to 16777215. *(default = 1500 bytes)*

name

(*Read-only*) The name of the interface that will be used as a unique key to retrieve the object.

nextHop

The immediate next hop IP address on the way to the destination address, in IPv4/IPv6 format.

(default = 0.0.0.0 for IPv4)

(default = 0:0:0:0:0:0:0:0 for IPv6)

noOfRoutes

The number of routes to be generated for this route range, based on the network address plus the network mask. The valid range is 1 to 16777215. *(default = 1)*

originatingAS

(Available only for External route ranges.) The external AS where this route was originated. The valid range is 1 to 4294967295. *(default = 1)*

protocolId

(Available only for External route ranges.) The external protocol where the route was originated, if applicable. One of:

Option	Value	Usage
eigrpIGRP	1	*(default)* Interior Gateway Routing Protocol
eigrpEnhancedIGRP	2	Enhanced Interior Gateway Routing Protocol (EIGRP)
eigrpStatic	3	Routes are statically configured; no routing protocol is used.
eigrpRIP	4	Routing Information Protocol
eigrpHello	5	Hello message protocol
eigrpOSPF	6	Open Shortest Path First Protocol
eigrpISIS	7	Intermediate System to Intermediate System Protocol
eigrpEGP	8	Exterior Gateway Protocol
eigrpBGP	9	Border Gateway Protocol (BGP4)
eigrpIDRP	10	Inter-Domain Routing Protocol
eigrpConnected	11	Direct connection

reliability

The reliability factor. The valid range is 0 to 255 (100% reliable). *(default = 255)*

source

(Available only for External route ranges.) The IPv4/IPv6 address for the external source of the route information, in dotted decimal format. *(default = 0.0.0.0)*

type

The type of route range: internal or external to the AS. One of:

Option	Value	Usage
eigrpExternal	1	The route range is external to the EIGRP AS.
eigrpInternal	2	*(default)* The route range is internal to the EIGRP AS.

COMMANDS

The *eigrpRouteRange* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

eigrpRouteRange setDefault

Sets default values for all of the configuration values.

EXAMPLES

See examples under *eigrpServer*.

SEE ALSO

eigrpInterface, *eigrpRouter*, *eigrpServer*, *eigrpRouteLearnedInfo*.

NAME - eigrpServer

eigrpServer — accesses the EIGRP component of the protocol server for a particular port.

SYNOPSIS

eigrpServer *subcommand options*

DESCRIPTION

The *eigrpServer* command is necessary in order to access the EIGRP component of the protocol server for a particular port. The *select* subcommand **must** be used before all other EIGRP commands. Refer to *eigrpServer* for an overview of this command. The optional EIGRP test package must be installed in order for this command to operate.

STANDARD OPTIONS

none

No standard options are defined for the *eigrpServer* command.

COMMANDS

The *eigrpServer* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

eigrpServer addRouter *routerName*

Adds a router to the router list.

eigrpServer clearAllRouters

Clears the router list.

eigrpServer delRouter *[routerName]*

Deletes a router from the router list. If no routerName is specified, it deletes the current one.

eigrpServer getFirstRouter

Gets the first router from the router list and refreshes the options.

eigrpServer getNextRouter

Gets the next router from the router list and refreshes the options.

eigrpServer getRouter *routerName*

Gets the specified router from the router list and refreshes the options.

eigrpServer select *chassis card port*

Selects the chassis, card, and port to operate on.

eigrpServer setRouter *[routerName]*

Edit the "routerName" on the fly. If no routerName is specified, the current one will be modified.

eigrpServer showRouterNames

Returns the name of routers in the list on the selected port. Calling the *select* command is recommended before calling this command.

EXAMPLES

```
package require IxTclHal

set hostname loopback

ixConnectToChassis $hostname

set chId [chassis cget -id]

set cardId 1

set portId 1

protocolServer config –enableEigrpService true

protocolServer set $chId $cardId $portId

eigrpServer select $chassis $card $port

eigrpServer clearAllRouters

eigrpInterface setDefault

eigrpInterface config -enable true

eigrpInterface config -interfaceId 3

eigrpInterface config -helloInterval 5

eigrpInterface config -holdTime 15

eigrpInterface config -poisonedReverse 1

eigrpInterface config -bandwidth 10000

eigrpInterface config -delay 0

eigrpInterface config -load 0

eigrpInterface config -reliability 255

eigrpInterface config -mtu 1500

eigrpInterface config -maxTlvPerPacket 30

eigrpInterface config -protocolInterfaceDescription

“ProtocolInterface - 01:03 - 1”

eigrpRouter addInterface Interface1

eigrpRouteRange setDefault

eigrpRouteRange config -enable
eigrpRouteRange config -firstRoute “1.1.1.1”
eigrpRouteRange config -maskWidth 24
eigrpRouteRange config -noOfRoutes 5
eigrpRouteRange config -nextHop “0.0.0.0”
eigrpRouteRange config -hopCount 1
eigrpRouteRange config -routeTag 0
eigrpRouteRange config -externalMetric 1
eigrpRouteRange config -bandwidth 10000
eigrpRouteRange config -delay 0
eigrpRouteRange config -load 0
eigrpRouteRange config -reliability 255
eigrpRouteRange config -mtu 1500
eigrpRouteRange config -source “0.0.0.0”
eigrpRouteRange config -originatingAS 1
eigrpRouteRange config -protocolId 1
eigrpRouteRange config -flag 1
eigrpRouteRange config -type 2
eigrpRouteRange config -enablePacking true
eigrpRouteRange config -destCount 90
eigrpRouter addRouteRange RouteRange1
eigrpRouteRange setDefault
eigrpRouteRange config -enable true
eigrpRouteRange config -firstRoute “5.0.0.0”
eigrpRouteRange config -maskWidth 24
eigrpRouteRange config -noOfRoutes 10000
eigrpRouteRange config -nextHop “0.0.0.0”
eigrpRouteRange config -hopCount 0
eigrpRouteRange config -arbitraryTag 0
eigrpRouteRange config -externalMetric 1
eigrpRouteRange config -bandwidth 10000
eigrpRouteRange config -delay 0
eigrpRouteRange config -load 0
eigrpRouteRange config -reliability 255
eigrpRouteRange config -mtu 1500
eigrpRouteRange config -source “0.0.0.0”
eigrpRouteRange config -originatingAS 1
eigrpRouteRange config -protocolId 1
eigrpRouteRange config -flag 1
eigrpRouteRange config -type 2
eigrpRouteRange config -enablePacking true
eigrpRouteRange config -destCount 90
eigrpRouter addRouteRange RouteRange2
eigrpRouter setDefault
eigrpRouter config -enable true
eigrpRouter config -routerId “172.26.0.1”
eigrpRouter config -ASNumber 1
eigrpRouter config -routeActiveTime 3
eigrpRouter config -enableDiscardLearnedRoutes false
eigrpRouter config -k1 1
eigrpRouter config -k2 0
eigrpRouter config -k3 1
eigrpRouter config -k4 0
eigrpRouter config -k5 0
eigrpRouter config -eigrpMajorVersion 1
eigrpRouter config -eigrpMinorVersion 2
eigrpRouter config -iosMajorVersion 12
eigrpRouter config -iosMinorVersion 3
eigrpServer addRouter Router1
eigrpServer write
# examples
# Get Router Names
eigrpServer showRouterNames
# Disable interface Interface1
eigrpServer select $ chId $ cardId $ portId
eigrpServer getRouter Router1
eigrpRouter getInterface Interface1
eigrpInterface config -enable false
eigrpRouter setInterface Interface1
```

```
eigrpServer write
# Route Ranges can also be enabled or disabled.
# Get the name of the second Router in the list
eigrpServer select $ chId $ cardId $ portId
eigrpServer getFirstRouter
eigrpServer getNextRouter
set name [eigrpRouter cget -name]
# Delete a Route Range RouteRange1
eigrpServer select $ chId $ cardId $ portId
eigrpServer getRouter Router1
eigrpRouter delRouteRange RouteRange1
eigrpServer write
# Set Router Id of Router1
eigrpServer select $ chId $ cardId $ portId
eigrpServer getRouter Router1
eigrpRouter config -routerId < some value e.g.
120.0.0.1
eigrpServer setRouter Router1
eigrpServer write
# Get the first learned route
eigrpServer select $ chId $ cardId $ portId
eigrpServer getFirstRouter
eigrpRouter requestLearnedInfo
eigrpRouter getLearnedInfoList
eigrpRouter getFirstLearnedRoute
showCmd eigrpRouteLearnedInfo
# Get the 3rd learned route
eigrpServer select $ chId $ cardId $ portId
eigrpServer getFirstRouter
eigrpRouter requestLearnedInfo
eigrpRouter getLearnedInfoList
eigrpRouter getFirstLearnedRoute
eigrpRouter getNextLearnedRoute
eigrpRouter getNextLearnedRoute
```

```
  showCmd eigrpRouteLearnedInfo
```

SEE ALSO

eigrpInterface, *eigrpRouteRange*, *eigrpRouter*, `eigrpRouteLearnedInfo`

NAME - elmiServer

elmiServer — accesses the ELMI component of the protocol server for a particular port.

SYNOPSIS

elmiServer *subcommand options*

DESCRIPTION

The *elmiServer* command is necessary in order to access the ELMI component of the protocol server for a particular port. The *select* subcommand **must** be used before all other ELMI commands.

APIs Supported

addUni

This is used to add a single UNI to the elmiServer object. Arguments: uniName (string). Return value: 0 if success, Error value if failure.

delUni

This is used to remove a particular UNI from the elmiServer object. Arguments: uniName (string). Return value: 0 if success, Error value if failure.

getUni

This is used for getting one elmiUni instance from the elmiServer. Arguments: uniName Return value: 0 if success, Error value if failure.

setUni

This is used for saving/setting a particular elmiUni instance to the elmiServer. Arguments : uniName (string). Return value: 0 if success, Error value if failure (Note: Only one elmiUni can be enabled at a time).

getFirstUni

This is used to get the first instance from the list of elmiUnis configured under elmiServer. Return value: 0 if success, Error value if failure.

getNextUni

This is used to get the next instance from the list of elmiUnis configured under elmiServer. This should be called after a call to the getFirstUni. Return value: 0 if success, Error value if failure.

clearAllUnis

This deletes all the elmiUnis configured under elmiServer. Return value: 0 if success, Error value if failure.

showUniNames

This is used to display the names of all UNIs configured under elmiServer.

SEE ALSO

eigrpRouteRange, *eigrpRouter*, *eigrpServer*, *eigrpRouteLearnedInfo*

NAME - elmiUni

elmiUni — accesses the ELMI component of the user network interface.

SYNOPSIS

elmiUni *subcommand options*

DESCRIPTION

The *elmiUNI* command is necessary in order to access the ELMI component of the user network interface. The *select* subcommand **must** be used before all other ELMI commands.

Attributes

enabled

If enabled, it signifies the protocol.

protocolInterface

It signifies the drop down list of configured protocol interface. User has to select one interface to enable configuring UNI. Until and unless protocol interface is selected user will not be able to configure and enable UNI. Default is unassigned.

Object references are:

- null
- /vport/interface ...

mode

It is a type of UNI end point. It is a drop down of ('UNI-C' and 'UNI-N'). Default is UNI-C.

Possible values include:

- uniC
- uniN

protocolVersion

This one-octet field indicates the version supported by the sending entity (UNI-C or UNI-N). Default value is ox1. Max 255, Min - 1.

overrideSendSequenceNumber

If enabled, it updates the send sequence number. This is used for negative testing. Default is false. Change of value in this field takes effect when protocol is running.

sendSequenceNumber

This one-octet field indicates the sequence number to be sent in the 'Send Sequence Number' field in transmitted packet. It will be configurable only if Override Send Sequence Number is enabled. By default it is grayed out with default value zero. Max 255, Min - 0 Change of value in this field takes effect when protocol is running.

overrideReceiveSequenceNumber

If enabled, it updates the receive sequence number. This is used for negative testing. Default is false. Change of value in this field takes effect when protocol is running.

receiveSequenceNumber

This one-octet field indicates the sequence number to be sent in the 'Receive Sequence Number' in transmitted packet. It will be configurable only if Override Receive Sequence Number is enabled. By default it is grayed out with default value zero. Max 255, Min - 0. Change of value in this field takes effect when protocol is running.

overrideDataInstance

If enabled, it updates the Data Instance field of Data Instance Information Element (IE). Default is false. Change of value in this field takes effect when protocol is running.

dataInstance

This four-octet field indicates the Data Instance value to be sent in transmitted packet. It will be configurable only if Override Data Instance is enabled. By default it is grayed out with default value 0x0 for UNI-C and 0x1 for UNI-N. Max 4 octet max value, Min 0/1. Change of value in this field takes effect when protocol is running.

pollingCounter

It signifies the full status (status of UNI and all EVCs) polling count. Range is 1-65k. Default is 360. This is applicable only for UNI-C.

statusCounter

It signifies the count of consecutive errors. Range is 2 10. Default is 4.

pollingTimer

It transmits STATUS ENQUIRY. The range is 5-30 in secondss. Default is 10 seconds. This is applicable only for UNI-C. This will be grayed out in case of UNI-N.

enablePollingVerificationTimer

If enabled, it shows the default value. This will be grayed out in case of UNI-C.

pollingVerificationTimer

It transmits STATUS. This is applicable only for UNI-N. Range is 5-30 secs. Default is 15 seconds. This will be grayed out in case of UNI-C and if 'Enable Polling Verification Timer' is false.

APIs Supported

addUniStatus

delUniStatus

getUniStatus

setUniStatus

getFirstUniStatus

getNextUniStatus

clearAllUniStatus

showUniStatusNames

addEvc

delEvc

getEvc

setEvc

getFirstEvc

getNextEvc

clearAllEvcs

showEvcNames

refreshLmiStatusLearnedInfo

getLmiStatusLearnedInfoList

getFirstLmiStatusLearnedInfo

getNextLmiStatusLearnedInfo

refreshUniStatusLearnedInfo

getUniStatusLearnedInfoList

getFirstUniStatusLearnedInfo

getNextUniStatusLearnedInfo

refreshEvcStatusLearnedInfo

getEvcStatusLearnedInfoList

getFirstEvcStatusLearnedInfo

getNextEvcStatusLearnedInfo

NAME - elmiEvc

elmiEvc — accesses the ELMI component of the ethernet virtual connection.

SYNOPSIS

elmiEvc *subcommand options*

DESCRIPTION

The *elmiEvc* command is necessary in order to access the ELMI component of the ethernet virtual connection. The *select* subcommand **must** be used before all other ELMI commands.

Attributes

enabled

If enabled, it signifies the protocol.

referenceId

It signifies the reference ID of the protocol.

evcType

It signifies the type of the EVC value.

evcId

It signifies the ID of the EVC value.

defaultEvc

evcIdentifierLength

evcStatus

untaggedPriorityTagged

APIs Supported

addBwProfile

This is used to add a single UNI to the elmiEvc object. Arguments: bwProfileName (string). Return value: 0 if success, ErrorValue if failure.

delBwProfile

This is used to remove a particular UNI from the elmiEvc object. Arguments: bwProfileName (string). Return value: 0 if success, ErrorValue if failure.

getBwProfile

This is used for getting one elmiBwProfile instance from the elmiEvc. Arguments: bwProfileName Return value: 0 if success, ErrorValue if failure.

setBwProfile

This is used for saving/setting a particular elmiBwProfile instance to the elmiEvc. Arguments : bwProfileName (string). Return value: 0 if success, ErrorValue if failure.

getFirstBwProfile

This is used to get the first instance from the list of elmiBwProfiles configured under elmiEvc. Return value: 0 if success, ErrorValue if failure.

getNextBwProfile

This is used to get the next instance from the list of elmiBwProfiles configured under elmiEvc. This should be called after a call to the getFirstBwProfile. Return value: 0 if success, ErrorValue if failure.

clearAllBwProfiles

This deletes all the elmiBwProfiles configured under elmiEvc. Return value: 0 if success, ErrorValue if failure.

showBwProfileNames

This is used to display the names of all UNIs configured under elmiEvc.

addEvcMapEntry

This is used to add a single UNI to the elmiCeVlanIdPerEvcMap object. Arguments: evcMapEntryName (string). Return value: 0 if success, ErrorValue if failure.

delEvcMapEntry

This is used to remove a particular UNI from the elmiCeVlanIdPerEvcMap object. Arguments: evcMapEntryName (string). Return value: 0 if success, ErrorValue if failure.

getEvcMapEntry

This is used for getting one elmiEvcMapEntry instance from the elmiCeVlanIdPerEvcMap. Arguments: evcMapEntryName Return value: 0 if success, ErrorValue if failure.

setEvcMapEntry

This is used for saving/setting a particular elmiEvcMapEntry instance to the elmiCeVlanIdPerEvcMap. Arguments : evcMapEntryName (string). Return value: 0 if success, ErrorValue if failure.

getFirstEvcMapEntry

This is used to get the first instance from the list of elmiEvcMapEntrys configured under elmiCeVlanIdPerEvcMap. Return value: 0 if success, ErrorValue if failure.

getNextEvcMapEntry

This is used to get the next instance from the list of elmiEvcMapEntrys configured under elmiCeVlanIdPerEvcMap. This should be called after a call to the getFirstEvcMapEntry. Return value: 0 if success, ErrorValue if failure.

clearAllEvcMapEntrys

This deletes all the elmiEvcMapEntrys configured under elmiCeVlanIdPerEvcMap. Return value: 0 if success, ErrorValue if failure.

showEvcMapEntryNames

This is used to display the names of all UNIs configured under elmiCeVlanIdPerEvcMap.

addCeVlanIdRange addEvcMapEntry

delCeVlanIdRange delEvcMapEntry

getCeVlanIdRange getEvcMapEntry

setCeVlanIdRange setEvcMapEntry

getFirstCeVlanIdRange getFirstEvcMapEntry

getNextCeVlanIdRange getNextEvcMapEntry

clearAllCeVlanIdRanges clearAllEvcMapEntrys

showCeVlanIdRanges showEvcMapEntryNames

NAME - elmiCeVlanIdRange

elmiCeVlanIdRange — This one octet field indicates the number of EVC MAP to be carried in this CE-VLAN ID/EVC MAP. Default is 1. Max 1000 Min 1

SYNOPSIS

elmiCeVlanIdRange *subcommand options*

DESCRIPTION

The *elmiCeVlanIdRange* command indicates the number of EVC MAP to be carried in this CE-VLAN ID/EVC MAP. The *select* subcommand **must** be used before all other ELMI commands.

Attributes

enabled

If enabled, it shows the EVC MAP Entry value.

startVlanId

It starts the Vlan ID. Default is 1.

incrementStep

It shows the Increment Step of Vlan ID. Default is 1.

count

It signifies the number of Vlan Ids to be carried in this EVC MAP Entry. Default is 1.

NAME - elmiBwProfile

elmiBwProfile — This one octet field indicates number of Bandwidth profile to be configured for this ELMI. Number of BW profiles for an UNI can be Maximum 1.

SYNOPSIS

elmiEvcMapEntry *subcommand options*

DESCRIPTION

The *elmiBwProfile* command indicates number of Bandwidth profile to be configured for this ELMI. The *select* subcommand **must** be used before all other ELMI commands.

Attributes

enabled

If enabled, it shows the Bandwidth profile.

cm

If enabled, Colored Mode Flag is 1. Default is false.

cf

If enabled, Coupling Flag is set to 1. Default is 0.

perCos

If enabled, Per CoS Flag shows user_priority bit values as significant and the value is set to 1. If the value is set to 0, the user_priority bit values as ignored and not processed. Default is 0.

cirMagnitude

It signifies one octet field. Default is 1.

cirMultiplier

It signifies two octet field. Default is 1.

cbsMagnitude

It signifies one octet field. Default is 1.

cbsMultiplier

It signifies one octet field. Default is 1.

eirMagnitude

It signifies one octet field. Default is 1.

eirMultiplier

It signifies two octet field. Default is 1.

ebsMagnitude

It signifies one octet field. Default is 1.

ebsMultiplier

It signifies one octet field. Default is 1.

userPriorityBits000

If enabled, Bandwidth Profile applies to frames with user_priority as 000 and the value is set to 1. Default is 0.

userPriorityBits001

If enabled, Bandwidth Profile applies to frames with user_priority as 001 and the value is set to 1. Default is 0.

userPriorityBits010

If enabled, Bandwidth Profile applies to frames with user_priority as 010 and the value is set to 1. Default is 0.

userPriorityBits011

If enabled, Bandwidth Profile applies to frames with user_priority as 011 and the value is set to 1. Default is 0.

userPriorityBits100

If enabled, Bandwidth Profile applies to frames with user_priority as 100 and the value is set to 1. Default is 0.

userPriorityBits101

If enabled, Bandwidth Profile applies to frames with user_priority as 101 and the value is set to 1. Default is 0.

userPriorityBits110

If enabled, Bandwidth Profile applies to frames with user_priority as 110 and the value is set to 1. Default is 0.

userPriorityBits111

If enabled, Bandwidth Profile applies to frames with user_priority as 111 and the value is set to 1. Default is 0.

NAME - elmiLmiStatusLearnedInfo

elmiLmiStatusLearnedInfo — It signifies an object corresponding to the General tab. It will contain one row.

SYNOPSIS

elmiLmiStatusLearnedInfo *subcommand options*

DESCRIPTION

The *elmiLmiStatusLearnedInfo* command indicates an object corresponding to the General tab. The *select* subcommand **must** be used before all other ELMI commands.

Attributes

protocolVersion

(read only) . This one-octet field indicates the version supported by the sending entity (UNI-C or UNI-N). Default value is ox1. Max 255, Min - 1.

sendSequenceNumber

(read only) This one-octet field indicates the sequence number to be sent in the 'Send Sequence Number' field in transmitted packet. It will be configurable only if Override Send Sequence Number is enabled. By default it is grayed out with default value zero. Max 255, Min - 0 Change of value in this field takes effect when protocol is running.

receiveSequenceNumber

(read only) This one-octet field indicates the sequence number to be sent in the 'Receive Sequence Number' in transmitted packet. It will be configurable only if Override Receive Sequence Number is enabled. By default it is grayed out with default value zero. Max 255, Min - 0 Change of value in this field takes effect when protocol is running.

dataInstance

(read only) This four-octet field indicates the Data Instance value to be sent in transmitted packet. It will be configurable only if Override Data Instance is enabled. By default it is grayed out with default value 0x0 for UNI-C and 0x1 for UNI-N. Max 4 octet max value, Min 0/1. Change of value in this field takes effect when protocol is running.

invalidProtocolVersion

(read only) It signifies the invalid version supported by the sending entity (UNI-C or UNI-N).

invalidEvcReferenceId

(read only) It can have more than one EVC reference Id.

invalidMsgType

(read only) It signfies the invalid message type.

outOfSequenceIe

(read only) It can have more than one IE name.

duplicatedIe

(read only) It can have more than one duplicated IE name.

mandatoryIeMissing

(read only) It can have more than one missing IE name.

invalidMandatoryIe

(read only) It can have more than one invalid mandatory IE name.

invalidNonMandatoryIe

(read only) It can have more than one invalid non-mandatory IE name.

unrecognizedIe

(read only) It can have more than one unrecognized IE name.

unexpectedIe

(read only) It can have more than one unexpected IE name.

shortMsgCounter

(read only) It signifies the short message value.

lmiStatus

(read only) It signifies the status value for the ELMI.

APIs Supported

requestGeneralLearnedInfo

This is to request the Per-UNI General Learned Info. Return value: 0 if success, ErrorValue if failure.

getGeneralLearnedInfo

This is to retrieve the General Learned Info after requestGeneralLearnedInfo has been called. Return value: 0 if success, ErrorValue if failure.

NAME - evcStatusLearnedInfo

evcStatusLearnedInfo — It signifies object corresponding to the 'EVC Status' tab. It may contain multiple rows.

SYNOPSIS

evcStatusLearnedInfo *subcommand options*

DESCRIPTION

The *evcStatusLearnedInfo* command indicates object corresponding to the 'EVC Status' tab. The *select* subcommand **must** be used before all other ELMI commands.

Attributes

referenceId

(read only) It signifies the ID of the reference value.

status

(read only) It signifies the status value of the ethernet virtual connection.

evcId

(read only) It signifies the ID of the Ethernet Virtual Connection.

evcType

(read only) It signifies the type of EVC value.

cm

(read only) If enabled, Colored Mode Flag is 1. Default is false.

cf

(read only) If enabled, Coupling Flag is set to 1. Default is 0.

perCos

(read only) If enabled, Per CoS Flag shows user_priority bit values as significant and the value is set to 1. If the value is set to 0, the user_priority bit values as ignored and not processed. Default is 0.

cirMagnitude

(read only) It signifies one octet field. Default is 1.

cirMultiplier

(read only) It signifies two octet field. Default is 1.

cbsMagnitude

(read only) It signifies one octet field. Default is 1.

cbsMultiplier

(read only) It signifies one octet field. Default is 1.

eirMagnitude

(read only) It signifies one octet field. Default is 1.

eirMultiplier

(read only) It signifies two octet field. Default is 1.

ebsMagnitude

(read only) It signifies one octet field. Default is 1.

ebsMultiplier

(read only) It signifies one octet field. Default is 1.

userPriorityBits000

(read only)If enabled, Bandwidth Profile applies to frames with user_priority as 000 and the value is set to 1. Default is 0.

userPriorityBits001

(read only) If enabled, Bandwidth Profile applies to frames with user_priority as 001 and the value is set to 1. Default is 0.

userPriorityBits010

(read only) If enabled, Bandwidth Profile applies to frames with user_priority as 010 and the value is set to 1. Default is 0.

userPriorityBits011

(read only) If enabled, Bandwidth Profile applies to frames with user_priority as 011 and the value is set to 1. Default is 0.

userPriorityBits100

(read only) If enabled, Bandwidth Profile applies to frames with user_priority as 100 and the value is set to 1. Default is 0.

userPriorityBits101

(read only) If enabled, Bandwidth Profile applies to frames with user_priority as 101 and the value is set to 1. Default is 0.

userPriorityBits110

(read only) If enabled, Bandwidth Profile applies to frames with user_priority as 110 and the value is set to 1. Default is 0.

userPriorityBits111

(read only) If enabled, Bandwidth Profile applies to frames with user_priority as 111 and the value is set to 1. Default is 0.

untaggedOrPriorityTag

(read only) It signifies the priority tag or the untagged value.

defaultEvc

(read only) It signifies the default EVC value.

vlanId

(read only) It signifies the ID of the virtual local area network.

evcIdLength

statusType

APIs Supported

requestEvcStatusLearnedInfo

This is to request the Per-UNI EVC Status Learned Info. Return value: 0 if success, ErrorValue if failure.

getEvcStatusLearnedInfo

This is to retrieve the EVC Status Learned Info after requestEvcStatusLearnedInfo has been called. Return value: 0 if success, ErrorValue if failure.

NAME - uniStatusLearnedInfo

uniStatusLearnedInfo — It signifies object corresponding to the 'UNI Status' tab. It will contain one row.

SYNOPSIS

uniStatusLearnedInfo *subcommand options*

DESCRIPTION

The *uniStatusLearnedInfo* command indicates object corresponding to the 'EVC Status' tab. The *select* subcommand **must** be used before all other ELMI commands.

Attributes

cbsMagnitude

cbsMultiplier

cf

cirMagnitude

cirMultiplier

cm

ebsMagnitude

ebsMultiplier

eirMagnitude

eirMultiplier

evcMapType

perCos

uniId

uniIdLength

userPriorityBits000

userPriorityBits001

userPriorityBits010

userPriorityBits011

userPriorityBits100

userPriorityBits101

userPriorityBits110

userPriorityBits111

NAME - igmpAddressTable

igmpAddressTable — configures the IGMP address table parameters for a port on a card on a chassis.

SYNOPSIS

igmpAddressTable *subcommand options*

DESCRIPTION

The **igmpAddressTable** command is used to configure the IGMP address table-specific information used when building IGMP address table. Entries may be added or deleted; editing of entries is accomplished by deleting the old entry and adding a new one.

STANDARD OPTIONS

none

COMMANDS

The **igmpAddressTable** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

igmpAddressTable addItem

Creates IGMP and MAC address ranges. Specific errors are:

- The configured parameters are not valid for this port.

igmpAddressTable cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **igmpAddressTable** command.

igmpAddressTable clear

Clears the IGMP address table.

igmpAddressTable config *option value*

Modify the IGMP address table configuration options of the port. If no *option* is specified, returns a list describing all of the available igmpAddressTable options (see STANDARD OPTIONS) for port.

igmpAddressTable delItem

Deletes IGMP and MAC address ranges.

igmpAddressTable get *chasID cardID portID*

Gets the current IGMP address table configuration of the port with id *portID* on card *cardID*, chassis *chasID*. Call this command before calling **igmpAddressTable** cget *option value* to get the value of the configuration option.

igmpAddressTable getFirstItem

Gets the first IGMP and MAC address range out of the IGMP address table.

igmpAddressTable getNextItem

Gets the next IGMP and MAC address range out of the IGMP address table.

igmpAddressTable set *chasID cardID portID*

Sets the IGMP address table configuration of the port with id *portID* on card *cardID*, chassis *chasID* by reading the configuration option values set by the **igmpAddressTable** config *option value* command.

igmpAddressTable setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *igmpServer*.

SEE ALSO

igmpServer

NAME - igmpAddressTableItem

igmpAddressTableItem — configures the IGMP address table parameters for a port on a card on a chassis.

SYNOPSIS

igmpAddressTableItem *subcommand options*

DESCRIPTION

The **igmpAddressTableItem** command is used to configure the IGMP address table-specific information used when building IGMP address table.

STANDARD OPTIONS

fromClientAddress

The first client address for the client address range. *(default = 0.0.0.0)*

fromGroupAddress

The first group address for the group address range. *(default = 0.0.0.0)*

numClientAddresses

Number of client consecutive addresses*. (default = 1)*

numGroupAddresses

Number of group consecutive addresses*. (default = 1)*

toClientAddress

(Read-Only.) The last client address for the client address range. *(default = 0.0.0.0)*

toGroupAddress

(Read-Only.) The last group address for the group address range. *(default = 0.0.0.0)*

COMMANDS

The **igmpAddressTableItem** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

igmpAddressTableItem cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **igmpAddressTableItem** command.

igmpAddressTableItem config *option value*

Modify the IGMP address table configuration options of the port. If no *option* is specified, returns a list describing all of the available igmpAddressTableItem options (see STANDARD OPTIONS) for port.

igmpAddressTableItem get

Gets the current IGMP address table item configuration. Call this command before calling **igmpAddressTableItem** cget *option value* to get the value of the configuration option.

igmpAddressTableItem set

Sets the IGMP address table item configuration, by reading the configuration option values set by the **igmpAddressTableItem** config *option value* command.

igmpAddressTableItem setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *igmpServer*.

SEE ALSO

igmpAddressTable

NAME - igmpGroupRange

igmpGroupRange — configures a multicast group range for a simulated IGMP host.

SYNOPSIS

igmpGroupRange *subcommand options*

DESCRIPTION

Each port's IGMP implementation includes a number of hosts, which are described in *igmpHost*. Each host is interested in any number of multicast groups, described in this command. For each multicast group range, a set of source addresses may be specified in *igmpSourceRange*. Each IGMP source range is added to the group range using the *addSourceRange* subcommand.These source ranges constitute a set of IPv4 sources that are to be included or excluded from the group range.

Refer to *IGMP (New)* for an overview.

STANDARD OPTIONS

enable true / false

Enables the use of this group range in the IGMP simulation. *(default = false)*

enablePacking true | false

If true, then *recordsPerFrame* multicast address groups are included in each transmitted listener response message. *sourcesPerRecord* source addresses are placed in each group record. By default — when packing is NOT enabled — all records will be sent in one frame. If the user wants a specified number of records to be sent in each frame, packing should be enabled (*enablePacking* is true), and the number of records indicated with the *recordsPerFrame* option. *(default = false)*

groupCount

The number of IPv4 addresses in the group range. *(default = 1)*

groupIpFrom

The starting IPv4 address for the group range. *(default = 224.0.0.0)*

incrementStep

The increment applied between IPv4 addresses in the range, if *groupCount* is more than 1. *(default = 1)*

recordsPerFrame

If *enablePacking* is true, then this is the number of multicast address groups that will be included in each transmitted listener response message (frame). If *enablePacking* is false, all records will be sent in one frame. *(default = 0)*

sourceMode

This option indicates the mode applied to the associated list of source ranges. One of:

Option	Value	Usage
multicastSourceModeInclude	0	Indicate that the source range addresses are to be included.
multicastSourceModeExclude	1	*(default)* Indicate that the source range addresses are to be excluded.

This option and all associated source ranges are ignored when IGMP version 1 or 2 is set in *igmpHost*.

sourcesPerRecord

If *enablePacking* is true, then this is the number of source addresses that will be included in each group record. *(default = 0)*

COMMANDS

The **igmpGroupRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

igmpGroupRange addSourceRange *sourceRangeId*

Adds the source range described in the *igmpSourceRange* command to the list of source ranges associated with the host. The range's entry in the list is given an identifier of *sourceRangeId*. Specific errors are:

- The parameters in igmpSourceRange are invalid
- A host with this *sourceRangeId* exists already in the list

igmpGroupRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *igmpGroupRange* command.

igmpGroupRange clearAllSourceRanges

Deletes all of the group ranges.

igmpGroupRange config *option value*

Modify the configuration options of the *igmpGroupRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for igmpGroupRange.

igmpGroupRange delSourceRange *sourceRangeId*

Deletes the group range with an identifier of *sourceRangeId*. Specific errors are:

- No host with this *sourceRangeId* exists in the list

igmpGroupRange generateStreams *chasID cardID portID [action]*

This command creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	*(default)* Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for this group range; each stream covers the set of IPv4 addresses associated with the group range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the group range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the group range; it should not be reprogrammed.

igmpGroupRange getFirstSourceRange

Access the first source range in the list. The results may be accessed using the *igmpSourceRange* command. Specific errors are:

- There are no source ranges in the list.

igmpGroupRange getNextSourceRange

Access the next source range in the list. The results may be accessed using the *igmpSourceRange* command. Specific errors are:

- *igmpGroupRange getFirstSourceRange* has not been called.
- There is no more source ranges in the list.

igmpGroupRange getSourceRange *sourceRangeId*

Accesses the source range's entry in the list with an identifier of *sourceRangeId*. The source range is accessed in the *igmpSourceRange* command. Specific errors are:

- A source range with this *sourceRangeId* does not exist in the list.

igmpGroupRange setDefault

Sets default values for all configuration options.

igmpGroupRange setSourceRange *sourceRangeId*

Sets the values for the source range's entry in the list with an identifier of *sourceRangeId* based on changes made through the *igmpSourceRange* command. This command can be used to change a running configuration and must be followed by an *igmpVxServer* *write* command in order to send these changes to the protocol server. Specific errors are:

- A source range with this *sourceRangeId* does not exist in the list.

EXAMPLES

See examples under *igmpVxServer*.

SEE ALSO

igmpHost, *igmpSourceRange*, *igmpVxServer*

NAME - igmpHost

igmpHost — configures a simulated IGMP host

SYNOPSIS

igmpHost *subcommand options*

DESCRIPTION

Each port's IGMP implementation includes a number of hosts, which are described in this command. Each host is interested in any number of multicast groups, described in *igmpGroupRange*. Each IGMP group range is added to the host using the *addGroupRange* subcommand. For each multicast group range, a set of source addresses may be specified in *igmpSourceRange*. These source ranges constitute a set of IPv4 sources that are to be included or excluded from the group range.

Refer to *IGMP (New)* for an overview.

STANDARD OPTIONS

enable true / false

Enables the use of this host in the IGMP simulation. *(default = false)*

enableGeneralQuery true | false

Enables responses to general queries received on the interface described in *protocolInterfaceDescription. (default = true)*

enableGroupSpecific true | false

Enables responses to group specific queries received on the interface described in *protocolInterfaceDescription. (default = true)*

enableImmediate Response true | false

Causes the simulated host to immediately respond to a received Query message, rather than waiting a random amount of time between 0 and the *Maximum Response Delay* field value of the Query message. *(default = false)*

enableRouterAlert *true | false*

Sets the router alert bit in transmitted listener reports. *(default = true)*

enableSupressReports true | false

If true, will cause the host to suppress the transmission of a listener report that duplicates one received on the interface. *(default = false)*

enableUnsolicited true | false

If true, will cause the host to transmit unsolicited listener reports at the interval specified in *reportFrequency*. *(default = false)*

protocolInterface Description

The *description* option associated with an *interfaceEntry* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

reportFrequency

If *enableUnsolicited* is set to true, then this is the frequency with which unsolicited listener reports will be sent, expressed in seconds. *(default = 120)*

version

The version of IGMP to be used. *(default = 2)*

COMMANDS

The **igmpHost** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

igmpHost addGroupRange *groupRangeId*

Adds the group range described in the *igmpGroupRange* command to the list of group ranges associated with the host. The range's entry in the list is given an identifier of *groupRangeId*. Specific errors are:

- The parameters in *mldGroupRange* are invalid.
- A host with this *groupRangeId* exists already in the list.

igmpHost cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *igmpHost* command.

igmpHost clearAllGroupRanges

Deletes all of the group ranges.

igmpHost config *option value*

Modify the configuration options of the *igmpHost*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for igmpHost.

igmpHost delGroupRange *groupRangeId*

Deletes the group range with an identifier of *groupRangeId*. Specific errors are:

- No host with this *groupRangeId* exists in the list.

igmpHost generateStreams *chasID cardID portID [action]*

This command creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	*(default)* Replace the port's current streams.

Option	Value	Usage
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each group range associated with the host; each stream covers the set of IPV4 addresses associated with each group range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the group range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the group range; it should not be reprogrammed.

igmpHost getFirstGroupRange

Access the first group range in the list. The results may be accessed using the *igmpGroupRange* command. Specific errors are:

- There are no group ranges in the list.

igmpHost getNextGroupRange

Access the next group range in the list*.* The results may be accessed using the *igmpGroupRange* command. Specific errors are:

- igmpHost getFirstGroupRange has not been called.
- There is no more group ranges in the list.

igmpHost getGroupRange *groupRangeId*

Accesses the group range's entry in the list with an identifier of *groupRangeId*. The group range is accessed in the *igmpGroupRange* command. Specific errors are:

- A group range with this *groupRangeId* does not exist in the list.

igmpHost setDefault

Sets default values for all configuration options.

igmpHost setGroupRange *groupRangeId*

Sets the values for the group range's entry in the list with an identifier of *groupRangeId* based on changes made through the *igmpGroupRange* command. This command can be used to change a running configuration and must be followed by an *igmpVxServer* *write* command in order to send these changes to the protocol server. Specific errors are:

- A group range with this *groupRangeId* does not exist in the list.
- Too many groups defined.

EXAMPLES

See examples under *igmpVxServer*.

SEE ALSO

igmpGroupRange, *igmpSourceRange*, *igmpVxServer*

NAME - igmpLearnedInfo

igmpLearnedInfo — views retrieved Learned IGMP information.

SYNOPSIS

igmpLearnedInfo *subcommand options*

DESCRIPTION

The *igmpLearnedInfo* command is used to look at the information retrieved when using the *requestLearnedInfo* and *getLearnedInfoList* subcommands of the *igmpQuerier* command.

Refer to *igmpLearnedInfo* for an overview of this command. The optional IGMP test package must be installed in order for this command to operate.

STANDARD OPTIONS

compatibilityMode

(Read-only.) What version of IGMP this group address is. One of:

Option	Value	Usage
IGMPV1	0	Uses IGMP version 1.
IGMPV2	1	Uses IGMP version 2.
IGMPV3	2	Uses IGMP version 3.

compatibilityTimer

(Read-only.) The number of seconds remaining in the compatibility timer. (Integer)

filterMode

(Read-only.) Whether this group address is included or excluded. One of:

Option	Value	Usage
INCLUDE	0	
EXCLUDE	1	

groupAddress

(Read-only.) The IPv4 address for the router group. (IPv4-format address)

groupTimer

(Read-only.) The number of seconds remaining in the group address timer. (Integer)

sourceAddress

(Read-only.) The IPv4 address for the group source. (IPv4-format address)

sourceTimer

(Read-only.) The number of seconds remaining in the group address timer. (Integer)

COMMANDS

The *igmpLearnedInfo* command is invoked with the following subcommand. If no sub-command is specified, returns a list of all subcommands available.

igmpLearnedInfo setDefault

Sets the options to default values.

EXAMPLES

See examples under *igmpVxServer*.

SEE ALSO

igmpGroupRange, *igmpHost*, *igmpVxServer*, *igmpQuerier*

NAME - igmpQuerier

igmpQuerier — configures an IGMP Querier.

SYNOPSIS

igmpQuerier *subcommand options*

DESCRIPTION

Each port's IGMP implementation includes a number of Queriers, which are included in *igmpServer*.

Refer to *IGMP (New)* for an overview of IGMP.

STANDARD OPTIONS

discardLearnedInfo true / false

When disabled, the emulated Querier maintains a complete record state for received reports and sent queries (based on the timer expiry for received groups and sources. (Default = disabled)

When enabled, the Querier does not maintain any database and only sends periodic General Queries. The Specific Query group/source record information is not calculated based on any earlier received report, but solely based on the last received report.

enable true / false

If set to True, enables this IGMP Querier.

enableRouterAlert true / false

If enabled, sets the "Send Router Alert" bit in the IP header.

generalQueryInterval

The amount of time (in seconds) between IGMP General Query messages sent by the querier. (Integer) (Default = 125)

genQueryResponseInterval

The maximum amount of time (in seconds) that the IGMP querier waits to receive a response to a General Query message. (Integer) (Default = 10 seconds, and must be less than the Query Interval)

robustnessVariable

Defines the subnet vulnerability to lost packets. IGMP can recover from robustness variable minus 1 lost packets. The robustness variable should be set to a value of 2 or greater. (Integer) (Default = 2)

specQueryResponseInterval

The maximum amount of time (in seconds) that the IGMP querier waits to receive a response to a Specific Query message. (Integer) (Default = 10 seconds, and must be less than the Query Interval).

specQueryTransmissionCount

Indicates the total number of Specific Query messages sent every Specific Query Response Interval (in seconds) before assuming that there is no interested listener for the particular group/source. (Integer)

startupQueryCount

The number of IGMP General Query messages sent at startup. (Integer) (Default = 2)

supportElection true / false

Indicates whether the Querier participates in Querier election or not. If disabled, then all incoming Query messages are discarded.

supportOlderVersionHost true / false

Indicates whether the Querier will comply with RFC 3376 Section 7.3.2 and RFC 3810 Section 8.3.2. If disabled, all membership reports with version less than the current version are discarded.

supportOlderVersionQuerier

Indicates whether the Querier downgrades to the lowest version of received Query messages. If disabled, all Query messages with version less than the current version are discarded.

version

Indicates the IGMP protocol version to be used. One of:

Option	Value	Usage
igmpQuerierVersion1	1	Uses IGMP Version 1
igmpQuerierVersion2	2	Uses IGMP Version 2
igmpQuerierVersion3	3	Uses IGMP Version 3

isQuerier true / false

(Read-only) If true, indicates that the currently-elected querier is self. If false, indicates that the currently-elected querier is other.

querierAddress

(Read-only) Indicates the IPv4 address of the currently-elected querier. (String)

querierWorkingVersion

(Read-only) Indicates the working version of the IGMP querier at that point in time. (Integer)

COMMANDS

The **igmpQuerier** command is invoked with the following sub-commands. If no subcommand is specified, returns a list of all subcommands available.

igmpQuerier getFirstLearnedInfo

Retrieves the first entry of IGMP learned info from the list.

igmpQuerier getLearnedInfoList

Populates the Learned info list for the IGMP Querier. When it returns TCL_OK, it means that learned info is returned.

igmpQuerier getNextLearnedInfo

Retrieves the next entry of IGMP learned info from the list.

igmpQuerier requestLearnedInfo

Requests the learned IGMP information for the respective IGMP Querier.

igmpQuerier setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *igmpVxServer*.

SEE ALSO

igmpGroupRange, *igmpHost*, *igmpVxServer*, igmpLearnedInfo

NAME - igmpServer

igmpServer — configures the IGMP server parameters.

SYNOPSIS

igmpServer *subcommand options*

DESCRIPTION

The **igmpServer** command is used to configure the IGMP server parameters.

STANDARD OPTIONS

rate

The rate at which reports are sent, expressed in frames per second. *(default = 25)*

reportFrequency

Report frequency in seconds. *(default = 120)*

reportMode

Options are:igmpGroupRange, igmpHost, igmpVxServer, igmpLearnedInfo

Option	Value	Usage
igmpReportToOneWhenQueried	0	
igmpReportToAllWhenQueried	1	
igmpReportToAllUnsolicited	2	(default)

sendRouterAlert true | false

If true, enables sending router alert. *(default = false)*

version

The version number of IGMP. Options are:

Option	Value	Usage
igmpVersion1	1	IGMP Version 1.
igmpVersion2	2	IGMP Version 2.(default)

DEPRECATED OPTIONS

enableQueryResponse true | false

Enables responses after initial join is sent. *(default = true)*

repeatCount

Number of IGMP reports to be sent. Not implemented. *(default = 3)*

COMMANDS

The **igmpServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

igmpServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **igmpServer** command.

igmpServer config *option value*

Modify the IGMP server configuration options of the port. If no *option* is specified, returns a list describing all of the available igmpServer options (see STANDARD OPTIONS) for port.

igmpServer get *chasId cardId portId*

Gets the current IGMP server configuration. Call this command before calling **igmpServer** cget *option value* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.
- Invalid port number.

igmpServer set *chasId cardId portId*

Sets the IGMP server configuration, by reading the configuration option values set by the **igmpServer** config *option value* command. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- The configured parameters are not valid for this port.

igmpServer setDefault

Sets default values for all configuration options.

EXAMPLES

```
package req IxTclHal

set username user

# Check if we're running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts "Could not connect to $host"

return 1

}

}
```

```
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chassis [ixGetChassisID $host]
set chassis [chassis cget -id]
set card 4
set port 1
set portList [list [list $chassis $card $port]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
port setFactoryDefaults $chassis $card $port
# Set up IP address table so we can respond to ARPs
ipAddressTable setDefault
ipAddressTable config -defaultGateway {1.1.1.1}
ipAddressTableItem setDefault
ipAddressTableItem config -fromIpAddress {2.2.2.2}
ipAddressTableItem config -fromMacAddress {00 DE BB 00 00
01}
ipAddressTableItem config -numAddresses 1
ipAddressTableItem set
ipAddressTable addItem
ipAddressTable set $chassis $card $port
# Set up IGMP server to send reports
```

```
igmpServer setDefault
igmpServer config -reportMode 1
igmpServer config -reportFrequency 100
igmpServer config -repeatCount 10
igmpServer set $chassis $card $port
# Set up IGMP table for group addresses
igmpAddressTable clear
igmpAddressTableItem setDefault
igmpAddressTableItem config -fromGroupAddress {224.0.1.1}
igmpAddressTableItem config -fromClientAddress {2.2.2.2}
igmpAddressTableItem config -numGroupAddresses 10
igmpAddressTableItem config -numClientAddresses 1
igmpAddressTableItem set
igmpAddressTable addItem
igmpAddressTable set $chassis $card $port
# Start the protocol server for Arp and IGMP
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enableIgmpQueryResponse true
protocolServer set $chassis $card $port
# Tell the hardware about it
ixWritePortsToHardware portList
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

igmpAddressTable

NAME - igmpSourceRange

igmpSourceRange — configures a multicast source range for an IGMP group range.

SYNOPSIS

igmpSourceRange *subcommand options*

DESCRIPTION

Each port's IGMP implementation includes a number of hosts, which are described in *igmpHost*. Each host is interested in any number of multicast groups, described in *igmpGroupRange*. For each multicast group range, a set of source addresses may be specified in this command.These source ranges constitute a set of IPv4 sources that are to be included or excluded from the group range.

Refer to *IGMP (New)* for an overview.

STANDARD OPTIONS

count

The number of IPv4 addresses in the source range. *(default = 1)*

sourceIpFrom

The starting IPv4 dress for the source range. *(default = 0.0.0.0)*

COMMANDS

The **igmpSourceRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

igmpSourceRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *igmpSourceRange* command.

igmpSourceRange config *option value*

Modify the configuration options of the *igmpSourceRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for igmpSourceRange.

igmpSourceRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *igmpVxServer*.

SEE ALSO

igmpGroupRange, *igmpHost*, *igmpVxServer*

NAME - igmpVxServer

igmpVxServer — accesses the IGMP component of the protocol server for a particular port.

SYNOPSIS

igmpVxServer *subcommand options*

DESCRIPTION

The *igmpVxServer* command is necessary in order to access the IGMP protocol server for a particular port. The *select* subcommand **must** be used before all other IGMP commands. The IGMP simulation covers IGMPv1, IGMPv2 and IGMPv3, although it is only available for processor-based boards.

Each port's IGMP implementation includes a number of hosts, which are described in *igmpHost*. A host is added to the server with the *addHost* subcommand. Each host is interested in any number of multicast groups, described in *igmpGroupRange*. For each multicast group range, a set of source addresses may be specified in *igmpSourceRange*. These source ranges constitute a set of IPV4 sources that are to be included or excluded from the group range.

Refer to *IGMP (New)* for an overview.

STANDARD OPTIONS

enableSendLeaveOnStop *true / false*

If true, enables the Send Leaves on Stop feature (for IGMP versions 2 and 3).

numGroups

The number of multicast groups to transmit every *timePeriod* milliseconds. A value of 0 disables this feature and transmits all groups immediately for all updates. *(default = 0)*

timePeriod

The time period to use for throttling updates, expressed in milliseconds. A value of 0 disables this feature and transmits all groups immediately for all updates. *(default = 0)*

COMMANDS

The **igmpVxServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

igmpVxServer addHost *hostId*

Adds the IGMP host described in the *igmpHost* command to the list of hosts associated with the port. The host's entry in the list is given an identifier of *hostId*. Specific errors are:

- igmpVxServer select has not been called.
- The host parameters in *igmpHost* are invalid.
- A host with this *hostId* exists already in the list.

- Too many groups defined.
- Too many hosts defined.

igmpVxServer addQuerier *routerId*

Adds the IGMP querier described in the *igmpQuerier* command to the list of hosts associated with the port. The host's entry in the list is given an identifier of *routerId*.

igmpVxServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *igmpVxServer* command.

igmpVxServer clearAllHosts

Deletes all the IGMP hosts in the list. Specific errors are:

- igmpVxServer select has not been called.

igmpVxServer clearAllQueriers

Deletes all the IGMP queriers in the list.

igmpVxServer config *option value*

Modify the configuration options of the igmpVxServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for igmpVxServer.

igmpVxServer delHost *hostId*

Deletes the IGMP host described that has an identifier of *hostId*. Specific errors are:

- igmpVxServer select has not been called.
- There is no host with this *hostId* in the list.

igmpVxServer delQuerier *routerId*

Deletes the IGMP querier described that has an identifier of *routerId*.

igmpVxServer generateStreams *chasID cardID portID [action]*

This command creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	*(default)* Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each simulated host and included group range; each stream covers the set of IPV4 addresses associated with the group range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the group range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the group range; it should not be reprogrammed.

igmpVxServer get

Gets the current IGMP server configuration for the last port selected with the *select* subcommand. Call this command before calling the *cget* subcommand to get the value of the configuration option.

igmpVxServer getFirstHost

Access the first IGMP host in the list. The results may be accessed using the *igmpHost* command. Specific errors are:

- igmpVxServer select has not been called.
- There are no hosts in the list.

igmpVxServer getFirstQuerier

Access the first IGMP querier in the list*.* The results may be accessed using the *igmpQuerier* command.

igmpVxServer getNextHost

Access the next IGMP host in the list. The results may be accessed using the *igmpHost* command. Specific errors are:

- igmpVxServer select has not been called.
- igmpVxServer getFirstHost has not been called.
- There are no more hosts in the list.

igmpVxServer getNextQuerier

Access the next IGMP querier in the list*.* The results may be accessed using the *igmpQuerier* command.

igmpVxServer getHost *hostId*

Access the IGMP host with an identifier of *hostId.* The results may be accessed using the *igmpHost* command. Specific errors are:

- igmpVxServer select has not been called.
- There is no host with this *hostId* in the list.

igmpVxServer getQuerier *routerId*

Access the IGMP querier with an identifier of *routerId.* The results may be accessed using the *igmpQuerier* command.

igmpVxServer select *chasID cardID portID*

Accesses the IGMP component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The IGMP protocol package has not been installed.
- Invalid port specified.
- igmpVxServer is not supported on this older style port.

igmpVxServer set

Sets the configuration of the IGMP server in IxHAL for the port last selected with the *select* subcommand by reading the configuration option values set by the *config subcommand*. Specific errors are:

- No connection to a chassis.
- The port is being used by another user.
- Configured parameters are not valid for this setting.

igmpVxServer setDefault

Sets default values for all configuration options.

igmpVxServer setHost *[hostId]*

Sets the values for the host's entry in the list with an identifier of *hostId* based on changes made through the *igmpHost* command. This command should be used to change a running configuration and must be followed by an *igmpVxServer write* command in order to send these changes to the protocol server. Specific errors are:

- A host with this *hostId* does not exist in the list.
- Too many groups defined.
- Too many hosts defined.

igmpVxServer setQuerier *[routerId]*

Sets the values for the querier's entry in the list with an identifier of *routerId* based on changes made through the *igmpQuerier* command. This command should be used to change a running configuration and must be followed by an *igmpVxServer write* command in order to send these changes to the protocol server.

igmpVxServer write

Sends any changes made with the IGMP suite of commands to the protocol server for immediate application. This command **must** be used in order for their changes to have an effect.

EXAMPLES

```
package req IxTclHal
set host astro
set username user
# Check if we’re running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts “Could not connect to $host”
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set ch [ixGetChassisID $host]
set card 12
set port 1
set portList [list [list $ch $card $port]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
port setFactoryDefaults $ch $card $port
```

```
protocolServer config -enableIgmpQueryResponse true
protocolServer set $ch $card $port
#Select the port
igmpVxServer select $ch $card $port
igmpVxServer clearAllHosts
# Configure a source range
igmpSourceRange config -sourceIpFrom 12.1.4.5
igmpSourceRange config -count 1
# Add the source range to the group range
if [igmpGroupRange addSourceRange source1] {
logMsg "Error in adding sourceRange"
}
# Configure groupRange
igmpGroupRange config -enable true
igmpGroupRange config -groupIpFrom 224.0.0.1
# Add the group range to the host
if [igmpHost addGroupRange group1] {
logMsg "Error adding groupRange group1"
}
igmpGroupRange config -enable true
igmpGroupRange config -groupIpFrom 224.1.2.6
if [igmpHost addGroupRange group2] {
logMsg "Error adding groupRange group2"
}
#Configure host - assume that an interface description exists
igmpHost config -enable true
igmpHost config -protocolInterfaceDescription "$card:0$port"
igmpHost config -version igmpHostVersion3
# Add the host to the IGMPv3 server
if [igmpVxServer addHost host1] {
logMsg "Error adding host"
}
# Send to the hardware
igmpVxServer set
```

```
if [igmpVxServer write] {
logMsg “Error writing”
}
# The configuration was sent to the hardware. At this point if you
refresh
# To get an object:
# Make sure you apply the hierarchy to get objects
# Be consistent in using Ids. If you are using Ids, use it all the
way
# and don’t mix it with getFirst/getNext methods.
# Example of disabling host on the fly ( when IGMP server is
running )
# Select the port
igmpVxServer select $ch $card $port
# Get the host by name
igmpVxServer getHost host1
# Disable it
igmpHost config -enable 0
# And set back to the hardware
igmpVxServer setHost host1
igmpVxServer write
# Example of modifying group Range on the fly
igmpVxServer select $ch $card $port
igmpVxServer getHost host1
igmpHost getGroupRange group2
igmpGroupRange config -groupIpFrom 224.1.20.100
if [igmpHost setGroupRange group2 ] {
logMsg “Error in setting group range group2”
}
igmpVxServer write
# Example of modifying source Range on the fly
igmpVxServer select $ch $card $port
igmpVxServer getHost host1
igmpHost getGroupRange group2
```

```
igmpGroupRange getSourceRange source1
igmpSourceRange config -count 20
if [igmpGroupRange setSourceRange source1] {
logMsg “Error in setting source range”
}
igmpVxServer write
# Example of generating streams at server level for enabled hosts
and group ranges.
set targetCh 1
set targetCard 12
set targetPort 2
set targetPortList [list [list $targetCh $targetCard $targetPort]]
igmpVxServer select $ch $card $port
igmpVxServer generateStreams $targetCh $targetCard $targetPort
ixWriteConfigToHardware targetPortList
# Example of generating streams at group range level for enabled
group range.
# You can get the group range by name too. Here is an example of
using getFirst/getNext.
igmpVxServer select $ch $card $port
igmpVxServer getFirstHost
igmpHost getFirstGroupRange
igmpGroupRange generateStreams $targetCh $targetCard $targetPort
ixWriteConfigToHardware targetPortList
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

igmpGroupRange, *igmpHost*, *igmpSourceRange*, *igmpQuerier*, *igmpLearnedInfo*

NAME - isisGrid

isisGrid — sets up a simulated grid of ISIS routers.

SYNOPSIS

isisGrid *subcommand options*

DESCRIPTION

This command allows a grid of IS-IS routers to be defined. The grid is added to a particular simulated router via the *isisRouter* *addGrid* command. The features of the grid configured in this command are:

- Size — the number of columns and rows of nodes.
- Entry point — the location in the grid to which the simulated router is connected.
- Router IDs — the router IDs associated with each element of the grid.
- Link type — whether the links between the simulated router.

The features of a grid which are held in this command are:

- Internode routes — the networks between the simulated router and the grid and the nodes within the grid. These are configured using the *isisGridInternodeRoute* object and added to this command with the *addInternodeRoute* subcommand.
- Node routes — the routes advertised by each of the nodes within the grid. These are configured using the *isisGridRoute* object and added to this command with the *addRoute* subcommand.
- Outside links — attachments to the grid to other outside points, which may be places in other grids. These are configured using the *isisGridOutsideLink* command and added to this command with the *addOutsideLink* subcommand.
- Traffic Engineering data — a default set of TE data may be associated with all nodes using the *isisGridRangeTe* command. This data may be overridden for the node connecting the grid to the simulated router using the *isisGridEntryTe* command. Individual paths through the grid may further override these values using the *isisGridTePath* command and added to this command with the *addTePath* subcommand.

STANDARD OPTIONS

enable true | false

Enables the use of this ISIS grid. *(default = false)*

enableTe true | false

Enables the generation of Traffic Engineering data, as described in the *teRouterId, teRouterIdIncrementBy* and *overrideEntryTe* options and in the *isisGridRangeTe*, *isisGridEntryTe* and *isisGridTePath* commands. *(default = false)*

enableUserWideMetric true | false

Enables the use of wide metrics for *interfaceMetric. (default = true)*

entryPointColumn

The simulated router is connected to a router in the grid at a particular row and column location. This option is the column number. *(default = 1)*

entryPointRow

The simulated router is connected to a router in the grid at a particular row and column location. This option is the row number. *(default = 1)*

firstRouterId

This is the router ID of the first router in the grid (at row = 0, column = 0), in 6-byte hex format. Routers are assigned individual router IDs by adding the value of the *routerIdIncrementBy* value to each element in the grid, in a row first manner. *(default = {00 00 00 00 00 00})*

interfaceMetric

The metric for the interface connected to the grid. *(default = 1)*

linkType

The type of link between the simulated router and the entry point node and between all nodes in the grid. One of:

Option	Value	Usage
isisPointToPoint	0	*(default)* a point to point network
isisBroadcast	1	a broadcast network

numColumns

The number of columns in the simulated grid. *(default = 3)*

numRows

The number of rows in the simulated grid. *(default = 3)*

overrideEntryTe true | false

If *true,* then the TE default values for all nodes set in the *isisGridRangeTe* command are overridden for the entry point grid node by the TE values set in the *isisGridEntryTe* element. *(default = false)*

routerIdIncrementBy

The value used to increment *firstRouterId* by as each router is assigned a unique router ID. *(default = {00 00 00 00 00 01})*

teRouterId

If *enableTe* is *true*, then this is the TE router ID of the first router in the grid (at row = 0, column = 0), in IPv4 format. Routers are assigned individual TE router IDs by adding the value of the *teRouterIdIncrementBy* value to each element in the grid, in a row first manner. *(default = 0.0.0.1)*

teRouterIncrementBy

The value used to increment *teRouterId* by as each router is assigned a unique TE router ID. *(default = 0.0.0.1)*

enableHostName

If true, the given dynamic host name is transmitted in all the packets sent from this router.

hostNamePrefix

Allows to add a host name to this network range. The name prefix is appended by row ID and column ID in ".<rowid.<colid" combination.

COMMANDS

The **isisGrid** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisGrid addInternodeRoute

Adds the network described in the *isisGridInternodeRoute* command to the list of internode routes associated with the grid. Each network is applied for all the connections between the simulated router and the grid entry point as well as all node to node connections. Specific errors are:

- Invalid internode route configuration

isisGrid addOutsideLink

Adds the outside link described in the *isisGridOutsideLink* command to the list of outside links associated with the grid. Specific errors are:

- Invalid outside link configuration.

isisGrid addRoute

Adds the node route described in the *isisGridRoute* command to the list of routes associated with the grid. Each route is applied for all the interfaces in the grid. Specific errors are:

- Invalid grid route configuration.

isisGrid addTePath

Adds the TE path described in the *isisGridTePath* command to the list of TE paths associated with the grid. Specific errors are:

- Invalid TE path configuration.

isisGrid cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *isisGrid* command.

isisGrid clearAllInternodeRoutes

Deletes all the internode routes in the list.

isisGrid clearAllOutsideLinks

Deletes all the outside links in the list.

isisGrid clearAllRoutes

Deletes all the node routes in the list.

isisGrid clearAllTePaths

Deletes all the TE paths in the list.

isisGrid config *option value*

Modify the configuration options of the isisGrid. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisGrid.

isisGrid delInternodeRoute

Deletes the currently selected internode route, as accessed through the use of the *getFirstInternodeRoute* and *getNextInternodeRoute* subcommands. Specific errors are:

- *isisGrid getFirstInternodeRoute* has not been called
- *isisGrid getNextInternodeRoute* has run off the end of the list

isisGrid delOutsideLink

Deletes the currently selected outside link, as accessed through the use of the *getFirstOutsideLink* and *getNextOutsideLink* subcommands. Specific errors are:

- *isisGrid getFirstOutsideLink* has not been called
- *isisGrid getNextOutsideLink* has run off the end of the list

isisGrid delRoute

Deletes the currently selected node route, as accessed through the use of the *getFirstRoute* and *getNextRoute* subcommands. Specific errors are:

- *isisGrid getFirstRoute* has not been called
- *isisGrid getNextRoute* has run off the end of the list

isisGrid delTePath

Deletes the currently selected TE path, as accessed through the use of the *getFirstTePath* and *getNextTePath* subcommands. Specific errors are:

- *isisGrid getFirstTePath* has not been called
- *isisGrid getNextTePath* has run off the end of the list

isisGrid getFirstInternodeRoute

Access the first internode route in the list. The results may be accessed using the *isisGridInternodeRoute* command. Specific errors are:

- There are no internode routes in the list.

isisGrid getFirstOutsideLink

Access the first outside link in the list*.* The results may be accessed using the *isisGridOutsideLink* command. Specific errors are:

- There are no outside links in the list.

isisGrid getFirstRoute

Access the first node route in the list*.* The results may be accessed using the *isisGridRoute* command. Specific errors are:

- There are no node routes in the list.

isisGrid getFirstTePath

Access the first TE path in the list. The results may be accessed using the *isisGridTePath* command. Specific errors are:

- There are no TE paths in the list.

isisGrid getNextInternodeRoute

Access the next internode route in the list. The results may be accessed using the *isisGridInternodeRoute* command. Specific errors are:

- *isisGrid getFirstInternodeRoute* has not been called.
- There are no more internode routes in the list.

isisGrid getNextOutsideLink

Access the next outside link in the list. The results may be accessed using the *isisGridOutsideLink* command. Specific errors are:

- *isisGrid getFirstOutsideLink* has not been called.
- There are no more outside links in the list.

isisGrid getNextRoute

Access the next node route in the list. The results may be accessed using the *isisGridRoute* command. Specific errors are:

- *isisGrid getFirstInternodeRoute* has not been called.
- There are no more node routes in the list.

isisGrid getNextTePath

Access the next TE path in the list*.* The results may be accessed using the *isisGridTePath* command. Specific errors are:

- *isisGrid getFirstInternodeTePath* has not been called.
- There are no more TE paths in the list.

isisGrid setDefault

Sets default values for all configuration options.

EXAMPLES

```
package req IxTclHal
# Define parameters used by ISIS router
set host localhost
set username user
# Check if we're running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set ch [ixGetChassisID $host]
# Port is: card 4, port 1
set ca 4
set po 1
set pl [list [list $ch $ca $po]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
```

```
if [ixTakeOwnership $pl] {
ixPuts $::ixErrorInfo
return 1
}
set myMac {00 0a de 01 01 01}
set router 101.101.12.2
set neighbor 101.101.12.1
set interfaceIpMask 255.255.255.0
set numberOfRoute 1650
# Select port to operate
isisServer select $ch $ca $po
# Basic grid parameters
isisGrid config -enable true
isisGrid config -numRows 10
isisGrid config -numColumns 10
# Configure the addresses of all of the nodes and interfaces
isisGridRoute config -networkIpAddress 10.0.0.0
if [isisGrid addRoute ] {
logErr "Error in isisGrid addRoute"
}
# Add three outside links
# 1) Single host route from 0.65.20.0
# 2) Single host route from 10.0.0.0
# 3) An IPv6 set of networks from 0:0:0:0:0:0:FE06:7000
isisGridInternodeRoute setDefault
isisGridInternodeRoute config -ipAddress {0.65.20.0}
isisGridInternodeRoute config -ipType addressTypeIpV4
isisGridInternodeRoute config -ipMask 32
if [isisGridOutsideLink addRoute ] {
logErr "Error in isisGrid addRoute"
}
isisGridInternodeRoute setDefault
isisGridInternodeRoute config -ipAddress {10.0.0.0}
isisGridInternodeRoute config -ipType addressTypeIpV4
isisGridInternodeRoute config -ipMask 32
if [isisGridOutsideLink addRoute ] {
logErr "Error in isisGrid addRoute"
}
isisGridInternodeRoute setDefault
isisGridInternodeRoute config -ipAddress {0:0:0:0:0:0:FE06:7000}
isisGridInternodeRoute config -ipType addressTypeIpV6
isisGridInternodeRoute config -ipMask 64
if [isisGridOutsideLink addRoute ] {
logErr "Error in isisGrid addRoute"
}
# Create an outside link to (1,1)
isisGridOutsideLink setDefault
isisGridOutsideLink config -connectionRow 1
isisGridOutsideLink config -connectionColumn 1
isisGridOutsideLink config -metric 10
isisGridOutsideLink config -administrativeGroup {00 00 21 21}
isisGridOutsideLink config -maxBandwidth 0.0
if [isisGrid addOutsideLink] {
logErr "Error in isisGrid addOutsideLink"
}
# Enable Traffic engineering and set the TE IDs
isisGrid config -enableTe true
isisGridRangeTe config -linkMetric 1
# Now add the grid to the router and the router to the server
if [isisRouter addGrid grid1] {
logErr "Error in isisRouter addGrid"
}
if [isisServer addRouter r1] {
logErr "Error in isisServer addRouter"
}
if [isisServer write ] {
logErr "Error in isisServer write"
}
```

```
# Change the Mask of internodeRoute of the first outside Link on
fly
isisServer select $chassis $card $port
isisServer getRouter r1
isisRouter getGrid grid1
isisGrid getFirstOutsideLink
#delete the outsideLink here and add it later to see the changes
isisGrid delOutsideLink
isisGrid getFirstRoute
isisGrid getNextRoute
isisGrid delRoute
isisGridInternodeRoute config -ipMask 12
isisGridOutsideLink addRoute
isisGrid addOutsideLink
isisRouter setGrid grid1
isisServer write
# Let go of the ports that we reserved
ixClearOwnership $pl
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

isisRouter, *isisGridEntryTe*, *isisGridInternodeRoute*, *isisGridOutsideLink*, *isisGridRangeTe*, *isisGridRoute*, *isisGridTePath*

NAME - isisGridEntryTe

isisGridEntryTe — describes the TE parameters associated with the entry point node in an ISIS grid.

SYNOPSIS

isisGridEntryTe *subcommand options*

DESCRIPTION

This command overrides the default TE data values for all nodes in the ISIS grid set with the *isisGridRangeTe* command. The *enableTe* and *overrideEntryTe* options in the *isisGrid* command must be set to *true* for this data to be used.

STANDARD OPTIONS

administrativeGroup

The administrative group associated with the node, in 4-byte hex format. *(default = {00 00 00 00})*

linkMetric

The metric associated with the interface that the TE data is advertised on. *(default = 0)*

maximumBandwidth

The maximum bandwidth to be advertised. *(default = 0.0)*

maxReservable Bandwidth

The maximum reservable bandwidth to be advertised. *(default = 0.0)*

unreservedBandwidth Priority0-7

The unreserved bandwidth for each priority to be advertised. There are eight distinct options. *(default = 0.0)*

COMMANDS

The **isisGridEntryTe** command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

isisGridEntryTe cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisGridEntryTe** command.

isisGridEntryTe config *option value*

Modify the configuration options of the isisGridEntryTe. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisGridEntryTe.

isisGridEntryTe setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *isisGrid*.

SEE ALSO

isisRouter, *isisGrid*, *isisGridInternodeRoute*, *isisGridOutsideLink*, *isisGridRangeTe*, *isisGridRoute*, *isisGridTePath*

NAME - isisGridInternodeRoute

isisGridInternodeRoute — defines a network between grid elements.

SYNOPSIS

isisGridInternodeRoute *subcommand options*

DESCRIPTION

This command is used in conjunction with *isisGrid* and *isisGridOutsideLink*.

In conjunction with *isisGrid*:

Internode routes are the networks between the simulated router and the grid and the nodes within the grid. These are configured using this object and added to the grid with the *isisGrid* *addInternodeRoute* subcommand.

The network specified in the options of this command is applied iteratively to:

- The connection between the simulated router and the entry point node in the grid. A new interface is created on the simulated router.
- The connections between all the nodes in each row. That is, the connection between the first and second nodes in the first row, then the second and third nodes in the first row... then between the first and second nodes in the second row and so forth.
- The connection between each node in a row and the node below it, in a row first manner. That is, the connection between the first node in the first and second rows, then the second node in the first and second rows...then between the first node in the second and third rows and so forth.

The network part of the *ipAddress* option is incremented by *ipStep* between uses. The interface to the left or above in the connection receives the ".1" address on the network and the other receives the next address. The simulated router is considered to be "above" all others.

In conjunction with *isisGridOutsideLink*:

This command is used to describe a network associated with an outside link to the grid.

STANDARD OPTIONS

ipAddress

The IP address of the network to be used. *(default = 0.0.0.0)*

ipMask

The number of bits in the network mask of *ipAddress*. *(default = 24)*

ipStep

The step between assigned networks. *(default = 1)*

ipType

The IP addressing type of *ipAddress*, one of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address.
addressTypeIpV6	18	An IPv6 address.

COMMANDS

The **isisGridInternodeRoute** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisGridInternodeRoute cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisGridInternodeRoute** command.

isisGridInternodeRoute config *option value*

Modify the configuration options of the isisGridInternodeRoute. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisGridInternodeRoute.

isisGridInternodeRoute setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *isisGrid*.

SEE ALSO

isisRouter, *isisGrid*, *isisGridEntryTe*, *isisGridOutsideLink*, *isisGridRangeTe*, *isisGridRoute*, *isisGridTePath*

NAME - isisGridOutsideLink

isisGridOutsideLink — sets up the outside links between an ISIS grid and another ISIS grid.

SYNOPSIS

isisGridOutsideLink *subcommand options*

DESCRIPTION

This command describes an attachment from the grid to other outside points, which may be places in other grids. These are added to the grid with the *isisGrid addOutsideLink* command.

Multiple IPv4 and IPv6 networks may be associated with the outside link. These are set up with *isisGridInternodeRoute* and added to this command with the *addRoute* subcommand.

STANDARD OPTIONS

administrativeGroup

The administrative group associated with the outside router being linked to, in 4-byte hex format. *(default = {00 00 00 00})*

connectionColumn

The outside link occurs at a particular node within the grid. This is the column number of the node. *(default = 1)*

connectionRow

The outside link occurs at a particular node within the grid. This is the row number of the node. *(default = 1)*

linkedRouterId

The router ID of the outside router. *(default = {00 00 00 00 00 00})*

maximumBandwidth

The maximum bandwidth to be advertised to the outside link. *(default = 0.0)*

maxReservable Bandwidth

The maximum reservable bandwidth to be advertised to the outside link. *(default = 0.0)*

metric

The metric associated with the connection. *(default = 0)*

unreservedBandwidth Priority0-7

The unreserved bandwidth for each priority to be advertised to the outside link. There are eight distinct options. *(default = 0.0)*

COMMANDS

The **isisGridOutsideLink** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisGridOutsideLink addRoute

Adds the network described in the *isisGridInternodeRoute* command to the list of routes associated with the outside link. Specific errors are:

- Invalid outside link configuration

isisGridOutsideLink cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisGridOutsideLink** command.

isisGridOutsideLink clearAllRoutes

Deletes all the routes in the list.

isisGridOutsideLink config *option value*

Modify the configuration options of the isisGridOutsideLink. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisGridOutsideLink.

isisGridOutsideLink delRoute

Deletes the currently selected route, as accessed through the use of the *getFirstRoute* and *getNextRoute* subcommands. Specific errors are:

- *isisGridOutsideLink getFirstRoute* has not been called
- *isisGridOutsideLink getNextRoute* has run off the end of the list

isisGridOutsideLink getFirstRoute

Access the first route in the list. The results may be accessed using the *isisGridInternodeRoute* command. Specific errors are:

- There are no routes in the list

isisGridOutsideLink getNextRoute

Access the next route in the list. The results may be accessed using the *isisGridInternodeRoute* command. Specific errors are:

- *isisGridOutsideLink getFirstRoute* has not been called
- There are no more routes in the list

isisGridOutsideLink setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under isisGrid.

SEE ALSO

isisRouter, *isisGrid*, *isisGridEntryTe*, *isisGridInternodeRoute*, *isisGridRangeTe*, *isisGridRoute*, *isisGridTePath*

NAME - isisGridRangeTe

isisGridRangeTe — describes the default TE parameters associated with all nodes in an ISIS grid.

SYNOPSIS

isisGridRangeTe *subcommand options*

DESCRIPTION

This command sets the default TE data values for all nodes in the ISIS grid. The *enableTe* option in the *isisGrid* command must be set to *true* for this data to be used. This data may be overridden for the node connecting the grid to the simulated router using the *isisGridEntryTe* command. Individual paths through the grid may further override these values using the *isisGridTePath* command and added to this command with the *addTePath* subcommand.

STANDARD OPTIONS

administrativeGroup

The administrative group associated with the node, in 4-byte hex format. *(default = {00 00 00 00})*

linkMetric

The metric associated with the interface that the TE data is advertised on. *(default = 0)*

maximumBandwidth

The maximum bandwidth to be advertised. *(default = 0.0)*

maxReservable Bandwidth

The maximum reservable bandwidth to be advertised. *(default = 0.0)*

unreservedBandwidth Priority0-7

The unreserved bandwidth for each priority to be advertised. There are eight distinct options. *(default = 0.0)*

COMMANDS

The **isisGridRangeTe** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisGridRangeTe cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisGridRangeTe** command.

isisGridRangeTe config *option value*

Modify the configuration options of the isisGridRangeTe. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisGridRangeTe.

isisGridRangeTe setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *isisGrid*.

SEE ALSO

isisRouter, *isisGrid*, *isisGridEntryTe*, *isisGridInternodeRoute*, *isisGridOutsideLink*, *isisGridRoute*, *isisGridTePath*

NAME - isisGridRoute

isisGridRoute — sets up the route ranges advertised between ISIS grid nodes.

SYNOPSIS

isisGridRoute *subcommand options*

DESCRIPTION

These are the routes advertised by each of the nodes within the grid. These are added to the grid using the *isisGrid* *addRoute* command.

The route range specified in the options of this command is applied iteratively to:

- The interface connecting each node in a row with its right neighbor. That is, the interface from the first node to the second node in the first row, then the second and third nodes in the first row... then between the first and second nodes in the second row and so forth.
- The interface connecting each node in a row with its neighbor below it, in a row first manner. That is, the interface from the first node in the first and second rows, then the second node in the first and second rows...then between the first node in the second and third rows and so forth.

The number of networks indicated by *numberOfNetworks* is advertised for a node and then the network part of the *networkIpAddress* option is incremented by *nodeStep* between uses. For example, if the following settings were used:

Option	Value
ipType	addressTypeIpV4
networkIpAddress	60.0.0.0
nodeStep	256
numberOfNetworks	50
prefix	24

Then the interface from the node at row 0, column 0 to the node at row 0, column 1 would advertise the networks 60.0.0.0/24 through 60.0.49/24. The interface from row 0, column 1 to row 0, column2 would advertise the networks 60.1.0.0/24 through 60.1.49.0/24.

STANDARD OPTIONS

enable true / false

Enables the use of this route range. *(default = false)*

enableRedistributed true | false

If *true,* the route will be distributed down. If *false*, the route will be redistributed up. *(default = false)*

ipType

The IP addressing type of the range, one of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

metric

The cost metric associated with the route. *(default = 0)*

networkIpAddress

The IP address of the routes to be advertised. *(default = 0.0.0.0)*

nodeStep

The increment to be applied to the network part of *networkIpAddress* between per node uses. *(default = 256)*

numberOfNetworks

The number of prefixes to be advertised. *(default = 1)*

prefix

The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0. *(default = 24)*

routeOrigin

The origin of the route, one of:

Option	Value	Usage
isisRouteInternal	0	*(default)* The route originated internally.
isisRouteExternal	1	The route originated externally.

COMMANDS

The **isisGridRoute** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisGridRoute cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisGridRoute** command.

isisGridRoute config *option value*

Modify the configuration options of the isisGridRoute. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisGridRoute.

isisGridRoute setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *isisGrid*.

SEE ALSO

isisRouter, *isisGrid*, *isisGridEntryTe*, *isisGridInternodeRoute*, *isisGridOutsideLink*, *isisGridRangeTe*, *isisGridTePath*

NAME - isisGridTePath

isisGridTePath — describes the TE parameters associated with a particular path through an ISIS grid.

SYNOPSIS

isisGridTePath *subcommand options*

DESCRIPTION

This command overrides the default TE data values for all nodes in the ISIS grid set with the *isisGridRangeTe* command. The *enableTe* option in the *isisGrid* command must be set to *true* for this data to be used.

The path starts and ends with particular nodes in the grid. The row and column of the end point must be greater than or equal to those of the start point. The path through the grid is described in row and column step sizes. For example, if the grid is 8 x 8, then a path starting at (row = 0, column = 0) and ending at (3, 6) with a row step of 1 and a column step of 2 will go through the following grid nodes: (0,0), (1, 2), (2, 4) and (3, 6). Any excess row or column step values which would take the path past the endpoint are truncated.

STANDARD OPTIONS

administrativeGroup

The administrative group associated with the path, in 4-byte hex format. *(default = {00 00 00 00})*

columnStep

The column step size for the path. *(default = 0)*

enableBidirectional true | false

If *true*, then the reverse path with also be advertised. *(default = true)*

endColumn

The column number of the end of the path. *(default = 1)*

endRow

The row number of the end of the path. *(default = 1)*

maximumBandwidth

The maximum bandwidth to be advertised. *(default = 0.0)*

maxReservable Bandwidth

The maximum reservable bandwidth to be advertised. *(default = 0.0)*

metric

The metric associated with the interface that the TE data is advertised on. *(default = 0)*

rowStep

The row step size for the path. *(default = 0)*

startColumn

The column number of the start of the path. *(default = 1)*

startRow

The row number of the start of the path. *(default = 1)*

unreservedBandwidth Priority0-7

The unreserved bandwidth for each priority to be advertised. There are eight distinct options. *(default = 0.0)*

COMMANDS

The **isisGridTePath** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisGridTePath cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisGridTePath** command.

isisGridTePath config *option value*

Modify the configuration options of the isisGridTePath. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisGridTePath.

isisGridTePath setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *isisGrid*.

SEE ALSO

isisRouter, *isisGrid*, *isisGridEntryTe*, *isisGridInternodeRoute*, *isisGridOutsideLink*, *isisGridRangeTe*, *isisGridTePath*

NAME - isisInterface

isisInterface — displays an interface for an ISIS router.

SYNOPSIS

isisInterface *subcommand options*

DESCRIPTION

The *isisInterface* holds the information related to a single interface on the simulated router. Interfaces are added into the *isisRouter* interface list using the *isisRouter addInterface* command. Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on ISIS testing with Ixia equipment. Refer to *isisInterface* for an overview. This information is pulled from existing interface during *isisRouter addInterface* command, and is read-only.

STANDARD OPTIONS

administrativeGroup

The administrative group associated with the interface. *(default = {00 00 00 00})*

enableBfdRegistration

Indicates if a BFD session is to be created to the ISIS peer IP address once the ISIS session is established. This allows ISIS to use BFD to maintain IPv4 connectivity the ISIS peer.

circuitAuthType

The type of authentication to be used for IIHs. One of:

Option	Value	Usage
isisAuthTypeNone	0	(default) No authentication
isisAuthTypePassword	1	Clear text passwords are used. The password in *circuitTxPassword* is used in transmitted IIHs and the passwords in *circuitRxPassword* are used as valid passwords on received IIHs.
isisAuthTypeMD5		Message Digest 5 (MD5) authentication is used. An *md5Key* (password) must be specified by the user.

circuitRxPasswordList

If *circuitAuthType* is *isisAuthTypePassword*, then this is a list of passwords that the router will accept on received IIHs. Note that passwords set through IxExplorer that use characters special to Tcl (such as {}, [] and quotes) may not be retrieved correctly from an *isisRouterget* operation. *(default = {})*

circuitTxPassword

If *circuitAuthType* is *isisAuthTypePassword*, then this is the password (or MD5Key) that will be sent with transmitted IIHs. Note that passwords set through IxExplorer that use characters special to Tcl (such as {}, [] and quotes) may not be retrieved correctly from an *isisRouterget* operation. *(default = "")*

configuredHoldTime

The configured hold time before this interface accepts messages.

connectToDut true / false

If set, this IS-IS interface is directly connected to the DUT. *(default = false)*

deadIntervalLevel1
dealIntervalLevel2

The dead interval used with the Level 1 or Level 2 aspect of the interface, expressed in seconds. Used to determine if neighbor routers are non-operational. *(default = 30)*

enable true / false

If set, enables the use of this route range for the simulated router. *(default = false)*

enableAutoConfigure Area true / false

If *true*, the area for the interface is determined during the hello interchange with the DUT. *(default = true)*

enableAutoAdjustMTU true / false

If *true*, and a padded HELLO message is received on the interface, then the interfaces MTU will be adjusted to match the packet length of the received HELLO message. *(default = false)*

enableAutoAdjust Protocols true / false

If *true*, and a HELLO message is received which contains a protocols TLV, then the interfaces protocols will be adjusted to match the received TLV. *(default = false)*

enableConfiguredHold Time true / false

If *true*, a hold time based on the value entered in *configuredHoldTime* is observed before the interface accepts Messages. *(default = false)*

helloIntervalLevel1/
helloIntervalLevel2

The hello interval used with the Level 1 or Level 2 aspect of the interface, expressed in seconds. Used to send regular messages to neighbor IS-IS routers. *(default = 10)*

interfaceId

Read only. The OSI interface ID for this interface. This value is pulled from existing interface during the *isisRouter addInterface* command. *(default = {00 00 00 00 00 00})*

level

The IS-IS level associated with the interface, one of:

Option	Value	Usage
isisLevel1	1	level 1 interface
isisLevel2	2	*(default)* level 2 interface
isisLevel1Level2	3	level 1 and level 2 interface

maximumBandwidth

The maximum bandwidth to be advertised. *(default = 0.0)*

maxReservable Bandwidth

The maximum reservable bandwidth to be advertised. *(default = 0.0)*

metric

The cost metric associated with the route. *(default = 10)*

networkType

Indicates the type of network attached to the interface: broadcast or point-to-point. One of:

Options	Value	Usage
isisBroadcast	1	a broadcast network
isisPointToPoint	2	*(default)* a point to point network

priorityLevel1/ priorityLevel2

The priority level associated with the Level 1 or Level 2 aspect of the interface. This is used in master election. *(default = X)*

protocolInterface Description

The *description* option associated with an *isisServer* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

teMetric

The traffic engineering metric, used for both L1 and L2 routes. *(default = 0)*

unreservedBandwidth Priority0-7

The unreserved bandwidth for each priority to be advertised. There are eight distinct options. *(default = 0.0)*

enable3Way Handshaking

If true, Ixia emulated point-to-point circuit will include 3-way TLV in its P2P IIH and attempt to establish the adjacency as specified in RFC 5303.

extendedLocalCircuitId

The integer value of the local circuit ID.

The default is 1.

enableHelloPadding

If true, hellopadding is enabled.

DEPRECATED OPTIONS

ipAddress

The IP address for this interface. Only used if *protocolInterfaceDescription* is empty. *(default = 0.0.0.0)*

ipMask

The IP mask associated with the IP address for this interface. Only used if *protocolInterfaceDescription* is empty. *(default = 255.255.255.0)*

COMMANDS

The **isisInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisInterface** command.

isisInterface config *option value*

Modify the configuration options of the isisInterface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisInterface.

isisInterface setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *isisRouteRange*

NAME - isisRouter

isisRouter — configures an ISIS router.

SYNOPSIS

isisRouter *subcommand options*

DESCRIPTION

The *isisRouter* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *isisRouteRange* command.
- Interfaces — router interface, constructed in the *isisInterface* command.
- Grids — simulated grids of routers behind the router. A virtual interface is generated for each grid.

Routers defined in this command are added to an *isisServer* using the *isisServer addRouter* command. Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on IS-IS testing with Ixia equipment. Refer to *isisRouter*4 for an overview of this command.

STANDARD OPTIONS

areaAddressList

The list of area addresses to use. Expressed as a Tcl list with commas separating the elements. *(default = { })*

areaAuthType

The type of authentication to be used for Level 1 LSPs. One of:

Option	Value	Usage
isisAuthTypeNone	0	(default) No authentication
isisAuthTypePassword	1	Clear text passwords are used. The password in *areaTxPassword* is used in transmitted LSPs and the passwords in *areaRxPassword* are used as valid passwords on received LSPs.
isisAuthTypeMD5	2	Message Digest 5 (MD5) authentication is used. An *md5Key* (password) must be specified by the user.

areaRxPasswordList

If *areaAuthType* is *isisAuthTypePassword*, then this is a list of passwords that the router will accept on received LSPs. Note that passwords set through IxExplorer that use characters special to Tcl (such as {}, [] and quotes) may not be retrieved correctly from an *isisRouter get* operation. *(default = {})*

areaTxPassword

If *areaAuthType* is *isisAuthTypePassword*, then this is the password (or MD5Key) that will be sent with transmitted LSPs. Note that passwords set through IxExplorer that use characters special to Tcl (such as {}, [] and quotes) may not be retrieved correctly from an *isisRouterget* operation. *(default = "")*

domainAuthType

The type of authentication to be used for Level 2 LSPs. One of:

Option	Value	Usage
isisAuthTypeNone	0	(default) No authentication
isisAuthTypePassword	1	Clear text passwords are used. The password in *domainTxPassword* is used in transmitted LSPs and the passwords in *domainRxPassword* are used as valid passwords on received LSPs.
isisAuthTypeMD5	2	Message Digest 5 (MD5) authentication is used. An *md5Key* (password) must be specified by the user.

domainRxPasswordList

If *domainAuthType* is *isisAuthTypePassword*, then this is a list of passwords that the router will accept on received LSPs. Note that passwords set through IxExplorer that use characters special to Tcl (such as {}, [] and quotes) may not be retrieved correctly from an *isisRouterget* operation. *(default = {})*

domainTxPassword

If *domainAuthType* is *isisAuthTypePassword*, then this is the password (or MD5Key) that will be sent with transmitted LSPs. Note that passwords set through IxExplorer that use characters special to Tcl (such as {}, [] and quotes) may not be retrieved correctly from an *isisRouter get* operation. *(default = "")*

enable true | false

Enables the use of this router in the simulated ISIS network. *(default = false)*

enableAttached true | false

"

For L2 only, if *true*, indicates that the *AttachedFlag* is set. This indicates that this ISIS router can use L2 routing to reach other areas. *(default = true)*

enableAutoLoopback Address true | false

If *true*, a /32 route is added that matches *routerId*. *(default = false)*

enableDiscardLearned LSPs true | false

If *true*, LSPs learned from this router's interfaces will be discarded. *(default = true)*

enableHitlessRestart true | false

If *true,* hitless restart support is enabled on this router. *(default = false)*

enableOverload true | false

If *true*, the LSP database overload bit is set, indicating that the LSP database on this router does not have enough memory to store a received LSP. *(default = false)*

enablePartitionRepair true | false

If *true*, enables the optional partition repair option specified in ISO/IEC 10589 and RFC 1195 for Level 1 areas. *(default = false)*

enableTraffic Engineering true | false

If *true*, enables the traffic engineering data. *(default = false)*

enableWideMetric true | false

If *true*, enables the use of wide metrics. *(default = false)*

hitlessRestartMode

If *enableHitlessRestart* is *true*, this indicates the mode in which this router is to operate. One of:

Option	Value	Usage
isisNormalRouter	0	(default) The router operates as a normal router.
isisRestartingRouter	1	The router simulates a restarting router. It sends an IIH containing a Restart TLV with the RR bit set to the neighbor routers.
isisStartingRouter	2	The router simulates a starting router. It sends an IIH containing a Restart TLV with the SA bit set to the neighbor routers.
isisHelperRouter	3	The router acts as the "helper" router for the DUT that is restarting. It acknowledges the Restart TLV sent by the DUT by sending an IIH containing a Restart TLV with the RA bit set.This is only applicable when *hitlessRestartVersion* is 3.

enableHostName

If true, the given dynamic host name is transmitted in all the packets sent from this router.

hostName

Allows to add a host name to this router.

hitlessRestartTime

If *enableHitlessRestart* is *true*, this indicates amount of time that the router will wait for restart completion. *(default = 30)*

hitlessRestartVersion

If *enableHitlessRestart* is *true*, this indicates the version of the *draft-ietf-isis-restart-nn* document that the router should conform to. One of:

Option	Value	Usage
isisDraftVersion3	1	Draft version 3.
isisDraftVersion4	2	(default) Draft version 4.

ignoreRecvAuthentication

If *ignoreRecvAuthentication* is *true*, the ISIS router will not authenticate received packets.

lspLifetime

The maximum age for retaining a learned LSP on this router. *(default = 1,200)*

lspMaxsize

The largest size LSP that this router may generate. *(default = 1,492)*

lspRefreshRate

The rate at which LSPs are resent. *(default = 900)*

maxNumberOf Addresses

The Number of Area Addresses permitted for this IS area. *(default = 3)*

routerId

The ID of the router, usually the lowest IP address on the router. *(default = 0.0.0.0)*

trafficEngineering RouterId

The TE ID of the router, usually the lowest IP address on the router. *(default = 0.0.0.0)*

COMMANDS

The **isisRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisRouter addGrid *gridLocalId*

Adds the grid described in the isisGrid command to the list of grids associated with the router. The grid's entry in the list is given an identifier of *gridLocalId*. Specific errors are:

- The parameters in *isisGrid* are invalid.
- A grid with this *gridLocalId* exists already in the list.

isisRouter addInterface *interfaceLocalId*

Adds the router interface described in the *isisInterface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *interfaceLocalId*. Specific errors are:

- The parameters in *isisInterface* are invalid.
- A router with this *interfaceLocalId* exists already in the list.

isisRouter addRouteRange *routeRangeLocalId*

Adds the route range described in the *isisRouteRange* command to the list of route ranges associated with the router. The range's entry in the list is given an identifier of *routeRangeLocalId*. Specific errors are:

- The parameters in *isisRouteRange* are invalid.
- A router with this *routeRangeLocalId* exists already in the list.

isisRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *isisRouter* command.

isisRouter clearAllGrids

Deletes all of the grids.

isisRouter clearAllInterfaces

Deletes all of the router interfaces.

isisRouter clearAllRouteRanges

Deletes all of the route ranges.

isisRouter config *option value*

Modify the configuration options of the isisRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisRouter.

isisRouter delGrid *gridLocalId*

Deletes the grid with an identifier of *gridLocalId*. Specific errors are:

- No grid with this *gridLocalId* exists in the list.

isisRouter delInterface *interfaceLocalId*

Deletes the router interface with an identifier of *interfaceLocalId*. Specific errors are:

- No router with this *interfaceLocalId* exists in the list.

isisRouter delRouteRange *routeRangeLocalId*

Deletes the route range with an identifier of *routeRangeLocalId*. Specific errors are:

No router with this *routeRangeLocalId* exists in the list.

isisRouter getFirstGrid

Access the first grid in the list. The results may be accessed using the *isisGrid* command. Specific errors are:

- There are no grids in the list.

isisRouter getFirstInterface

Access the first interface in the list. The results may be accessed using the *isisInterface* command. Specific errors are:

- There are no interfaces in the list.

isisRouter getFirstRouteRange

Access the first route range in the list. The results may be accessed using the *isisRouteRange* command. Specific errors are:

There are no route ranges in the list.

isisRouter getGrid *gridLocalId*

Accesses the grid's entry in the list with an identifier of *gridLocalId*. The grid is accessed in the *isisGrid* command. Specific errors are:

- A grid with this *gridLocalId* does not exist in the list.

isisRouter getInterface *interfaceLocalId*

Accesses the interface's entry in the list with an identifier of *interfaceLocalId*. The router interface is accessed in the *isisInterface* command. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

isisRouter getNextGrid

Access the next grid in the list. The results may be accessed using the *isisGrid* command. Specific errors are:

- There is no more grids in the list.

isisRouter getNextInterface

Access the next interface in the list. The results may be accessed using the *isisInterface* command. Specific errors are:

- There is no more interfaces in the list.

isisRouter getNextRouteRange

Access the next route range in the list. The results may be accessed using the *isisRouteRange* command. Specific errors are:

- isisRouter getFirstRouteRange has not been called.
- There is no more route ranges in the list.

isisRouter getRouteRange *routeRangeLocalId*

Accesses the range's entry in the list with an identifier of *routeRangeLocalId*. The router range is accessed in the *isisRouteRange* command. Specific errors are:

- A router with this *routeRangeLocalId* does not exist in the list.

isisRouter setDefault

Sets default values for all configuration options.

isisRouter setGrid *gridLocalId*

Sets the values for the grid's entry in the list with an identifier of *gridLocalId* based on changes made through the *isisGrid* command. This command can be used to change a running configuration and must be followed by an *isisServer write* command in order to send these changes to the protocol server. Specific errors are:

- A grid with this *gridLocalId* does not exist in the list.

isisRouter setInterface *interfaceLocalId*

Sets the values for the interface's entry in the list with an identifier of *interfaceLocalId* based on changes made through the *isisInterface* command. This command can be used to change a running configuration and must be followed by an *isisServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

isisRouter setRouteRange *interfaceLocalId*

Sets the values for the route range's entry in the list with an identifier of *interfaceLocalId* based on changes made through the *isisRouteRange* command. This command should be used to change a running configuration and must be followed by an *isisServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

isisRouter refreshLearnedInformation

This command refreshes the IPv4/IPv6 learned information.

isisRouter getLearnedInformation

This command fetches the IPv4/IPv6 learned information.

isisRouter getFirstLearnedIpv4PrefixesInfo

This command gets the first IPv4 prefixes learned information.

isisRouter getNextLearnedIpv4PrefixesInfo

This command gets the next IPv4 prefixes learned information.

isisRouter getFirstLearnedIpv6PrefixesInfo

This command gets the first IPv6 prefixes learned information.

isisRouter getNextLearnedIpv6PrefixesInfo

This command gets the next IPv6 prefixes learned information.

API

addTopologyRange

This command adds the topology range

delTopologyRange

This command deletes the topology range.

getTopologyRange

This command gets the topology range.

setTopologyRange

This command sets the topology range.

getFirstTopologyRange

This command gets the first topology range.

getNextTopologyRange

This command gets the next topology range

clearAllTopologyRanges

This command clears all topology ranges.

addSpbTopologyRange

This command adds the SPB topology range.

delSpbTopologyRange

This command deletes the SPB topology range.

getSpbTopologyRange

This command gets the SPB topology range.

setSpbTopologyRange

This command sets the SPB topology range.

getFirstSpbTopologyRange

This command sets the first SPB topology range.

getNextSpbTopologyRange

This command gets the next SPB topology range.

clearAllSpbTopologyRanges

This command clears all SPB topology ranges.

addSpbmNetworkRange

This command adds the SPBM network range.

delSpbmNetworkRange

This command deletes the SPBM network range.

getSpbmNetworkRange

This command gets the SPBM network range.

setSpbmNetworkRange

This command sets the SPBM network range.

getFirstSpbmNetworkRange

This command sets the first SPBM network range.

getNextSpbmNetworkRange

This command gets the next SPBM network range.

clearAllSpbmNetworkRanges

This command clears all SPBM network ranges.

NAME - IsisSpbTopologyRange

isisSpbTopologyRange — sets up the parameters associated with an ISIS SPB topology range.

SYNOPSIS

isisSpbTopologyRange *subcommand options*.

DESCRIPTION

The SPBM topology range is added to a particular simulated router via the *isisRouter addSpbTopologyRange* command.

STANDARD OPTIONS

enable true/false

Enables the use of this SPB topology range for the simulated router.

topologyId

It is the ID of the topology.

cistExternalRootCost

It signifies the CIST external root cost.

bridgePriority

It signifies the priority of the bridge value.

spSourceId

It signifies the source ID for SP.

linkMetric

It signifies the link metric value.

numberOfPorts

It signifies the number of ports.

portIdentifier

It signifies the port identifier.

vBit true/false

Enables the bit value.

cistRootIdentifier

It signifies the cist root identifier value.

mcidConfigName

It signifies the config name for MCID.

mcidSignature

It signifies the signature for MCID.

AuxMcidConfigName

It signifies the config name for AUX MCID value.

AuxMcidSignature

It signifies the signature of AUX MCID value.

API

addSpbBaseVidRange

This command adds the SPB base VID range.

delSpbBaseVidRange

This command deletes the SPB base VID range.

getSpbBaseVidRange

This command gets the SPB base VID range.

setSpbBaseVidRange

This command sets the SPB base VID range.

getFirstSpbBaseVidRange

This command sets the first SPB base VID range.

getNextSpbBaseVidRange

This command gets the next SPB base VID range.

clearAllSpbBaseVidRanges

This command clears all SPB base VID ranges.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *IsisSpbBaseVidRange*, *IsisSpbISidRange*

NAME - IsisSpbBaseVidRange

isisSpbBaseVidRange — sets up the parameters associated with an ISIS SPB base VID range.

SYNOPSIS

isisSpbBaseVidRange *subcommand options*.

DESCRIPTION

The SPBM Base Vid range is added to a particular simulated topology via the *isisSpbTopologyRangeaddSpbBaseVidRange* command.

STANDARD OPTIONS

baseVId

It is the base v identifier.

ectAlgorithmType

It signifies the ECT algorithm type.

bMacAddress

It signifies the b MAC address.

useFlagBit

Enables the use of flag bit.

bVlanPriority

It signifies the priority value of B VLAN.

API

addSpbISidRange

This command adds the SPB ISID range.

delSpbISidRange

This command deletes the SPB ISID range.

getSpbISidRange

This command gets the SPB ISID range.

setSpbISidRange

This command sets the SPB ISID range.

getFirstSpbISidRange

This command sets the first SPB ISID range.

getNextSpbISidRange

This command gets the next SPB ISID range.

clearAllSpbISidRanges

This command clears all SPB ISID ranges.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *IsisSpbmNetworkRange*, *IsisSpbmNodeTopologyRange*, *IsisSpbmNodeISidRange*

NAME - IsisSpbISidRange

isisSpbISidRange — sets up the parameters associated with an ISIS SPB ISID range.

SYNOPSIS

isisSpbISidRange *subcommand options*.

DESCRIPTION

The SPBM Is Id range is added to a particular simulated Base VID via the *isisSpbBaseVidRange addSpbIsIdRange* command.

STANDARD OPTIONS

enabled *true/false*

Enables the use of this SPB ISID range for the simulated router.

iSid

It signifies the identifier of IS.

transmissionType

It signifies the transmission type.

tBit

It signifies the t bit value.

rBit

It signifies the r bit value.

vlanType

It signifies the VLAN type.

sVlan

It singnifies the s VLAN value.

cVlan

It signifies the c VLAN value.

cMacAddressCount

It signifies the count value for c MAC address.

startCMacAddress

It starts the c MAC address.

cMacAddressStep

It signifies the c MAC address step value.

iTagEthernetType

It signifies the ethernet type of i tag.

TrafficDestMacAddress

It signifies the taffic destination of MAC address.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *IsisSpbTopologyRange*, *IsisSpbBaseVidRange*

NAME - IsisSpbmNetworkRange

isisSpbmNetworkRange — sets up the parameters associated with an ISIS SPBM network range.

SYNOPSIS

isisSpbmNetworkRange *subcommand options*.

DESCRIPTION

This command allows a grid of IS-IS SPBM routers to be defined. The grid is added to a particular simulated router via the *isisRouter addSpbmNetworkRange* command.

STANDARD OPTIONS

enabled true/false

Enables the use of SPBM network range for the simulated router.

numRows

It signifies the number of rows.

numColumns

It signifies the number of columns.

startSystemId

It starts the system ID.

systemIdIncrementBy

It signifies the value by which the system ID increments.

enableHostName true/false

Enables the host name for the simulated router.

hostNamePrefix

It prefixes the host name.

entryPointRow

It signifies the entry value for point row.

entryPointColumn

It signifies the entry value for point column.

interfaceMetric

It signifies the metric valu of the interface.

API

addSpbmNodeTopologyRange

This command adds the SPBM node topology range.

delSpbmNodeTopologyRange

This command deletes the SPBM node topology range.

getSpbmNodeTopologyRange

This command gets the SPBM node topology range.

setSpbmNodeTopologyRange

This command sets the SPBM node topology range.

getSpbmNodeTopologyRange

This command sets the first SPBM node topology range.

getSpbmNodeTopologyRange

This command gets the next SPBM node topology range.

clearAllSpbmNodeTopologyRanges

This command clears all SPBM node topology ranges.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *IsisSpbmNodeTopologyRange*, *IsisSpbmNodeBaseVidRange*, *IsisSpbmNodeISidRange*

NAME - IsisSpbmNodeTopologyRange

isisSpbmNodeTopologyRange — sets up the parameters associated with an ISIS SPBM node topology range.

SYNOPSIS

isisSpbmNodeTopologyRange *subcommand options*.

DESCRIPTION

This command allows a ISIS SPBM Node topology ranges to be defined. The SPBM Node topology range is added to a particular simulated router via the *isisSpbmNetworkRange addSpbmNodeTopologyRange* command.

STANDARD OPTIONS

enable true/false

Enables the use of this SPBM node topology range for the simulated router.

topologyId

It is the ID of the topology.

cistExternalRootCost

It signifies the CIST external root cost.

bridgePriority

It signifies the priority of the bridge value.

spSourceId

It signifies the source ID for SP.

interNodeSpSourceIdIncrement

It signifies the source identifier increment for inter node SP.

linkMetric

It signifies the link metric value.

interNodeLinkMetricIncrement

It signifies the linc metric increment for inter node.

numberOfPorts

It signifies the number of ports.

portIdentifier

It signifies the port identifier.

vBit true/false

Enables the bit value.

cistRootIdentifier

It signifies the cist root identifier value.

API

addSpbmNodeBaseVidRange

This command adds the SPBM node base VID range.

delSpbmNodeBaseVidRange

This command deletes the SPB node base VID range.

getSpbmNodeBaseVidRange

This command gets the SPB node base VID range.

setSpbmNodeBaseVidRange

This command sets the SPB node base VID range.

getSpbmNodeBaseVidRange

This command sets the first SPB node base VID range.

getNextSpbmNodeBaseVidRange

This command gets the next SPB node base VID range.

clearAllSpbmNodeBaseVidRanges

This command clears all SPB node base VID ranges.

EXAMPLES

See examples under *isisServer*

SEE ALSO

isisServer, *isisRouter*, *IsisSpbmNetworkRange*, *IsisSpbmNodeBaseVidRange*, *IsisSpbmNodeISidRange*

NAME - IsisSpbmNodeBaseVidRange

isisSpbmNodeBaseVidRange — sets up the parameters associated with an ISIS SPBM node base VID range.

SYNOPSIS

isisSpbmNodeBaseVidRange *subcommand options*.

DESCRIPTION

This command allows an ISIS SPBM node Base Vid Range to be defined. The SPBM node Base Vid range is added to a particular simulated topology via the *isisSpbmNodeTopologyRange addSpbmNodeBaseVidRange* command.

STANDARD OPTIONS

baseVId

It is the base v identifier.

ectAlgorithmType

It signifies the ECT algorithm type.

bMacAddress

It signifies the b MAC address.

useFlagBit

Enables the use of flag bit.

bVlanPriority

It signifies the priority value for b VLAN.

API

addSpbmNodeIsidRange

This command adds the SPBM node ISID range.

delSpbmNodeIsidRange

This command deletes the SPBM node ISID range.

getSpbmNodeIsidRange

This command gets the SPBM node ISID range.

setSpbmNodeIsidRange

This command sets the SPBM node ISID range.

getSpbmNodeIsidRange

This command sets the first SPBM node ISID range.

getNextSpbmNodeISidRange

This command gets the next SPBM node ISID range.

clearAllSpbmNodeISidRanges

This command clears all SPBM node ISID ranges.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *IsisSpbmNetworkRange*, *IsisSpbmNodeTopologyRange*, *IsisSpbmNodeISidRange*

NAME - IsisSpbmNodeISidRange

isisSpbmNodeISidRange — sets up the parameters associated with an ISIS SPBM node ISID range.

SYNOPSIS

isisSpbmNodeISidRange *subcommand options*.

DESCRIPTION

This command allows a ISIS SPBM node Is Id Range to be defined. The SPBM node Is Id range is added to a particular simulated node Base VID via the *isisSpbmNodeBaseVidRange addSpbmNodeIsIdRange* command.

STANDARD OPTIONS

enabled true/false

Enables the use of this SPBM node ISID range for the simulated router.

iSid

It signifies the identifier of IS.

interNodeISidIncrement

It signifies the increment of the identifier for inter node IS.

transmissionType

It signifies the transmission type.

tBit

It signifies the t bit value.

rBit

It signifies the r bit value.

vlanType

It signifies the VLAN type.

startCVlan

It starts the c VLAN.

startSVlan

It starts the s VLAN.

cMacAddressCount

It signifies the count value for c MAC address.

startCMacAddress

It starts the c MAC address.

cMacAddressStep

It signifies the c MAC address step value.

iTagEthernetType

It signifies the ethernet type for i tag.

trafficDestMacAddress

It signifies the destination MAC address.

interNodeSvlan

It signifies the inter node value for s VLAN.

interNodeCvlan

It signifies the inter node value for c VLAN.

interNodeCmacAddress

It signifies the inter node value for c MAC address.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *IsisSpbmNetworkRange*, *IsisSpbmNodeTopologyRange*, *IsisSpbmNodeBaseVidRange*

NAME - IsisSpbOutsideLinks

isisSpbOutsideLinks — sets up the parameters associated with an ISIS SPB outside links.

SYNOPSIS

isisSpbOutsideLinks *subcommand options*.

DESCRIPTION

The SPBM outside links is added to a particular simulated router via the *isisSpbmNodeISidRange addSpbOutsideLinks* command.

STANDARD OPTIONS

connectionRow

Signifies to connection to the row.

connectionColmn

Sigfiies the connection to the column.

linkedRouterId

Signifies the router identifier that is linked.

API

addSpbOutsideLinks

This command adds the SPB Outside links.

getSpbOutsideLinks

This command gets the SPB outside links.

setSpbOutsideLinks

This command sets the SPB outside links.

delSpbOutsideLinks

This command deletes the SPB outside links.

getFirstSpbOutsideLinks

This command gets the first SPB outside links.

getNextSpbOutsideLinks

This command gets the next SPB outside links.

clearAllSpbOutsideLinks

This command clears all SPB outside links.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisRouter*, *IsisSpbmNetworkRange*, *IsisSpbmNodeTopologyRange*, *IsisSpbmNodeBaseVidRange*, *IsisSpbmNodeISidRange*

NAME - isisRouteRange

isisRouteRange — sets up the parameters associated with an ISIS route range.

SYNOPSIS

isisRouteRange *subcommand options*

DESCRIPTION

The *isisRouteRange* command describes an individual set of routes. Route ranges are added into *isisRouter* lists using the *isisRouter addRouteRange* command. Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on ISIS testing with Ixia equipment. Refer to *isisRouteRange* for an overview of this command.

STANDARD OPTIONS

enable *true / false*

Enables the use of this route range for the simulated router. *(default = false)*

enableRedistributed true | false

If *true,* the route will be distributed down. If *false*, the route will be redistributed up. *(default = false)*

enableMultiTopology

If true, enables the multi topology option.

ipType

The IP addressing type of the range, one of:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

metric

The cost metric associated with the route. *(default = 0)*

networkIpAddress

The IP address of the routes to be advertised. *(default = 0.0.0.0)*

numberOfNetworks

The number of prefixes to be advertised. *(default = 1)*

prefix

The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0. *(default = 24)*

routeOrigin

The origin of the route, one of:

Option	Value	Usage
isisRouteInternal	0	*(default)* The route originated internally.
isisRouteExternal	1	The route originated externally.

DEPRECATED OPTIONS

extendedDefaultMetric

COMMANDS

The **isisRouteRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisRouteRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisRouteRange** command.

isisRouteRange config *option value*

Modify the configuration options of the isisRouteRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisRouteRange.

isisRouteRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *isisServer*.

SEE ALSO

isisServer, *isisInterface*, *isisRouter*

NAME - isisServer

isisServer — accesses the ISIS component of the protocol server for a particular port.

SYNOPSIS

isisServer *subcommand options*

DESCRIPTION

The *isisServer* command is necessary in order to access the IS-IS protocol server for a particular port. The *select* subcommand **must** be used before all other IS-IS commands. Refer to *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on ISIS testing with Ixia equipment. Refer to *isisServer* for an overview.

STANDARD OPTIONS

emulationType

It signifies the emulation type of the router.

- spbIsis

spbHelloMulticastMac

It signifies the hello multicast MAC value.

spbAllL1BridgesMac

It signifies the all L1 bridges MAc value.

spbNlpId

It signifies the n IP ID value.

COMMANDS

The **isisServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisServer addRouter *routerLocalId*

Adds the IS-IS router described in the *isisRouter* command to the list of routers associated with the port. The router's entry in the list is given an identifier of *routerLocalId*. Specific errors are:

- isisServer select has not been called.
- The router parameters in *isisRouter* are invalid.
- A router with this *routerLocalId* exists already in the list.

isisServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **isisServer** command.

isisServer clearAllRouters

Deletes all the ISIS routers in the list. Specific errors are:

- isisServer select has not been called.
- There is no router with this *routerLocalId* in the list.

isisServer config *option value*

Modify the configuration options of the isisServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for isisServer.

isisServer delRouter *routerLocalId*

Deletes the ISIS router described that has an identifier of *routerLocalId*. Specific errors are:

- isisServer select has not been called.
- There is no router with this *routerLocalId* in the list.

isisServer generateStreams *chasID cardID portID action*

Generate streams creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each enabled route range associated with each ISIS router; each stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

isisServer getFirstRouter

Access the first ISIS router in the list. The results may be accessed using the *isisRouter* command. Specific errors are:

- isisServer select has not been called.
- There are no routers in the list.

isisServer getNextRouter

Access the next ISIS router in the list*.* The results may be accessed using the *isisRouter* command. Specific errors are:

- isisServer select has not been called.
- isisServer getFirstRouter has not been called.
- There is no more routers in the list.

isisServer getRouter *routerLocalId*

Access the ISIS router with an identifier of *routerLocalId.* The results may be accessed using the *isisRouter* command. Specific errors are:

- isisServer select has not been called.
- There is no router with this *routerLocalId* in the list.

isisServer select *chasID cardID portID*

Accesses the ISIS component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The ISIS protocol package has not been installed.
- Invalid port specified.

isisServer setRouter *routerLocalId*

Sets the values for the router's entry in the list with an identifier of *routerLocalId* based on changes made through the *isisRouter* command. This command should be used to change a running configuration and must be followed by an *isisServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *routerLocalId* does not exist in the list.

isisServer write

Sends any changes made with *isisRouter setInterface, isisRouter setRouteRange* or *isisServer setRouter* to the protocol server for immediate application. This command **must** be used after those mentioned above in order for their changes to have an effect.

EXAMPLES

```
package req IxTclHal

# Define parameters used by ISIS router

set host localhost
set username user
# Check if we’re running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts “Could not connect to $host”
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set ch [ixGetChassisID $host]
# Port is: card 4, port 1
set ca 4
set po 1
set pl [list [list $ch $ca $po]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl] {
ixPuts $::ixErrorInfo
return 1
}
set myMac {00 0a de 01 01 01}
set router 101.101.12.2
set neighbor 101.101.12.1
set interfaceIpMask 255.255.255.0
set numberOfRoute 1650
# Set up the interface table for an IPv4 and IPv6 interface
# on the port
interfaceTable select $ch $ca $po
interfaceTable clearAllInterfaces
interfaceIpV6 setDefault
interfaceIpV6 config -ipAddress
{0:0:0:0:0:0:0:1}
interfaceIpV6 config -maskWidth 64
interfaceEntry addItem addressTypeIpV6
interfaceIpV4 setDefault
interfaceIpV4 config -ipAddress
{192.168.1.2}
interfaceIpV4 config -gatewayIpAddress
{192.168.1.1}
interfaceIpV4 config -maskWidth 24
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable
true
interfaceEntry config -description
{Port 04:01 Interface}
interfaceEntry config -macAddress {00
00 04 90 25 c6}
interfaceTable addInterface
interfaceTable write
# Select port to operate
isisServer select $ch $ca $po
# Clear all routers
isisServer clearAllRouters
# Add the isis interface to the router
if [isisRouter addInterface interface1] {
logMsg "Error in adding isisInterface interface1"
}
```

```
# Configure the routeRange
isisRouteRange setDefault
isisRouteRange config -enable true
isisRouteRange config -metric 1
isisRouteRange config -numberOfNetworks $numberOfRoute
isisRouteRange config -prefix 24
isisRouteRange config -ipType addressTypeIpV4
isisRouteRange config -networkIpAddress {14.0.0.0}
# Add the isis routeRange to the router
if [isisRouter addRouteRange routeRange1] {
logMsg "Error in adding routeRange"
}
# Configure isis router
isisRouter setDefault
isisRouter config -routerId "00 0$ca 0$po 01 00 00"
isisRouter config -enable true
isisRouter config -areaAddressList {49 00 01}
# Add the router to the server
if [isisServer addRouter router1] {
logMsg "Error in adding router"
}
# Let the protocol server respond to ARP, ISIS
protocolServer config -enableArpResponse true
protocolServer config -enableIsisService true
protocolServer config -enablePingResponse false
protocolServer set $ch $ca $po
# Send the data to the hardware
logMsg "Writing the configuration to the hardware"
ixWriteConfigToHardware pl
# And start ISIS on the port
logMsg "Start isis server ..."
ixStartIsis pl
# Disable routeRange1 while isis server is runnung.
# This is the same as removing the route range from router
```

```
isisServer select $ch $ca $po
if [isisServer getRouter router1] {
logMsg "Error getting router1"
}
if [isisRouter getRouteRange routeRange1] {
logMsg "Error getting routeRange1"
}
# Disable the route range
# (You can also change other configuration if you want)
isisRouteRange config -enable false
if [isisRouter setRouteRange routeRange1] {
logMsg "Error setting routeRange1"
}
if [isisServer write] {
logMsg "Error writing isisServer"
}
after 10000
# Stop the server at the end
logMsg "Stop isis server ..."
ixStopIsis pl
# If you wanted to add a route range while isis server is running,
# -Configure it disabled before starting isis server and then
# enable it
# The same thing is also possible on isisInterface.
# You just need to get the item that you want and change the
configuration
# and set that item. Then write the changes to hardware by
isisServ
# Let go of the ports that we reserved
ixClearOwnership $pl
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
```

```
ixDisconnectTclServer $host

}
```

SEE ALSO

isisInterface, isisRouter, isisRouteRange

NAME - isisLearnedIpv4Prefixes

isisLearnedIpv4Prefixes — This command is used to fetch the learned information of L3 ISIS IPv4 prefixes.

SYNOPSIS

isisLearnedIpv4Prefixes*subcommand options*

DESCRIPTION

This command is used to fetch the learned information of L3 ISIS IPv4 prefixes.

STANDARD OPTIONS

lspId

The LSP number of the IPv4 prefix.

sequenceNumber

Sequence number of the LSP containing the route.

age

The age in time since last refreshed.

hostName

The host name as retrieved from the related packets.

metric

The route metric.

ipv4Prefix

The mask width of the IPv4 Prefix.

COMMANDS

The **isisLearnedIpv4Prefixes** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisLearnedIpv4Prefixes cget

This command fetches the IPv4 prefixes learned information.

Example

```
isisLearnedIpv4Prefixes cget -lspId

isisLearnedIpv4Prefixes cget -sequenceNumber

isisLearnedIpv4Prefixes cget -age

isisLearnedIpv4Prefixes cget – metric
```

```
  isisLearnedIpv4Prefixes cget -ipv4Prefix
```

SEE ALSO

isisRouter

NAME - isisLearnedIpv6Prefixes

isisLearnedIpv6Prefixes — This command is used to fetch the learned information of L3 ISIS IPv6 prefixes.

SYNOPSIS

isisLearnedIpv6Prefixes *subcommand options*

DESCRIPTION

This command is used to fetch the learned information of L3 ISIS IPv6 prefixes.

STANDARD OPTIONS

lspId

The LSP number of the IPv6 prefix.

sequenceNumber

Sequence number of the LSP containing the route.

age

The age in time since last refreshed.

hostName

The host name as retrieved from the related packets.

metric

The route metric.

ipv6Prefix

The mask width of the IPv6 Prefix.

COMMANDS

The **isisLearnedIpv6Prefixes** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

isisLearnedIpv6Prefixes cget

This command fetches the IPv6 prefixes learned information.

Example

```
isisLearnedIpv6Prefixes cget -lspId4

isisLearnedIpv6Prefixes cget -sequenceNumber5

isisLearnedIpv6Prefixes cget -age6

isisLearnedIpv6Prefixes cget metric7
```

```
  isisLearnedIpv6Prefixes cget ipv6Prefix8
```

SEE ALSO

isisRouter

NAME - isisDceTopologyRange

isisDceTopologyRange Sets the Network Range for a particular DCE ISIS Topology.

SYNOPSIS

isisDceTopologyRange *subcommand options*

DESCRIPTION

Sets the Network Range for a particular DCE ISIS Topology.

STANDARD OPTIONS

enable

Signifies if DCE Topology is enabled or disabled.

enableFtag

If true, the F tag is enabled.

nicknameList

The list of nicknames.

- nickname
- nicknamePriority
- broadcastrootPriority

noOfTreesToCompute

The number of trees to compute.

startFtagValue

If true, the Ftag value is started.

topologyCount

The count of the topology.

topologyId

The unique identification number of the topology range.

topologyIdStep

It shows the Increment Step of the ID of DCE Topology Range. Default is 1.

Example

SEE ALSO

isisInterface, *isisRouter*, *isisRouteRange*

NAME - IsisDceNodeTopologyRange

isisdceNodeTopologyRange Sets the DCE Node Topology of a particular DCE ISIS Topology Range.

SYNOPSIS

isisDceNodeTopologyRange*subcommand options*

DESCRIPTION

Sets the DCE Node Topology of a particular DCE ISIS Topology Range.

STANDARD OPTIONS

startNickname

If true, uses the nickname.

nicknameCount

The count of the nickname.

internodeNicknameIncrement

The increment step to be used for creating the internode increment.

noOfTreesToCompute

The number of trees to compute.

topologyId

The unique identification number of the topology range

topologyCount

The count of the topology.

broadcastPriority

Sets the priority in which the topology is broadcast.

includeL2Topology

If true, includes the L2 topology.

startVlanId

The VLAN Id of first VLAN. Default is 1.

vlanCount

The count of the VLAN.

includeInMGROUPPDU

If true, a custom VLAN is included in the MGROUP PDU.

includeInLSP

If true, a custom VLAN is included in the LSP.

noOfSpanningTreeRoots

The number of spanning tree roots for the VLAN.

startSpanningTreeRootBridgeId

If true, starts the spanning tree root Bridge Id.

m4BitEnabled

If true, the M4 bit is enabled.

m6BitEnabled

If true, the M6 bit is enabled.

omBitEnabled

If true, the om bit is enabled.

enabled

Signifies if DCE Interested Vlan range is enabled or disabled.

EXAMPLE

API

addDceNodeInterestedVlanRange

This command adds the Dce Node Interested Vlan range

delDceNodeInterestedVlanRange

This command deletes the Dce Node Interested Vlan range.

getDceNodeInterestedVlanRange

This command gets the Dce Node Interested Vlan range.

setDceNodeInterestedVlanRange

This command sets the Dce Node Interested Vlan range.

getFirstDceNodeInterestedVlanRange

This command gets the first Dce Node Interested Vlan range.

getNextDceNodeInterestedVlanRange

This command gets the next Dce Node Interested Vlanrange

clearAllDceNodeInterestedVlanRange

This command clears all Dce Node Interested Vlan ranges.

SEE ALSO

isisInterface, *isisRouter*, *isisRouteRange*

NAME - lacpLearnedInfo

lacpLearnedInfo — views retrieved Learned LACP information.

SYNOPSIS

lacpLearnedInfo *subcommand options*

DESCRIPTION

The *lacpLearnedInfo* command is used to look at the information retrieved when using the *lacpLink* and *lacpServer* commands.

The optional LACP test package must be installed in order for this command to operate

STANDARD OPTIONS

actorSystemId

(read only) The learned Actor system identifier, in 6 byte format.

actorSystemPriority

(read only) The learned Actor system priority, in hexadecimal format.

actorPortNumber

(read only) The learned Actor port number in hexadecimal format.

actorPortPriority

(read only) The learned Actor port priority, in hexadecimal format.

actorOperationalKey

(read only) The learned Actor operation key, in hexadecimal format.

administrativeKey

(read only) This field controls the aggregation of ports of the same system with similar Actor Key.

partnerSystemId

(read only) The learned Partner system identifier, in 6 byte format.

otherLagMemberCount

(read only) The total number of ports,excluding the individual port that are a part of the LAG.

otherLagMemberDetails

(read only) The detailed information of the other member ports of the same LAG, visible in card:port format.

partnerPortNumber

(read only) The learned Partner port number in hexadecimal format.

partnerOperationalKey

(read only) The learned Partner operation key, in hexadecimal format.

partnerPortPriority

(read only) The learned Partner port priority, in hexadecimal format.

partnerSystemPriority

(read only) The learned Partner system priority, in hexadecimal format.

partnerCollector MaxDelay

(read only) The learned maximum Collection Delay for the port, in microseconds.

enabledAggregation true / false

(read only) Learned Link Aggregation status of the link, whether Aggregated or Not Aggregated.

partnerLacpActivity

(read only) The learned Partner LACP activity mode, either Passive or Active.

partnerLink AggregationStatus

(read only) The le arned aggregatability status of the partner, whether Aggregatable or Individual.

partnerDistributing Flag true / false

(read only) The learned Partner Distributing Flag status, either True of False. If True, the Distributing Flag is enabled.

partnerLacpTimeout

(read only) The learned Partner LACPDU timeout mode, either Long or Short.

partnerSyncFlag true / false

(read only) The learned Partner synchronized status, either OUT_OF_SYNC/IN_SYNC.

partnerCollectingFlag true / false

(read only) The learned Partner Collecting Flag status, either True of False. If True, the Collecting Flag is enabled.

partnerDefaultedFlag true / false

(read only) The learned Partner Defaulted Flag status, either True of False. If True, the Defaulted Flag is enabled.

partnerExpiredFlag true / false

(read only) The learned Partner Expired Flag status, either True of False. If True, the Expired Flag is enabled.

actorLacpActivity

(read only) The learned Actor LACP activity mode, either Passive or Active.

actorLinkAggregation Status

(read only) The aggregatability status of the actor, whether Aggregatable or Individual.

actorDistributingFlag true / false

(read only) The learned Actor Distributing Flag status, either True of False. If True, the Distributing Flag is enabled.

actorLacpTimeout

(read only) The learned Actor LACPDU timeout mode, either Long or Short.

actorSyncFlag true / false

(read only) The learned Actor synchronized status, either OUT_OF_SYNC/IN_SYNC.

actorCollectingFlag true / false

(read only) The learned Actor Collecting Flag status, either True of False. If True, the Collecting Flag is enabled.

actorDefaultedFlag true / false

(read only) The learned Actor Defaulted Flag status, either True of False. If True, the Defaulted Flag is enabled.

actorExpiredFlag true / false

(read only) The learned Actor Expired Flag status, either True of False. If True, the Expired Flag is enabled.

COMMANDS

The *lacpLearnedInfo* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

lacpLearnedInfo setDefault

Sets default values for all of the options.

EXAMPLES

SEE ALSO

lacpLink, *lacpServer*

NAME - lacpLink

lacpLink — contains the configuration information for a IxNetwork LACP link.

SYNOPSIS

lacpLink *subcommand options*

DESCRIPTION

The *lacpLink* command is used to configure an LACP link.

The optional LACP test package must be installed in order for this command to operate.

STANDARD OPTIONS

enabled true/ false

If true, this particular LACP link entry is enabled.

actorSystemId

Specifies the system identifier for the link Actor. It is a 6 byte field, with a default of 00-00-00-00-00-01.

Min: 00-00-00-00-00-00

Max: FF-FF-FF-FF-FF-FF

actorSystemPriority

Specifies the system priority of the link Actor. It is a 2 byte field, with a default or 1.

Min: 0

Max: 65535

actorPortNumber

The port number assigned to the port by the Actor (the System sending the PDU). It is a 2 byte field with a default of 1.

Min: 0

Max: 65535

autoPickPortMac true/ false

If true, the source MAC is the interface MAC address. It is enabled by default.

portMac

Specifies the port MAC address. This option is grayed out if Auto Pick Port MAC is selected. This is a 6 byte field.

actorPortPriority

Specifies the port priority of the link Actor. It is a 2 byte field, with a default or 1.

Min: 0

Max: 65535

administrativeKey

Controls the aggregation of ports of the same system with similar Actor Key.

actorKey

The operational Key value assigned to the port by the Actor. This is a 2 byte field with a default of 1.

Min: 0

Max: 65535

enablePreservePartnerInfo

If true, and a new link of the same port is enabled, transmitted LACPDUs carry the information for learned partner. If false, the LACPDUs carry partner information as 0.

collectorMaxDelay

The maximum time in microseconds that the Frame Collector may delay the delivery of a frame received from an Aggregator to its MAC client. This is a 2 byte field with a default 0.

Min: 0

Max: 65535

lacpduPeriodic TimeInterval

Defines how frequently LACPDUs are sent to the link partner. One of:

Option	Value	Usage
fastInterval	0	1 second
slowInterval	1	30 second
autoInterval	2	0 second (default).
defaultInterval		Sets the LACPDU interval to the default setting.

lacpTimeout

This timer is used to detect whether received protocol information has expired. One of:

Option	Value	Usage
shortTimeOut	0	3 seconds
longTimeOut	1	90 seconds
autoTimeout	2	0 (default).
defaultTimeOut	3	The timeout value from received LACPDU and till any-LACPDU is received.

lacpActivity

Sets the value of LACPs Actor activity, either passive or active. One of:

Option	Value	Usage
active		Active LACP indicates the port's preference to participate in the protocol regardless of the Partner's control value. (default)
passive		Passive LACP indicates the port's preference for not transmitting LACPDUs unless its Partner's control is Active.
defaultActivity		Sets the value to the default option.

supportResponding ToMarker true/ false

When true, LACP doesn't respond to MARKER request PDUs from the partner.

sendPeriodicMarker Request true/ false

When true, periodic Marker Request PDUs are sent after both actor and partner are IN SYNC and our state is aggregated. The moment we come out of this state, the periodic sending of Marker will be stopped.

markerRequestMode

Sets the marker request mode for the Actor link. One of:

Option	Value	Usage
fixedMode		The inter-marker PDU interval has to be provided as a single number. (default)
randomMode		The interval can be specified as a range.
defaultRequestMode		Sets the value to the default option.

interMarkerPDUDelay

Sets the value for the Marker Request Mode, in seconds.

If the Marker Request Mode is Fixed, then enter a single number from 0 to 255 (default 6). If Marker Request Mode is Random, then enter a number range (each endpoint from 0 to 255, with a default of 0 - 15).

sendMarkerRequestOn LagChange true/ false

If true, causes LACP to send a Marker PDU on the following situations:

- System Priority has been modified
- System Id has been modified
- Actor Key has been modified

Port Number/Port Priority has been modified while we are in Individual mode.

aggregationFlagState

If true, sets the port status to allow aggregation.

Default: true.

syncFlag

If true, the actor port state is set to True based on Tx and Rx state machines. Otherwise, the flag in LACPDU remains reset for all packets sent.

One of:

Option	Value	Usage
disableFlag		not enabled
enableFlag		enabled (default)
defaultFlag		Sets the value to the default option.

markerResponseWait Time

The number of seconds to wait for Marker Response after sending a Marker Request. After this time, the Marker Response Timeout Count is incremented.

If a marker response does arrive for the request after this timeout, it is not considered as a legitimate response.

distributingFlag

If true, the actor port state Distributing is set to true based on Tx and Rx state machines. Otherwise, the flag in LACPDU remains reset for all packets sent. One of:

Option	Value	Usage
disableFlag		not enabled
enableFlag		enabled (default)
defaultFlag		Sets the value to the default option.

collectingFlag

If true, the actor port state Collecting is set to true based on Tx and Rx state machines. Otherwise, the flag in LACPDU remains reset for all packets sent. One of:

Option	Value	Usage
disableFlag		not enabled
enableFlag		enabled (default)
defaultFlag		Sets the value to the default option.

name

Sets a text string as the name of the link.

COMMANDS

The *lacpLink* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

lacpLink setDefault

Sets default values for all of the options.

EXAMPLES

SEE ALSO

lacpLearnedInfo, *lacpServer*

NAME - lacpServer

lacpServer — accesses the LACP component of the protocol server for a particular port.

SYNOPSIS

lacpServer *subcommand options*

DESCRIPTION

The *lacpServer* command is necessary in order to access the LACP component of the protocol server for a particular port. The *select* subcommand **must** be used before all other LACP commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on LACP testing with Ixia equipment. The optional LACP test package must be installed in order for this command to operate.

STANDARD OPTIONS

NONE

COMMANDS

The *lacpServer* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

lacpServer select *chasID cardID portID*

Accesses the LACP component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- Invalid port specified.

lacpServer addLink *linkName*

Adds a link to the list of LACP links at the current position. Specific errors are:

- A port has not been selected via the *lacpServerselect* command.
- The port is owned by another user.
- An object with this ID has already been added.

lacpServer delLink *inkName*

Removes a link from the list of LACP links at the current position. Specific errors are:

- A port has not been selected via the *lacpServer select* command.
- The port is owned by another user.
- An object with this ID has already been added.

lacpServer getLink *linkName*

Finds the link indicated by the *linkName*, sets it to the `current' link and retrieves the data so that it can be viewed and modified. Specific errors are:

- A port has not been selected via the *lacpServer select* command.
- There is no neighbor with this ID.

lacpServer setLink *linkName*

Replaces the data associated with a link, either the link with the indicated *linkName* or the currently selected neighbor, if the *linkName* argument is omitted. Specific errors are:

- A port has not been selected via the *lacpServer select* command.
- The port is owned by another user.
- Invalid neighbor configuration.
- There is no neighbor with this ID.

lacpServer getFirstLink

Makes the first link in the list the `current' link and retrieves the data. Specific errors are:

- A port has not been selected via the *lacpServer select* command.
- The list is empty.

lacpServer getNextLink

Makes the next link in the list the `current' link and retrieves the data. Specific errors are:

- A port has not been selected via the *lacpServerselect* command.
- There are no more objects in the list.

lacpServer clearAllLinks

Clears all the links in the list.

lacpServer write

Writes or commits the changes in IxHAL to hardware for the LACP related parameters on the port selected with the *select* subcommand*.* Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- Network problem between the client and chassis.

lacpServer sendUpdate

This command sends an update to the link partners after changing a link's configuration parameters.

lacpServer sendMarkerRequest

Sends Marker Requests at will. The contents of the marker PDU contain the current view of partner (which can be defaulted if no partner is present). The marker will be sent regardless of which state the link is in.

lacpServer startPDU

Used to start PDUs related to LACP (for example, LACPDU, Marker Request PDU, Marker Response PDU) while the protocol is running on the port.

lacpServer stopPDU

Used to stop PDUs related to LACP (for example, LACPDU, Marker Request PDU, Marker Response PDU) while the protocol is running on the port.

lacpServer requestLacpLearnedInfo

Requests learned information from the link partners.

lacpServer getLacpLearnedInfo

Retrieves the learned LACP information after a *requestLacpLearnedInfo* command.

lacpServer setDefault

Sets default values for all configuration options.

EXAMPLES

SEE ALSO

lacpLearnedInfo, *lacpLink*

NAME - mplsOamServer

mplsOamServer — configures the server options for the port for a MPLS OAM server.

SYNOPSIS

mplsOamServer *subcommand options*

DESCRIPTION

The *mplsOamServer* command is necessary in order to access the mpls Oam component of the protocol server for a particular port. The *select* subcommand **must** be used before all other mpls Oam commands. The optional Mpls Oam test package must be installed in order for this command to operate.

STANDARD OPTIONS

NONE

COMMANDS

The *mplsOamServer* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsOamServer *select*

This is used to select the chassis, card, and port to operate on.

mplsOamServer *getRouter*

This is used to get the mplsOamRouter instance from the mplsOamServer.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamServer *setRouter*

This is used to save/set a particular mplsOamRouter instance to the mplsOamServer. Only one router is supported at this time.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamServer *getfirstRouter*

This is used to get the first instance from the list of mplsOamRouter instances configured under mplsOamServer.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamServer *getnextRouter*

This is used to get the next instance from the list of mplsOamRouter instances configured under mplsOamServer.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamServer *delRouter*

This is used to delete the mplsOamRouter instance from the mplsOamServer.

mplsOamServer *delRouter*

This is used to delete the mplsOamRouter instance from the mplsOamServer.

mplsOamServer *write*

This is used to send any changes made with mplsOamServer setRouter, or mplsOamServer getRouter to the protocol server for immediate application.

mplsOamServer *showRouteNames*

This is used to display the names of all links, configured under mplsOamServer.

Note: At present, this shows the name of only one router.

EXAMPLES

SEE ALSO

NAME - mplsOamServer

NAME - mplsOamRouter

mplsOamRouter — configures the server options for the port for a MPLS OAM router.

SYNOPSIS

mplsOamRouter *subcommand options*

DESCRIPTION

The *mplsOamRouter* command is necessary in order to access the Mpls Oam component of the protocol server for a particular port.

STANDARD OPTIONS

NONE

COMMANDS

The *mplsOamRouter* command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

mplsOamRouter *getInterface*

This is used to get a particular instance of mplsOamInterface from the mplsOamRouter.

- Arguments: interfaceName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamRouter *setInterface*

This is used to save/set a particular instance of mplsOamInterface to the mplsOamRouter.

- Arguments: linkName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamRouter *getFirstInterface*

This is used to get the first instance from the list of mplsOamInterfaces configured under mplsOamRouter.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamRouter *getiNextInterface*

This is used to get the next instance from the list of mplsOamInterfaces configured under mplsOamRouter. This should be called after a call to the getFirstInterface.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamRouter *delInterface*

This is used to remove a particular mplsOamInterface instance from the mplsOamRouter object.

mplsOamRouter *set*

This is used to set the current configuration of the protocol server on the most recently selected port to its hardware.

mplsOamRouter *get*

This is used to get the current mplsOamRouter configuration.

mplsOamRouter *enabled*

If true, the mplsOamRouter object is enabled.

mplsOamRouter *routerId*

The assigned router ID.

SEE ALSO

NAME - mplsOamRouter

NAME - mplsOamInterface

mplsOamInterface — holds the information related to a single interface on the simulated router.

SYNOPSIS

mplsOamInterface *subcommand options*

DESCRIPTION

The *mplsOamInterface* holds the information related to a single interface on the simulated router.

STANDARD OPTIONS

enabled

If true, it enables or disables the simulated router.

enablePeriodicPing

If true, it enables periodic ping.

enableFecValidation

If true, it enables FEC validation.

enableIncludePadTlv

If true, it includes Pad TLV.

enableIncludeVendorEnterpriseNumberTlv

If true, it includes vendor enterprise number TLV.

enableDownStreamMappingTlv

If true, it downstreams mapping TLV.

enableDsIflag

If true, it enables DSI flag.

enableDsNflag

If true, it enables DSN flag.

replyMode

It signifies the reply mode. It may contain any of the values:

- doNotReply
- replyViaIpv4Ipv6UdpPacket

- replyViaIpv4Ipv6UdpPacketWithRouterAlert
- replyViaApplicationLevelControlChannel

padTlvFirstOctet

It signifies the first octate of Pad TLV. It may contain any of the values:

- dropPadTlvFromReply
- copyPadTlvToReply

controlChannel

It signifies the control channel. It may contain any of the values:

- controlChannelRouterAlert
- controlChannelPwAch

bfdCvType

It signifies the BFD CV type. It may contain any of the values:

- bfdCvTypeIpUdp
- bfdCvTypePwAch

downStreamAddressType

It downstreams address type. It may contain any of the values:

- ipv4NumberedDownStreamAddressType
- ipv4UnNumberedDownStreamAddressType

padTlvLength

It signifies the length of the Pad TLV.

vendorEnterpriseNumber

It signifies the vendor enterprise number.

echoResponseTimeout

It signifies the timeout of echo response.

echoRequestInterval

It signifies the interval of echo request.

minRxInterval

It signifies the minimum interval between receptions.

txInterval

It signifies the transmission interval.

multiplier

It signifies the multiplier value.

flapTxIntervals

It signifies the flap transmission intervals.

bfdDiscriminatorStart

It starts the BFD discriminator.

bfdDiscriminatorEnd

It ends the BFD discriminator.

downStreamInterfaceAddress

It downstreams interface address.

interfaces

It signifies the interface.

destinationIpv4Address

It signifies the destination of IPv4 address.

downStreamIpAddress

It downstreams IP address.

COMMANDS

The *mplsOamInterface* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsOamInterface *set*

This is used to save/set a particular instance of mplsOamInterface to the mplsOamServer.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamInterface *get*

This is used to get a particular instance of mplsOamInterface from the mplsOamServer.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

mplsOamInterface *destination Ipv4Address*

The allocated destination IPv4 address for this interfaceThe allocated destination IPv4 address for this interface

mplsOamInterface *echoRequestInterval*

The minimum interval, in milliseconds, between received Echo packets that this interface is capable of supporting.

mplsOamInterface *echoResponseTimeout*

The minimum tiomeout interval, in milliseconds, between received Echo packets that this interface is capable of supporting.

mplsOamInterface *enableFecValidation*

If true, the check box is selected to enable validation.

mplsOamInterface *enablePeriodicPing*

If true, the router is pinged at regular intervals.

mplsOamInterface *enabled*

If true, it enables or disables the simulated router.

mplsOamInterface *flapTxIntervals*

The number of seconds between route flaps for MPLS OAM. A value of zero means no flapping.

mplsOamInterface *enableIncludePadTlv*

If true, selects the check box to include Pad TLV.

mplsOamInterface *enableIncludeVendorEnterpriseNumberTlv*

If true, selects the checkbox to include the the TLV number of the vendor organization.

mplsOamInterface *minRxInterval*

The minimum interval, in milliseconds, between received BFD Control packets that this interface is capable of supporting.

mplsOamInterface *multiplier*

The negotiated transmit interval, multiplied by this value, provides the detection time for the interface.

mplsOamInterface *padTlvFirstOctet*

Select the first octate of the Pad TLV.

mplsOamInterface *padTlvLength*

Specifies the length of the Pad TLV.

mplsOamInterface *replyMode*

Selects the mode of reply.

mplsOamInterface *txInterval*

The minimum interval, in milliseconds, that the interface would like to use when transmitting Control packets.

mplsOamInterface *vendorEnterpriseNumber*

Specifies the enterprise number of the vendor.

EXAMPLES

SEE ALSO

NAME - mplsOamInterface

NAME - bgp4VpnL2Site

bgp4VpnL2Site — holds information about a VPN Layer 2 site.

SYNOPSIS

bgp4VpnL2Site *subcommand options*

DESCRIPTION

The *bgp4VpnL2Site* holds information about a VPN Layer 2 site.

STANDARD OPTIONS

NONE

COMMANDS

The *bgp4VpnL2Site* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

bgp4VpnL2Site *enableBfdVccv*

If true, enables the use of BFD VCCV. (default = false).

bgp4VpnL2Site *enableVccvPing*

If true, enables VCCV ping. (default = false).

EXAMPLES

SEE ALSO

NAME - bgp4VpnL2Site

NAME - ldpServer

ldpServer — accesses the LDP component of the protocol server for a particular port.

SYNOPSIS

ldpServer *subcommand options*

DESCRIPTION

The ldpServer accesses the LDP component of the protocol server for a particular port.

STANDARD OPTIONS

NONE

COMMANDS

The *ldpServer* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpServer *enableUseTransportLabelsForMplsOam*

If true, enables label exchange over LSP.

EXAMPLES

SEE ALSO

ldpServer

NAME - rsvpServer

rsvpServer — accesses the RSVP component of the protocol server for a particular port.

SYNOPSIS

rsvpServer *subcommand options*

DESCRIPTION

The *rsvpServer* accesses the RSVP component of the protocol server for a particular port.

STANDARD OPTIONS

NONE

COMMANDS

The *rsvpServer* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpServer *enableUseTransportLabelsForMplsOam*

If true, enables label exchange over LSP.

EXAMPLES

SEE ALSO

rsvpServer

NAME - mplsOamGeneralLearnedInfo

mplsOamGeneralLearnedInfo — holds the lists of the general learned route information.

SYNOPSIS

mplsOamGeneralLearnedInfo *subcommand options*

DESCRIPTION

The *mplsOamGeneralLearnedInfo* holds the lists of the general learned route information.

STANDARD OPTIONS

incomingLabelStack

It signifies the incoming value of label stack.

outgoingLabelStack

It signifies the outgoing value of label stack.

fec

It signifies forwarding equivalence class.

myBfdSessionState

It signifies BFD of the session state for the interface.

peerBfdSessionState

It signifies the BFD session state for the far side.

lspPingReachability

It signifies the LSP ping reachability.

returnCode

It signifies the return code value.

myIpAddress

The IP address for this interface.

peerIpAddress

The learnt IP address for the session.

receivedPeerFlags

It signifies the number of received peer flags.

minRtt

It signifies the minimum Round-Trip Time.

maxRtt

It signifies the maximum Round-Trip Time.

averageRtt

It signifies the average Round-Trip Time.

signalingProtocol

It signifies the signalling protocol.

ccInUse

It signifies the CC in use.

cvInUse

It signifies the CV in use.

tunnelType

It signifies the type of tunnel value.

tunnelEndpointType

It signifies the type of endpoint tunnel value.

incomingPwLabel

It signifies the incoming PW label value.

outgoingPwLabel

It signifies the outgoing PW label value.

incomingLspLabel

It signifies the incoming LSP label value.

outgoingLspLabel

It signifies the outgoing LSP label value.

pingAttempts

It signifies the number of ping attempts made.

pingSuccess

It signifies the number of successful ping attempts made.

pingFailures

It signifies the number of failed ping attempts made.

pingRequestRx

It signifies the number of ping requests received.

pingReplyTx

It signifies the ping reply transmitted.

returnSubCode

It signifies the return value of sub code.

myDiscriminator

The discriminator for the session on this interface.

peerDiscriminator

The discriminator for the far side of the session.

receivedMinRxInterval

The minimum receive interval, in milliseconds, for the far side of the session.

receivedTxInterval

The minimum transmit interval, in milliseconds, for the far side of the session.

receivedMultiplier

The number of received negotiated transmit intervals when multiplied by this value, provides the detection time for the interface.

COMMANDS

The *mplsOamGeneralLearnedInfo* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsOamGeneralLearnedInfo *cget* -averageRtt

The learned average MPLS OAM Round-Trip-Time.

mplsOamGeneralLearnedInfo *cget* -fec

(read only) Forwarding equivalence class (FEC) type.

mplsOamGeneralLearnedInfo *cget* -incomingLabelStack

(read only) BGP sends the assigned labels information to this MPLS OAM module which is used for validation of FEC stack received in an echo request.

mplsOamGeneralLearnedInfo *cget* -incomingLspLabel

(read only) The incoming LSP label value.

mplsOamGeneralLearnedInfo *cget* -incomingPwLabel

(read only) The incoming PW label value.

mplsOamGeneralLearnedInfo *cget* -lspPingReachability

(read only) Specifies whether the queried LSP Ping could be reached or not.

mplsOamGeneralLearnedInfo *cget* -maxRtt

(read only) Specifies the maximum Round Trip Time.

mplsOamGeneralLearnedInfo *cget* -minRtt

(read only) Specifies the minimum Round Trip Time.

mplsOamGeneralLearnedInfo *cget* -bfdSessionMyState

(read only) This window provides read-only information about the state of BFD interface on the specified emulated router.

mplsOamGeneralLearnedInfo *cget* -myDiscriminator

(read only) The discriminator for the session on this interface.

mplsOamGeneralLearnedInfo *cget* -myIpAddress

(read only) The IP address for this interface.

mplsOamGeneralLearnedInfo *cget* -outgoingLabelStack

(read only) BGP sends the assigned labels information to this MPLS OAM module which is used for validation of FEC outgoing Label stack that is received in an echo request.

mplsOamGeneralLearnedInfo *cget* -outgoingLspLabel

(read only) The outgoing LSP label value.

mplsOamGeneralLearnedInfo *cget* -outgoingPwLabel

(read only) The outgoing PW label value.

mplsOamGeneralLearnedInfo *cget* -bfdSessionPeerState

(read only) The state of the far side of the BFD session, either active or not.

mplsOamGeneralLearnedInfo *cget* -peerDiscriminator

(read only) The discriminator for the far side of the session.

mplsOamGeneralLearnedInfo *cget* -peerIpAddress

(read only) The learnt IP address for the session.

mplsOamGeneralLearnedInfo *cget* -pingAttempts

(read only) Specifies the number of ping attempts.

mplsOamGeneralLearnedInfo *cget* -pingFailures

(read only) Specifies the number of ping failures.

mplsOamGeneralLearnedInfo *cget* -pingReplyTx

(read only) Specifies the number of ping reply transmitted at regular intervals.

mplsOamGeneralLearnedInfo *cget* -pingRequestRx

(read only) Specifies the number of ping request received at regular intervals.

mplsOamGeneralLearnedInfo *cget* -pingSuccess

(read only) Specifies the number of ping request received at regular intervals.

mplsOamGeneralLearnedInfo *cget* -receivedMinRxInterval

(read only) The minimum receive interval, in milliseconds, for the far side of the session.

mplsOamGeneralLearnedInfo *cget* -receivedMultiplier

(read only) The number of received negotiated transmit intervals when multiplied by this value, provides the detection time for the interface.

mplsOamGeneralLearnedInfo *cget* -receivedPeerFlags

(read only) The number of peer generated flags received.

mplsOamGeneralLearnedInfo *cget* -receivedTxInterval

(read only) The minimum transmit interval, in milliseconds, for the far side of the session.

mplsOamGeneralLearnedInfo *cget* -returnCode

(read only) The return code value.

mplsOamGeneralLearnedInfo*cget* - returnSubCode

(read only) The return subcode value.

EXAMPLES

SEE ALSO

NAME - mplsOamGeneralLearnedInfo

NAME - mplsOamTriggeredPingLearnedInfo

mplsOamTriggeredPingLearnedInfo — holds lists of the triggered ping learned information.

SYNOPSIS

mplsOamTriggeredPingLearnedInfo *subcommand options*

DESCRIPTION

The *mplsOamTriggeredPingLearnedInfo* holds lists of the triggered ping learned information.

STANDARD OPTIONS

peerIpAddress

The learnt IP address for the session.

incomingLabelStack

The incoming label stack value.

outgoingLabelStack

The outgoing label stack value.

fec

Denotes the forwarding equivalence class type.

reachability

Specifies whether the queried MEP could be reached or not, Failure/ Partial/Complete.

returnCode

The return code value.

rtt

Denotes the Round Trip Time.

returnSubCode

The return subcode value.

COMMANDS

The *mplsOamTriggeredPingLearnedInfo* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsOamTriggeredPingLearnedInfo *cget* -fec

(read only) Forwarding equivalence class (FEC) type.

mplsOamTriggeredPingLearnedInfo *cget* -incomingLabelStack

(read only) The incoming label stack value.

.mplsOamTriggeredPingLearnedInfo *cget* -outgoingLabelStack

(read only) The outgoing label stack value.

mplsOamTriggeredPingLearnedInfo *cget* -peerIpAddress

(read only) The learnt IP address for the session.

mplsOamTriggeredPingLearnedInfo *cget* -reachability

(read only) Specifies whether the queried MEP could be reached or not, Failure/ Partial/Complete.

mplsOamTriggeredPingLearnedInfo *cget* -returnCode

(read only) The return code value.

mplsOamTriggeredPingLearnedInfo *cget* -returnSubCode

(read only) The return subcode value.

mplsOamTriggeredPingLearnedInfo *cget* -rtt

(read only) Denotes the Round Trip Time.

EXAMPLES

SEE ALSO

NAME - mplsOamTriggeredPingLearnedInfo

NAME - mplsOamTraceRouteLearnedInfo

mplsOamTriggeredPingLearnedInfo — holds lists of the trace route learned information.

SYNOPSIS

mplsOamTriggeredPingLearnedInfo *subcommand options*.

DESCRIPTION

The *mplsOamTraceRouteLearnedInfo* holds lists of the trace route learned information.

STANDARD OPTIONS

outgoingLabelStack

It signifies the outgoing label stack value.

incomingLabelStack

It signifies the incoming label stack value.

fec

It signifies the forwarding equivalence class type.

hops

It signifies the number hand over point types.

reachability

Specifies whether the queried MEP could be reached or not, Failure/ Partial/Complete.

numberOfReplyingHops

It signifies the number of replying HOPs.

senderHandle

It signifies the sender handle.

COMMANDS

The *mplsOamTraceRouteLearnedInfo* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsOamTraceRouteLearnedInfo *cget* -fec

(read only) Forwarding equivalence class (FEC) type. LDP FEC128:1.1.1.2-2.2.2.3 Type: 5 PW ID: 11

mplsOamTraceRouteLearnedInfo *cget* -hops

(read only) Hand over point types. (1, 1.1.1.2)

mplsOamTraceRouteLearnedInfo *cget* -incomingLabelStack

(read only) The incoming label stack value. 17-17

mplsOamTraceRouteLearnedInfo *cget* -numberOfReplyingHops

(read only) Number of replying HOPs. 1

mplsOamTraceRouteLearnedInfo *cget* -outgoingLabelStack

(read only) The outgoing label stack value. 17-17

mplsOamTraceRouteLearnedInfo *cget* -reachability

(read only) Specifies whether the queried MEP could be reached or not, Failure/ Partial/Complete. Reachable.

NAME - mplsOamHopLearnedInfo

mplsOamHopLearnedInfo — holds lists of the HOP learned information.

SYNOPSIS

mplsOamHopLearnedInfo *subcommand options*.

DESCRIPTION

The *mplsOamHopLearnedInfo* holds lists of the trace route learned information.

STANDARD OPTIONS

srcIp

It signifies the source of the IP.

returnCode

It signifies the return code value.

returnSubcode

It signifies the return subcode value.

ttl

Denotes the Time-to-live type.

NAME - linkOamServer

linkOamServer — configures the server options for the port for a Link OAM link

SYNOPSIS

linkOamServer *subcommand options*

DESCRIPTION

The *linkOamServer* command is necessary in order to access the mpls Oam component of the protocol server for a particular port. The *select* subcommand **must** be used before all other link Oam commands. Refer to the Ixia Reference Manual, Theory of Operations: Protocols chapter for a discussion on Link Oam testing with Ixia equipmentThe optional link Oam test package must be installed in order for this command to operate.

STANDARD OPTIONS

NONE

COMMANDS

The *linkOamServer* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

linkOamServer *chasID cardID portID*

Accesses the Link Oam component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis
- Invalid port specified

linkOamServer *addLink*

This is used to add a single link to the linkOamServer object.

- Arguments: linkName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

linkOamServer *getlink*

This is used for getting the linkOamLink instance from the linkOamServer.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

linkOamServer *setlink*

This is used for saving/setting a particular linkOamLink instance to the linkOamServer. It supports only one link.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

linkOamServer *showLinkNames*

This is used to display the names of all links, configured under linkOamServer.

Note: At present, this shows the name of only one link.

EXAMPLES

SEE ALSO

linkOamServer

NAME - linkOamLink

linkOamLink — contains the configuration information for a IxNetwork Link OAM link.

SYNOPSIS

linkOamLink *subcommand options*

DESCRIPTION

The *linkOamLink* command configures the link options for the port.

STANDARD OPTIONS

enabled

If true, this particular Link OAM link entry is enabled.

macAddress

Specifies the MAC address of the local DTE.

operationMode

Specifies the OAM operation mode in Ixia port. One of:

- Active (*default*)
- Passive

informationPduCount PerSecond

Specifies the timer that is used to ensure that OAM sub layer adheres to maximum number of OAMPDUs per second, and emits at least one OAMPDU per second. *Default= 1, Min= 1, Max= 10*.

localLostLinkTimer

Indicates the timer that is used to reset the Discovery state machine of local DTE when it is not receiving any Information PDU from remote DTE. *Default= 5, Min= 2, Max= 90*

supportsVariable Retrieval

If true, enables the variable retrieval support in ixia port. *Default= True*.

supportsRemote Loopback

If true, enables the remote loopback support in ixia port. *Default= True*.

supportsInterpreting LinkEvents

If true, enables the link event interpreting support in ixia port. *Default= True*.

supportsUnidirectional Mode

If true, enables the unidirectional mode in ixia port. *Default= True.*

enableCriticalEvent

If true, helps to indicate a critical event to remote peer by setting Critical Event bit in the flags field of Information PDUs transmitted thereafter. *Default= False*.

enableLinkFault

If true, indicates that a fault has occurred in the receive direction by setting Link Fault bit in the flags field of Information PDUs transmitted thereafter. Also the local DTE will move into FAULT state. *Default= False*.

enableDyingGasp

If true, helps to indicate an unrecoverable local failure condition to remote peer by setting Dying Gasp bit in the flags field of Information PDUs transmitted thereafter. *Default= False*.

enableVariable Response

If true, enables Variable response. This is used to determine whether to respond to variable request. *Default= True*.

Note: This is done for negative testing.

enableLoopback Response

If true, enables Loopback response. *Default= True*.

Note: This is done for negative testing.

disableInformationPduTx

If true, it controls the transmission of information PDU. Default= False.

disableNonInformation PduTx

If true, it controls the transmission of non- information PDU. Default= False.

overrideLocal Evaluating

If true, the local evaluating bit transmitted within Local Information TLVs is overridden. Default= False.

Note: This is done for negative testing.

overrideLocalSatisfied

If true, the local_satisfied variable used for discovery is overridden and the state of the local DTE is re-calculated accordingly. Default= False.

Note: This is done for negative testing.

overrideLocalStable

If true, the local stable flag is overridden. *Default= False*.

Note: This is done for negative testing.

overrideRemote Evaluating

If true, the remote evaluating bit transmitted within Local Information TLVs is overridden. *Default= False*.

Note: This is done for negative testing.

overrideRemoteStable

If true, the remote stable bit transmitted within Local Information TLVs is overridden. *Default= False*.

Note: This is done for negative testing.

overrideRevision

If true, overrides the revision field.

Note: If this not true, then *revision* is disabled.

overrideSequence Number

If true, the current sequence number can be overridden. *Default= 0*.

revision

Specifies the revision description. *Default= 0, Max= 65535*

sequenceNumber

Indicates the sequence number with which to override the current sequence number. This field remains false except when Override Sequence Number option is true. *Default= 0, Max= 65535.*

maxOamPduSize

The maximum OAMPDU size supported by local DTE. *Default= 1500 octets, Min= 64, Max= 1500*.

version

Specifies the version supported by this local DTE. It supports integer values. *Default= 0x01, Min= 0x00, Max= 0xFF*.

oui

Contains the 24 bit Organizationally Unique Identifier. *Default= 00 01 00*.

vendorSpecific Information

Contains the vendor specific information. This is used to differentiate the product modes/version of a vendor. *Default= 00000000*.

linkEventTxMode

Indicates the link event tx mode. One of:

- Single (*Default*)
- Periodic

eventInterval

Indicates the periodic interval of event pdu when event pdu is to be sent periodically. *Default= 1, Min= 1 sec, Max= 10*.

updateRequired

If true, sends the updated parameters information.

loopbackTimeout

Indicates the loopback timeout in milliseconds. Loopback timeout is the time period till which the local DTE will wait for the remote DTE's response to the Loopback Control PDU transmitted. *Default= 1000 ms*.

loopbackCommand

This contains the options of Enable OAM Remote Loopback. One of:

- disableLoopback
- enableLoopback (*default*)

variableResponse Timeout

The maximum time in seconds to wait for the variable response pdu. *Default= 1, Min= 500 ms, Max= 10 sec*.

COMMANDS

The *linkOamLink* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

addInterface

Used to add a *linkOamInterface* object to the *linkOamLink* object.

- Arguments: interfaceName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

delInterface

Used to remove a particular *linkOamInterface* from the *linkOamLink* object.

- Arguments: interfaceName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

getInterface

Used to get a particular instance of *linkOamInterface* from the *linkOamLink* object.

- Arguments: interfaceName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

setInterface

Used for setting a particular instance of the *linkOamInterface* to the *linkOamLink*.

- Arguments: interfaceName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

getFirstInterface

Used to get the first instance from the list of *linkOamInterfaces* configured under the *linkOamLink*.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

getNextInterface

Used to get the next instance from the list of *linkOamInterfaces* configured under *linkOamLink*. This should be called after a call to the *getFirstInterface*.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

clearAllInterfaces

Deletes all the *linkOamInterfaces* configured under *linkOamLink*.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

showInterfaceNames

Used to display the names of the all the *linkOamInterfaces* configured under the *linkOamLink* object.

getErroredSymbol PeriodEventTlv

Gets the Errored Symbol Period Event Tlv object instance configured under *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

setErroredSymbol PeriodEventTlv

Sets the instance of the Errored Symbol Period Event Tlv object to the *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

getErroredFrame EventTlv

Gets the Errored Frame Event Tlv object instance configured under *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

setErroredFrame EventTlv

Sets the instance of the Errored Frame Event Tlv object to the *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

getErroredFrame PeriodEventTlv

Gets the Errored Frame Period Event Tlv object instance configured under *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

setErroredFrame PeriodEventTlv

Sets the instance of the Errored Frame Period Event Tlv object to the *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

getErroredFrame SSEventTlv

Gets the Errored Frame Seconds Summary Event Tlv object instance configured under *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

setErroredFrame SSEventTlv

Sets the instance of the Errored Frame Seconds Summary Event Tlv object to the *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

getOrgSpecEventTlv

Gets the Organization Specific Event Tlv object instance configured under *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

setOrgSpecEventTlv

Sets the instance of the Organization Specific Event Tlv object to the *linkOamLink* object.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

addOrgSpecInfoTlv

Used to add a *linkOamOrgInfoTlv* object to the *linkOamLink* object.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

delOrgSpecInfoTlv

Used to remove a particular *linkOamOrgInfoTlv* from the *linkOamLink* object.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

getOrgSpecInfoTlv

Used to get a particular instance of *linkOamOrgInfoTlv* from the *linkOamLink* object.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

setOrgSpecInfoTlv

Used for setting a particular instance of the *linkOamOrgInfoTlv* to the *linkOamLink*.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

getFirstOrgSpecInfo Tlv

Used to get the first instance from the list of *linkOamOrgInfoTlv* configured under *linkOamLink*.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

getNextOrgSpecInfo Tlv

Used to get the next instance from the list of *linkOamOrgInfoTlv* configured under *linkOamLink*. This should be called after a call to the *getFirstOrgSpecInfoTlv*.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

clearAllOrgSpecInfo Tlvs

Deletes all the *linkOamOrgInfoTlv* configured under *linkOamLink*.

- Return value: 0 (success)
- ErrorValue: non-zero value failure

showOrgSpecInfoTlvNames

Used to display the names of the all the *linkOamOrgInfoTlvs* configured under the *linkOamLink* object.

addVariableResponseDbContainer

Used to add a *linkOamVarContainer* object to the *linkOamLink* object.

- Arguments: containerName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value failure

delVariableResponseDbContainer

Used to remove a particular *linkOamVarContainer* from the *linkOamLink* object.

- Arguments: containerName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getVariableResponseDbContainer

Used to get a particular instance of *linkOamVarContainer* from the *linkOamLink* object.

- Arguments: containerName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

setVariableResponseDbContainer

Used for setting a particular instance of the *linkOamVarContainer* to the *linkOamLink*.

- Arguments: containerName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getFirstVariable ResponseDbContainer

Used to get the first instance from the list of *linkOamVarContainer* configured under *linkOamLink*.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getNextVariable ResponseDbContainer

Used to get the next instance from the list of *linkOamVarContainer* configured under *linkOamLink*. This should be called after a call to the *getFirstVariableResponseDbContainer*.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

clearAllVariable ResponseDbContainer

Deletes all the *linkOamVarContainer* configured under linkOamLink.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

showVariable ResponseDbContainerNames

Used to display the names of the all the *linkOamVarContainer* configured under the *linkOamLink* object.

addOrgSpecTlv

Used to add a *linkOamOrgTlv* object to the *linkOamLink* object.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

Note: Only one instance of this tlv should be configured under the linkOamLink object.

delOrgSpecTlv

Used to remove a particular linkOamOrgTlv from the linkOamLink object.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getOrgSpecTlv

Used to get a particular instance of *linkOamOrgTlv* from the *linkOamLink* object.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

setOrgSpecTlv

Used for setting a particular instance of the *linkOamOrgTlv* to the *linkOamLink*.

- Arguments: tlvName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

clearAllOrgSpecTlvs

Deletes all the *linkOamOrgTlv* configured under *linkOamLink*.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

showOrgSpecTlv Names

Used to display the names of the all the *linkOamOrgTlv* configured under the *linkOamLink* object.

addVarDescriptor

Used to add a *linkOamVarDescriptor* object to the *linkOamLink* object.

- Arguments: descriptorName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

delVarDescriptor

Used to remove a particular *linkOamVarDescriptor* from the *linkOamLink* object.

- Arguments: descriptorName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getVarDescriptor

Used to get a particular instance of *linkOamVarDescriptor* from the *linkOamLink* object.

- Arguments: descriptorName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

setVarDescriptor

Used for setting a particular instance of the *linkOamVarDescriptor* to the *linkOamLink*.

- Arguments: descriptorName (string)
- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getFirstVarDescriptor

Used to get the first instance from the list of *linkOamVarDescriptor* configured under *linkOamLink*.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getNextVarDescriptor

Used to get the next instance from the list of *linkOamVarDescriptor* configured under *linkOamLink*. This should be called after a call to the *getFirstVarDescriptor*.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

clearAllVarDescriptors

Deletes all the *linkOamVarDescriptor* configured under linkOamLink.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

showVarDescriptorNames

Displays the names of the all the *linkOamVarDescriptor* configured under the *linkOamLink* object.

requestDiscLearned Info

Requests the Per-Link Discovered Learned Info.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

sendLoopback

Sends a Loopback request on a per-link basis.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getDiscLearnedInfo

Retrieves the Discovered Learned Info after *requestDiscLearnedInfo* is being called.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

requestEventNotifn LearnedInfo

Requests the Per-Link Event Notification Learned Info.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getEventNotifn LearnedInfo

Retrieve the Event Notification Learned Info after *requestEventNotifnLearnedInfo* is being called.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

requestVariable ResponseLearnedInfo

Requests the Per-Link Variable Response Learned Info.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getVariableResponseLearnedInfoList

Retrieves the list of *linkOamVarRequestLearnedInfo* after *requestVariable ResponseLearnedInfo* has been called.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getFirstVariable ResponseLearnedInfo

Retrieves the first *linkOamVarRequestLearnedInfo* from the list of linkOamVarRequestLearnedInfo obtained by the *getVariableResponseLearnedInfoList* call.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

getNextVariable ResponseLearnedInfo

Retrieves the next linkOamVarRequestLearnedInfo from the list of linkOamVarRequestLearnedInfo. This should be called after getFirstVariableResponseLearnedInfo.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

sendUpdated Parameters

Updates any change to the On-the-fly parameters to the daemon. This is an On-the-Fly trigger api.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

startEventPDU Transmission

Starts transmission of the OAM Event PDU. This is one-time or continous, depending whether *linkEventTxMode* is single or periodic. This is an On-the-Fly trigger api.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

stopEventPDU Transmission

Stops transmission of the OAM Event PDU. This is in effect if *linkEventTxMode* is periodic. This is an On-the-Fly trigger api.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

sendOrgSpecificPDU

Sends an Organization Specific PDU. This is an On-the-Fly trigger api.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

restartDiscovery

Restarts discovery mechanism of the OAM protocol. This is an On-the-Fly trigger api.

- Return value: 0 (success)
- ErrorValue: non-zero value (failure)

EXAMPLES

```
linkOamLink setDefault

linkOamLink config -enabled true

linkOamLink config -macAddress

"00:00:00:10:00:00"

linkOamLink config -operationMode activeMode

linkOamLink config -informationPDUCountPerSecond 1

linkOamLink config -localLostLinkTimer 5

linkOamLink config -supportsVariableRetrieval 1

linkOamLink config -supportsRemoteLoopback 1

linkOamLink config -supportsInterpretingLinkEvents 1

linkOamLink config -supportsUnidirectionalMode 1

linkOamLink config -enableCriticalEvent false

linkOamLink config -enableLinkFault false

linkOamLink config -enableDyingGasp false

linkOamLink config -enableVariableResponse true

linkOamLink config -enableLoopbackResponse true

linkOamLink config -disableInformationPDUTx 0

linkOamLink config -disableNonInformationPDUTx 0

linkOamLink config -overrideLocalEvaluating 0

linkOamLink config -overrideLocalSatisfied 0

linkOamLink config -overrideLocalStable 0

linkOamLink config -overrideRemoteEvaluating 0

linkOamLink config -overrideRemoteStable 0

linkOamLink config -overrideRevision 0

linkOamLink config -overrideSequenceNumber 0

linkOamLink config -revision 0

linkOamLink config -sequenceNumber 0

linkOamLink config -maxOAMPDUSize 1500

linkOamLink config -version 1
linkOamLink config -oui "00 01 00"
linkOamLink config -vendorSpecificInformation "00 00 00
00"
linkOamLink config -linkEventTxMode single
linkOamLink config -eventInterval 1
linkOamLink config -supportsUniqueSequenceNumber 0
linkOamLink config -updateRequired 1
linkOamLink config -loopbackTimeout 1000
linkOamLink config -loopbackCommand
enableLoopback
linkOamLink config -variableResponseTimeout 1000
linkOamLink config -ethernetTypeUsedForDataTraffic 65535
linkOamServer addLink Link1
```

SEE ALSO

linkOamLink

NAME - linkOamInterface

linkOamInterface — holds the information related to a single interface on the simulated link.

SYNOPSIS

linkOamInterface *subcommand options*

DESCRIPTION

The *linkOamInterface* holds the information related to a single interface on the simulated link.

STANDARD OPTIONS

enabled

Enables the protocol interface.

interfaceId

(read ony) Specifies the interface id.

protocolInterface Description

Specifies the pre-configured protocol interfaces.

EXAMPLES

```
linkOamInterface setDefault

linkOamInterface config -enabled true

linkOamInterface config -interfaceId 1

linkOamInterface config -protocolInterfaceDescription

"ProtocolInterface - 187:118 - 1"

if {[linkOamLink addInterface Interface1]} {

errorMsg "Error calling linkOamLink addInterface

Interface1"

set retCode $::TCL_ERROR
```

SEE ALSO

linkOamLink

NAME - linkOamSymTlv

linkOAMSymTlv — configures the error symbol period event tlv in the event notification packet.

SYNOPSIS

linkOamSymTlv *subcommand options*

DESCRIPTION

The *linkOamSymTlv* command configures the error symbol period event tlv in the event notification packet.

STANDARD OPTIONS

enabled

If true, includes the error symbol period event tlv in the event notification packet.

Default= False.

errSymbWindow

Indicates the number of symbols in the period, encoded as a 16-bit unsigned integer. *Default=10, Min= 1, Max= 65535.*

errSymb Threshold

Indicates the number of errored symbols in the period. This is required to be equal to or greater than, in order for the event to be generated. It is encoded as a 16-bit unsigned integer. *Default= 1, Min= 0 , Max= 65535*.

errSymbols

Indicates the number of symbol errors in the period, encoded as a 32-bit unsigned integer. *Default= 0, Min= 0, Max= 4 octets.*

EXAMPLES

```
linkOamSymTlv setDefault

linkOamSymTlv config -enabled true

linkOamSymTlv config -errSymbWindow 10

linkOamSymTlv config -errSymbThreshold 1

linkOamSymTlv config -errSymbols 0

linkOamLink setErroredSymbolPeriodEventTlv
```

SEE ALSO

linkOamLink

NAME - linkOamFrameTlv

linkOamFrameTlv — configures the errored frame event tlv in the event notification packet.

SYNOPSIS

linkOamFrameTlv *subcommand options*

DESCRIPTION

The *linkOamFrameTlv* command configures the errored frame event tlv in the event notification packet.

STANDARD OPTIONS

enabled

If true, it determines whether to include this tlv in the event notification packet. *Default= False*.

errFrameThreshold

Indicates whether the number of detected errored frames in the period is required to be equal to or greater than in order for the event to be generated. It is encoded as a 16-bit unsigned integer. *Default= 1, Min= 0, Max= 65535.*

errFrames

Indicates the number of detected errored frames in the period, encoded as a 16-bit unsigned integer. *Default= 1, Min= 0, Max= 4 octets.*

errFrameWindow

Indicates the duration of the period in terms of 100 ms intervals, encoded as a 16-bit unsigned integer. *Default= 1 second (10ms), Min= 1 second (10ms), Max= 1 minute (600 ms) .*

EXAMPLES

```
linkOamFrameTlv setDefault

linkOamFrameTlv config -enabled true

linkOamFrameTlv config -errFrameThreshold 1

linkOamFrameTlv config -errFrames 1

linkOamFrameTlv config -errFrameWindow 1

linkOamLink setErroredFrameEventTlv
```

SEE ALSO

linkOamLink

NAME - linkOamPeriodTlv

linkOamPeriodTlv — configures the errored frame period event tlv in the event notification packet.

SYNOPSIS

linkOamPeriodTlv *subcommand options*

DESCRIPTION

The *linkOamPeriodTlv* holds the information related to a single interface on the simulated link.

STANDARD OPTIONS

enabled

If true, it determines whether to include this tlv in the event notification packet. *Default= False*.

errFrameWindow

Indicates the duration of period in terms of frames, encoded as a 16-bit unsigned integer. *Default= 10, Min= 1, Max= 65535*.

errFrameThreshold

Indicates whether the number of errored frames in the period is required to be equal to or greater than in order for the event to be generated. It is encoded as a 16-bit unsigned integer. *Default= 1, Min= 0, Max= 65535.*

errFrames

Indicates the number of frame errors in the period, encoded as a 16-bit unsigned integer. *Default= 1, Min= 0, Max= 65535*.

EXAMPLES

```
linkOamPeriodTlv setDefault

linkOamPeriodTlv config -enabled true

linkOamPeriodTlv config -errFrameWindow 10

linkOamPeriodTlv config -errFrameThreshold 1

linkOamPeriodTlv config -errFrames 1

linkOamLink setErroredFramePeriodEventTlv
```

SEE ALSO

linkOamLink

NAME - linkOamSSTlv

linkOamSSTlv — configures the errored frame seconds summary event tlv in the event notification packet.

SYNOPSIS

linkOamSSTlv *subcommand options*

DESCRIPTION

The *linkOamSSTlv* command configures the errored frame seconds summary event tlv in the event notification packet.

STANDARD OPTIONS

enabled

If true, it determines whether to include this tlv in the event notification packet. *Default= False*.

errFrameSecSum Window

Indicates the duration of the period in terms of 100 ms intervals, encoded as a 16-bit unsigned integer. *Default= 60 seconds (600 ms* in terms of 100 ms *), Min= 10 seconds (100 ms* in terms of 100 ms*), Max= 900 seconds (9000 ms* in terms of 100 ms*).*

errFrameSecSum Threshold

Indicates whether the number of errored frame seconds in the period is required to be equal to or greater than, in order for the event to be generated. It is encoded as a 16-bit unsigned integer. *Default= one errored second, Min= zero errored second, Max= two octets errored seconds*.

errFrameSecSum

Indicates the number of errored frame seconds in the period, encoded as a 16-bit unsigned integer. *Default= 1, Min= 0, Max= 65535.*

EXAMPLES

```
linkOamSSTlv setDefault

linkOamSSTlv config -enabled true

linkOamSSTlv config -errFrameSecSumWindow 600

linkOamSSTlv config -errFrameSecSumThreshold 1

linkOamSSTlv config -errFrameSecSum 1

linkOamLink setErroredFrameSSEventTlv
```

SEE ALSO

linkOamLink

NAME - linkOamOrgEventTlv

linkOamOrgEventTlv — configures the organization specific event tlv in the event notification packet.

SYNOPSIS

linkOamOrgEventTlv *subcommand options*

DESCRIPTION

The *linkOamOrgEventTlv* command configures the organization specific event tlv in the event notification packet.

STANDARD OPTIONS

enabled

If true, it determines whether to include this tlv in the event notification packet. *Default= False*.

oui

This three-octet field contains a 24-bit Organizationally Unique Identifier (OUI).

Default= 00 01 00.

Note: Any three octets hex value may be given.

value

Indicates the value of the Organization Specific Event.

Note: This has an unspecified field length. Any hex value may be given.

EXAMPLES

```
linkOamOrgEventTlv setDefault

linkOamOrgEventTlv config -enabled true

linkOamOrgEventTlv config -oui "00

01 00"

linkOamOrgEventTlv config -value "00

00 00 "

linkOamLink setOrgSpecEventTlv
```

SEE ALSO

linkOamLink

NAME - linkOamOrgInfoTlv

linkOamOrgInfoTlv — configures the organization specific information tlv in the information OAMPDU.

SYNOPSIS

linkOamOrgInfoTlv *subcommand options*

DESCRIPTION

The *linkOamOrgInfoTlv* command configures the organization specific information tlv in the information OAMPDU.

STANDARD OPTIONS

enabled

If true, it determines whether to include this tlv in the information OAMPDU. *Default= False*.

oui

This three-octet field contains a 24-bit Organizationally Unique Identifier (OUI).

Default= 00 01 00.

Note: Any three octets hex value may be given.

value

Indicates the value of the Organization Specific Event.

Note: This has an unspecified field length. Any hex value may be given.

EXAMPLES

```
linkOamOrgInfoTlv setDefault

linkOamOrgInfoTlv config -enabled true

linkOamOrgInfoTlv config -oui "00 01

00"

linkOamOrgInfoTlv config -value "01 02

03 04 "

linkOamLink addOrgSpecInfoTlv
```

SEE ALSO

linkOamLink

NAME - linkOamVarContainer

linkOamVarContainer — configures the variable response container.

SYNOPSIS

linkOamVarContainer *subcommand options*

DESCRIPTION

The *linkOamVarContainer* command configures the variable response container.

STANDARD OPTIONS

enabled

If true, enables the container. *Default= false*.

variableBranch

Indicates the variable branch value in hex. *Default= 00.*

variableWidth

Indicates the length of the variable value field in octets. *Default= 00*.

variableIndication

If true, this indicates that the Leaf has some error. *Default: False*.

variableLeaf

Indicates the variable leaf value in hex. *Default= 00.*

variableValue

The Variable Value may be 1 to 128 octets in length. Its width is determined by the Variable Width field. *Default= 00*.

EXAMPLES

```
linkOamVarContainer setDefault

linkOamVarContainer config -enabled true

linkOamVarContainer config -variableBranch 7

linkOamVarContainer config -variableWidth 2

linkOamVarContainer config -variableIndication 0

linkOamVarContainer config -variableLeaf 1536

linkOamVarContainer config -variableValue "01

02 "

linkOamLink addVariableResponseDbContainer
```

SEE ALSO

linkOamLink

NAME - linkOamOrgTlv

linkOamOrgTlv — configures the organization specific OAMPDU.

SYNOPSIS

linkOamOrgTlv *subcommand options*

DESCRIPTION

The *linkOamOrgTlv* command configures the organization specific OAMPDU.

STANDARD OPTIONS

enabled

If true, it determines whether to include this tlv in the organization specific OAMPDU. *Default= False*.

oui

This three-octet field contains a 24-bit Organizationally Unique Identifier (OUI).

Default= 00-01-00.

Note: Any three octets hex value may be given.

value

Indicates the value of the Organization Specific Information TLV.

Note: This has an unspecified field length. Any hex value may be given.

EXAMPLES

```
linkOamOrgTlv setDefault

linkOamOrgTlv config -oui "00 01 00"

linkOamOrgTlv config -value ""

linkOamLink addOrgSpecTlv OrgSpecTLV1
```

SEE ALSO

linkOamLink

NAME - linkOamDiscLearnedInfo

linkOamDiscLearnedInfo — fetches and describes the discovered learned data.

SYNOPSIS

linkOamDiscLearnedInfo *subcommand options*

DESCRIPTION

The *linkOamDiscLearnedInfo* command fetches and describes the discovered learned data.

STANDARD OPTIONS

remoteMac Address

(Read only) Indicates the Mac address of the remote DTE for the link.

remoteStable

(Read only) Indicates the stability status of the remote DTE. It is displayed as either 0 or 1.

remoteEval

(Read only) Indicate whether remote DTE is in the discovery process or not. It is displayed as either 0 or 1.

remoteCritical Event

(Read only) Indicates whether any critical event has been received from the remote DTE. It is displayed as either 0 or 1.

remoteDying Gasp

(Read only) Indicates whether any unrecoverable failure has occurred on the remote DTE. It is displayed as either 0 or 1.

remoteLinkFault

(Read only) Indicates whether receive path has detected error on remote DTE. It is displayed as either 0 or 1.

remoteOam Version

(Read only) Indicates the OAM version supported by the remote DTE.

remoteMuxAction

(Read only) Indicates the state of multiplexer of remote DTE. One of:

- fwd (value 0): Remote DTE is forwarding non-OAMPDUs
- discard (value 1): Remote DTE is discarding non-OAMPDUs

remoteMode

(Read only) Indicates the configuration mode for the remote DTE. One of:

- active (value 0): Remote DTE is in active mode
- passive (value 1): Remote DTE is in passive mode

remoteParser Action

(Read only) Indicates the state of parser of remote DTE. One of:

- fwd (value 0): Remote DTE is forwarding non-OAMPDUs to higher layer
- lb (value 1): Remote DTE is looping back the non-OAMPDUs
- discard (value 2): Remote DTE is discarding non-OAMPDUs

remoteRevision

(Read only) Indicates the current revision of the information tlv of remote DTE.

remoteMaxPdu Size

(Read only) Indicates the maximum pdu size supported by the remote DTE.

remoteVariable Retrieval

(Read only) Indicates whether remote DTE supports responding to variable request. It is displayed as either 0 or 1.

remoteLinkEvent

(Read only) Indicates whether remote DTE supports interpreting link events. It is displayed as either 0 or 1.

remoteLoopback Support

(Read only) Indicates whether remote DTE is capable of remote loopback mode. It is displayed as either 0 or 1.

remote Unidirectional Support

(Read only) Indicates whether remote DTE is capable of sending OAMPDUs when the receive path is non operational. It is displayed as either 0 or 1.

remoteOui

(Read only) Specifies the remote OUI value.

remoteVendor SpecInfo

(Read only) Indicates the remote vendor specific information.

localDiscStatus

(Read only) Indicates the status of the discovery process. One of:

- fault
- activeSendLocal
- passiveWait
- sendLocalRemote
- sendLocalRemoteOk
- sendAny

localStable

(Read only) Indicates the stability status of local DTE. It is displayed as either TRUE or FALSE.

localEval

(Read only) Indicates whether the local DTE is in the discovery process or not. It is displayed as either TRUE or FALSE.

localRevision

(Read only) Indicates the current revision of the information tlv of local DTE.

localMuxAction

(Read only) Indicates the state of multiplexer of local DTE. One of:

- fwd (value 0): Local DTE is forwarding non-OAMPDUs
- discard (value 1): Local DTE is discarding non-OAMPDUs

localParserAction

(Read only) Indicates the state of parser of the local DTE. One of:

- fwd (value 0): Local DTE is forwarding non-OAMPDUs to higher layer
- lb (value 1): Local DTE is looping back the non-OAMPDUs
- discard (value 2): Local DTE is discarding non-OAMPDUs

remoteHeader Refreshed

(Read only) If true, check whether remote information is available or not.

remoteTlv Refreshed

(Read only) If true, checks whether remote tlv is available or not.

EXAMPLES

```
linkOamDiscLearnedInfo cget -localDiscStatus

linkOamDiscLearnedInfo cget -localEvaluating

linkOamDiscLearnedInfo cget -localParserAction

linkOamDiscLearnedInfo cget -localRevision
```

SEE ALSO

linkOamLink

NAME - linkOamEventNotifnInfo

linkOamEventNotifnInfo — fetches and describes the learned data for event notification.

SYNOPSIS

linkOamEventNotifnInfo *subcommand options*

DESCRIPTION

The *linkOamEventNotifnInfo* command fetches and describes the learned data for event notification.

STANDARD OPTIONS

remoteSymbolPeriodWindow

(Read only) The number of symbols in the period configured in the remote DTE.

remoteSymbolPeriod Threshold

(Read only) The number of errored symbols configured in the remote DTE to generate this event.

remoteSymbolPeriod Errors

(Read only) The number of errored symbols in the period received in the last received event.

remoteSymbolPeriod ErrorRunningTotal

(Read only) The total number of Errored Symbol Period Symbols Error received from the emulation start time.

remoteSymbolPeriod EventRunningTotal

(Read only) The total number of Errored Symbol Period Event TLVs received from the emulation start time.

remoteFrameWindow

(Read only) The duration of period in terms of 100 ms intervals configured in the remote DTE.

remoteFrameThreshold

(Read only) The number of errored frames configured in the remote DTE to generate this event.

remoteFrameError

(Read only) The number of errored frames in the period received in the last received event.

remoteFrameError RunningTotal

(Read only) The total number of Errored Frame Error received from the emulation start time.

remoteFrameEvent RunningTotal

(Read only) The total number of Errored Frame Event TLVs received from the emulation start time.

remoteFramePeriodWindow

(Read only) The duration of period in terms of frames configured in the remote DTE.

remoteFrame PeriodThreshold

(Read only) The number of errored frames configured in the remote DTE, to generate this event.

remoteFramePeriod Error

(Read only) The number of errored frames in the period received in the last received event.

remoteFramePeriod ErrorRunningTotal

(Read only) The total number of Errored Frame Period Frame Error received from the emulation start time.

remoteFramePeriod EventRunningTotal

(Read only) The total number of Errored Frame Period Event TLVs received from the emulation start time.

remoteFrameSecSum Window

(Read only) The duration of period in terms of 100 ms configured in the remote DTE.

remoteFrameSecSum Threshold

(Read only) The number of errored frames seconds configured in the remote DTE, to generate this event.

remoteFrameSecSumError

(Read only) The number of errored frames second in the period received in the last received event.

remoteFrameSecSumErrorRunningTotal

(Read only) The Total number of Errored Frame Sec Sum Error received from the emulation start time.

remoteFrameSecSum EventRunningTotal

(Read only) The total number of Errored Frame Sec Sum Event TLVs received from the emulation start time.

localSymbolPeriod ErrorRunningTotal

(Read only) The total number of Errored Symbol Period Symbols Error sent from the local DTE, from the emulation start time.

localSymbolPeriod EventRunningTotal

(Read only) The total number of Errored Symbol Period Event TLVs sent from local DTE, from the emulation start time.

localFrameError RunningTotal

(Read only) The total number of Errored Frame Error sent from local DTE, from the emulation start time.

localFrameEvent RunningTotal

(Read only) The total number of Errored Frame Event TLVs sent from local DTE, from the emulation start time.

localFramePeriodErrorRunningTotal

(Read only) The total number of Errored Frame Period Frame Error sent from local DTE, from the emulation start time.

localFramePeriod EventRunningTotal

(Read only) The total number of Errored Frame Period Event TLVs sent from local DTE, from the emulation start time.

localFrameSecSum ErrorRunningTotal

(Read only) The total number of Errored Frame Sec Sum Error sent from local DTE, from the emulation start time.

localFrameSec SumEventRunningTotal

(Read only) The total number of Errored Frame Sec Sum Event TLVs sent from local DTE, from the emulation start time.

EXAMPLES

```
linkOamEventNotifnLearnedInfo cget -remoteFrameWindow

linkOamEventNotifnLearnedInfo cget -remoteFrameThreshold

linkOamEventNotifnLearnedInfo cget -remoteFrameError

linkOamEventNotifnLearnedInfo cget -remoteFrameErrorRunningTotal

linkOamEventNotifnLearnedInfo cget -remoteFrameEventRunningTotal

linkOamEventNotifnLearnedInfo cget -localFrameErrorRunningTotal
```

```
  linkOamEventNotifnLearnedInfo cget -localFrameEventRunningTotal
```

SEE ALSO

linkOamLink

NAME - linkOamVarRequestLearnedInfo

linkOamVarRequestLearnedInfo — fetches and describes the learned data for variable request.

SYNOPSIS

linkOamVarRequestLearnedInfo *subcommand options*

DESCRIPTION

The *linkOamVarRequestLearnedInfo* command fetches and describes the learned data for variable request.

STANDARD OPTIONS

variableBranch

(Read only) Contains the value of the requesting branch.

variableWidth

(Read only) Indicates the length of the variable value. An encoding of 0x00 equals to 128 octets.

variableIndication

(Read only) Indicates the status of the retrieved variable container.

variableLeaf

(Read only) Contains the value of the requesting leaf value.

variableValue

(Read only) Indicates the variable value.

EXAMPLES

```
linkOamVarRequestLearnedInfo cget -variableBranch

linkOamVarRequestLearnedInfo cget -variableIndication

linkOamVarRequestLearnedInfo cget -variableLeaf

linkOamVarRequestLearnedInfo cget -variableValue

linkOamVarRequestLearnedInfo cget -variableWidth
```

SEE ALSO

linkOamLink

NAME - ldpAdvertiseFecRange

ldpAdvertiseFecRange — sets up the parameters associated with an LDP FEC range.

SYNOPSIS

ldpAdvertiseFecRange *subcommand options*

DESCRIPTION

The *ldpAdvertiseFecRange* command describes an individual set of FECs. FEC ranges are added into *ldpRouter* lists using the *ldpRouter addAdvertiseFecRange* command. Refer to *ldpAdvertiseFecRange* for an overview of this command.

STANDARD OPTIONS

enable true / false

Enables the use of this FEC range for the simulated router. *(default = false)*

enablePacking true | false

Enables the packing of multiple label mappings on sending labels and sending withdrawals into a single PDU*.* This is only applicable to sessions established in the downstream unsolicited mode. *(default = false)*

enableGracefulRestart true | false

If *true,* enables Graceful Restart in 'Helper" mode for this router. *(default = false)*

enablePduRateControl true | false

If *true*, enables the use of PDU rate control. PDU rate control only applies to Downstream Unsolicited Mode and is for use with Label Mapping and Label Withdraw PDUs. *(default = false)*

interPduGap

The gap time between PDUs, in milliseconds (ms). *(default = 50)*

labelIncrementMode

Whether to increment the label associated with networks or not. Options include:

Option	Value	Usage
ldpAdvertiseFecRangeFixed ldpAdvertiseFecRangeNone	0	(default) Do not change the label
ldpAdvertiseFecRangeIncrement	1	Increment the label by 1 for each new network advertised.

labelValueStart

The first label to be assigned to the FEC. *(default = 16)*

maskWidth

The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0. *(default = 24)*

networkIpAddress

The IP address of the routes to be advertised. *(default = 0.0.0.0)*

numRoutes

The number of routes to be advertised. *(default = 1)*

reconnectTim

The graceful restart reconnect time, expressed in milliseconds. This Fault Tolerant (FT) Reconnect Timer value is advertised in the FT Session TLV in the Initialization message sent by a neighbor LSR. A value of 0, indicates that the sender will not preserve its MPLS forwarding state across the restart. *(default = 120,000)*

recoveryTime

The graceful restart recovery time, expressed in milliseconds). The restarting LSR advertises the amount of time that it will retain its MPLS forwarding state. A value of 0 means that the restarting LSR was not able to preserve the MPLS forwarding state. *(default = 120,000)*

STANDARD OPTIONS

baseLabel

The first label to be assigned to the FEC. Now referred to as *labelValueStart*.

COMMANDS

The **ldpAdvertiseFecRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpAdvertiseFecRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ldpAdvertiseFecRange** command.

ldpAdvertiseFecRange config *option value*

Modify the configuration options of the **ldpAdvertiseFecRange**. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for **ldpAdvertiseFecRange**.

ldpAdvertiseFecRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAssignedAtmLabel, *ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpAssignedAtmLabel

ldpAssignedAtmLabel — accesses assigned ATM LDP labels.

SYNOPSIS

ldpAssignedAtmLabel *subcommand options*

DESCRIPTION

The *ldpAssignedAtmLabel* command holds an element of the LDP ATM assigned label list obtained in the *ldpInterface* command.

STANDARD OPTIONS

fec

(Read-only) The prefix associated with the label.

fecPrefixLength

(Read-only) The prefix length for the value in *fec*.

peerIpAddress

(Read-only) The IP address of the peer that the label was learned from.

state

(Read-only)

vci

(Read-only) The VCI of the assigned label.

vpi

(Read-only) The VPI of the assigned label.

COMMANDS

The *ldpAssignedAtmLabel* command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

ldpAssignedAtmLabel cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *ldpAssignedAtmLabel* command.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpATMRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpAtmRange

ldpAtmRange — accesses assigned ATM LDP labels.

SYNOPSIS

ldpAtmRange *subcommand options*

DESCRIPTION

The *ldpAtmRange* command holds an element of the LDP ATM assigned label list obtained in the *ldpInterface* command.

STANDARD OPTIONS

maxVci

The maximum VCI in the ATM label range.

maxVpi

The maximum VPI in the ATM label range.

minVci

The minimum VCI in the ATM label range.

minVpi

The minimum VPI in the ATM label range.

COMMANDS

The *ldpAtmRange* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpAtmRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *ldpAtmRange* command.

ldpAtmRange config *option value*

Modify the configuration options of the *ldpAtmRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ldpAtmRange.

ldpAtmRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpExplicitIncludeIpFec

ldpExplicitIncludeIpFec — configures an explicit IP FEC to filter received LDP FECs.

SYNOPSIS

ldpExplicitIncludeIpFec *subcommand options*

DESCRIPTION

The *ldpExplicitIncludeIpFec* command holds a single filter element, used in *ldpRouter*.

STANDARD OPTIONS

enable true | false

If *true*, then this entry is used for filtering. *(default = false)*

enableExactPrefix true | false

If *true*, then the mask width *(maskWidth)* of the received FEC must match as well as the networkIpAddress. Otherwise, any prefix match less than or equal to *maskWidth* will allow the received FEC to be learned. *(default = false)*

maskWidth

The prefix length of the *networkIpAddress*. *(default = 24)*

networkIpAddress

The IP address component of the FEC. *(default = 0.0.0.0)*

numNetworks

The number of networks covered by the FEC. *(default = 1)*

COMMANDS

The *ldpExplicitIncludeIpFec* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpExplicitIncludeIpFec cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *ldpExplicitIncludeIpFec* command.

ldpExplicitIncludeIpFec config *option value*

Modify the configuration options of the ldpExplicitIncludeIpFec. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ldpExplicitIncludeIpFec.

ldpExplicitIncludeIpFec setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpInterface

ldpInterface — configures an interface for an LDP router.

SYNOPSIS

ldpInterface *subcommand options*

DESCRIPTION

The *ldpInterface* holds the information related to a single interface on the simulated router. Interfaces are added into the *ldpRouter* interface list using the *ldpRouter addInterface* command. Refer to ldpInterface21 for an overview.

This command holds a list of targeted peers which are contacted directly during the discovery phase. This list is only used if the *discoveryMode* is set to *ldpInterfaceExtended*.

ATM sessions may be established by setting *enableAtmSessions*, *atmVcDirection* and then adding ATM VC ranges with the *addAtmVcRange* subcommand. Assigned labels are read through the use of the *requestAssignedAtmLabels, getAssignedAtmLabelList, getFirstAssignedAtmLabel* and *getNextAssignedAtmLabel* subcommands.

STANDARD OPTIONS

advertisingMode

The mode by which the simulate router advertises its FEC ranges. Options include

Option	Value	Usage
ldpInterfaceDownstream Unsolicited	1	(default) This is the only option currently available. The router distributes FEC ranges whenever it has a new one.
ldpInterface DownstreamOnDemand	2	The router only distributes FEC ranges when requested by a peer.

atmVcDirection

The ATM directional virtual circuit (VC) capability of this LSR. Unidirectional VC label assignment must be used when either or both of the LSRs specifies unidirectional VC capability. Options include

Option	Value	Usage
atmVcUnidirectional	0	Unidirectional. In a label mapping for a defined VPI, a defined VCI can be used for only one direction on the link.
atmVcBidirectional	1	*(default)* In a label mapping for a defined VPI, a defined VCI can be used for both directions on the link.

authenticationType

The cryptographic authentication type used for the interface. One of

Option	Value	Usage
ldpInterfaceNULL	0	(default) No cryptographic authentication will be used.

Option	Value	Usage
ldpInterfaceMD5	1	The Message Digest 5 (MD5) algorithm will be used for cryptographic authentication. If selected, an MD5 key must be configured. See *md5Key* below.

discoveryMode

The discovery mode used for this interface. Options include:

Option	Value	Usage
ldpInterfaceBasic	0	(default) The basic mode which does not attempt to contact targeted peers.
ldpInterfaceExtended	1	Attempts to contact targeted peers.
ldpInterfaceExtended Martini	2	Enables the advertisement of Layer 2 VC FECs from this interface.

enable true / false

If set, enables the use of this interface for the simulated router. *(default = false)*

enableAtmSession true / false

If set, enables the use of ATM sessions on this interface. ATM Label Ranges will be negotiated and used. *(default = false)*

labelSpaceId

The label space identifier for the interface. *(default = 0)*

md5Key

Used with MD5 Authentication. A user-defined string; maximum = 255 characters.

numLearnedLabels

Read-only. The number of labels learned (both IPv4 and Martini labels) as a result of a call to the *getLearnedLabelList* subcommand.

protocolInterface Description

The *description* option associated with an *interfaceEntry* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

requestingMode

The mode by which the router requests labels from peers. Only *independent* mode is currently implemented. *(default = 1)*

enableBFDRegistration

If true, enables BFD registration with LDP.

DEPRECATED STANDARD OPTIONS

ipAddress

The IP address for this interface. This option is only used if *protocolInterfaceDescription* is empty. *(default = 0.0.0.0)*

maskWidth

The mask associated with the interface. This option is only used if *protocolInterfaceDescription* is empty. *(default = 0)*

COMMANDS

The **ldpInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpInterface addAtmLabelRange *atmRangeLocalId*

Adds an ATM range for the interface from the options in the *ldpAtmRange* command. The peer is given a local id of *atmRangeLocalId*.

ldpInterface addTargetedPeer *targetedPeerLocalId*

Adds a targeted peer for the interface from the options in the *ldpTargetedPeer* command. The peer is given a local id of *targetedPeerLocalId*.

ldpInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ldpInterface** command.

ldpInterface clearAllAtmLabelRanges

Removes all ATM ranges from the interface.

ldpInterface clearAllTargetedPeers

Removes all targeted peers from the interface.

ldpInterface config *option value*

Modify the configuration options of the *ldpInterface*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *ldpInterface*.

ldpInterface delAtmLabelRange *atmRangeLocalId*

Deletes the ATM range indicated by the value of *atmRangeLocalId.* Specific errors include:

- No ATM range with the *atmRangeLocalId* label exists in the list.

ldpInterface getAssignedAtmLabelList

This command must be preceded by use of the *ldpInterface requestAtmLabelList* and followed by a calls to *ldpInterface getFirstAssignedAtmLabel/getNextAssignedAtmLabel.* This command determines whether the reading of assigned ATM labels from the protocol server has completed. This command should be called until it returns a `0' (TCL_OK), or until some suitable period of time has elapsed. The number of learned labels is available in the option.

ldpInterface getAtmLabelRange *atmRangeLocalId*

Deletes the ATM range indicated by the value of *atmRangeLocalId.* Specific errors include:

- No ATM range with the *atmRangeLocalId* label exists in the list.

ldpInterface getTargetedPeer *targetedPeerLocalId*

Finds the targeted peer with the label *targetedPeerLocalId* in the list and makes its data available through the use of the *ldpTargetedPeer* command. Specific errors include:

- No targeted peer with the *targetedPeerLocalId* label exists in the list.

ldpInterface getFirstAssignedAtmLabel

Retrieves the first assigned ATM label as a result of calls to *requestAssignedAtmLabels* and *getAssignedAtmLabelList.* The value of the label can be accessed through the options associated with the *ldpAssignedAtmLabel* command. Specific errors include:

- No assigned ATM labels in the list.

ldpInterface getFirstAtmLabelRange

Finds the first ATM range in the list and makes it available in the options of the *ldpAtmRange* command. Specific errors include:

- No ATM ranges in the list.

ldpInterface getFirstLearnedIpV4AtmLabel

Retrieves the first learned IPv4 ATM label as a result of calls to *requestLearnedLabels* and *getLearnedLabelList.* The value of the label can be accessed through the options associated with the *ldpLearnedIpV4AtmLabel* command. Specific errors include:

- No learned ATM labels in the list.

ldpInterface getFirstLearnedIpV4Label

Retrieves the first learned IPv4 label as a result of calls to *requestLearnedLabels* and *getLearnedLabelList.* The value of the label can be accessed through the options associated with the *ldpLearnedIpV4Label* command. Specific errors include:

- No learned labels in the list.

ldpInterface getFirstLearnedMartiniLabel

Retrieves the first learned Martini label as a result of calls to *requestLearnedLabels* and *getLearnedLabelList.* The value of the label can be accessed through the options associated with the *ldpLearnedMartiniLabel* command. Specific errors include:

- No learned labels in the list.

ldpInterface getFirstTargetedPeer

Finds the first targeted peer in the list and makes it available in the options of the *ldpTargetedPeer* command. Specific errors include:

- No targeted peers in the list.

ldpInterface getLearnedLabelList

This command must be preceded by use of the *ldpInterface requestLearnedLabelList* and followed by a calls to *ldpInterface getFirstLearnedIpV4Label/getNextLearnedIpV4Label/getFirstLearnedMartiniLabel/getNextLearnedMartiniLabel/getFirstLearnedIpV4A* This command determines whether the reading of learned labels from the protocol server has completed. This command should be called until it returns a `0' (TCL_OK), or until some suitable period of time has elapsed. The number of learned labels is available in the *numberOfLearnedLabels* option.

ldpInterface getNextAssignedAtmLabel

Retrieves the next assigned ATM label as a result of calls to *requestAssignedAtmLabels* and *getAssignedAtmLabelList.* The value of the label can be accessed through the options associated with the *ldpAssignedAtmLabel* command. Specific errors include:

- No more assigned ATM labels in the list.

ldpInterface getNextAtmLabelRange

Finds the next ATM range in the list and makes it available in the options of the *ldpAtmRange* command. Specific errors include:

- No more ATM ranges in the list.

ldpInterface getFirstLearnedIpV4AtmLabel

Retrieves the next learned IPv4 ATM label as a result of calls to *requestLearnedLabels* and *getLearnedLabelList.* The value of the label can be accessed through the options associated with the *ldpLearnedIpV4AtmLabel* command. Specific errors include:

- No more learned ATM labels in the list.

ldpInterface getNextLearnedIpV4Label

Retrieves the next learned IPv4 label as a result of calls to *requestLearnedLabels* and *getLearnedLabelList.* The value of the label can be accessed through the options associated with the *ldpLearnedIpV4Label* command. Specific errors include:

- No more learned labels in the list.

ldpInterface getNextLearnedMartiniLabel

Retrieves the next learned Martini label as a result of calls to *requestLearnedLabels* and *getLearnedLabelList.* The value of the label can be accessed through the options associated with the *ldpLearnedMartiniLabel* command. Specific errors include:

- No more learned labels in the list.

ldpInterface getNextTargetedPeer

Finds the next targeted peer in the list and makes it available in the options of the *ldpTargetedPeer* command. Specific errors include:

- *getFirstTargetedPeer*has not been called yet.

ldpInterface getTargetedPeer *targetedPeerLocalId*

Finds the targeted peer with the label *targetedPeerLocalId* in the list and makes its data available through the use of the *ldpTargetedPeer* command. Specific errors include:

- No targeted peer with the *targetedPeerLocalId* label exists in the list.

ldpInterface requestAssignedAtmLabels

Requests that the assigned ATM labels associated with this interface be retrieved from the protocol server. This command must be followed by call to *ldpInterface getAssignedAtmLabelList.*

ldpInterface requestLearnedLabels

Requests that the learned labels associated with this interface be retrieved from the protocol server. This command must be followed by call to *ldpInterface getLearnedLabelList.*

ldpInterface setDefault

Sets default values for all configuration options.

ldpInterface setAtmLabelRange *atmRangeLocalId*

Allows the options associated with an ATM range to be changed on the fly while the LDP protocol is running. The options in the *ldpAtmRange* command are set into the ATM range with id *atmRangeLocalId.* This must be followed with a call to the *ldpServer* *write* command in order to make the protocol server use the values. Specific errors include:

- No ATM range with the *atmRangeLocalId* label exists in the list.

ldpInterface setTargetedPeer *targetedPeerLocalId*

Allows the options associated with a targeted peer to be changed on the fly while the LDP protocol is running. The options in the *ldpTargetedPeer* command are set into the targeted peer with id *targetedPeerLocalId.* This must be followed with a call to the *ldpServer* *write* command in order to make the protocol server use the values. Specific errors include:

- No targeted peer with the *targetedPeerLocalId* label exists in the list.

ldpInterface getFirstLearnedMulticastIpV4Label

Retrieves the learned multicast IPv4 labels.

ldpInterface getNextLearnedMulticastIpV4Label

Retrieves the next learned multicast IPv4 labels.

EXAMPLES

The example included here relates to the use of LDP over ATM. For examples not related to ATM, see examples under *ldpServer*.

```
package req IxTclHal
set hostname test1600t
set retCode “PASS”
if {[ixConnectToChassis $hostname]} {
return “FAIL”
}
set chassis [chassis cget -id]
set card 5
set port 1
set portList [list]
port setFactoryDefaults $chassis $card $port
interfaceTable select $chassis $card $port
interfaceTable clearAllInterfaces
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress “20.20.20.2”
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress “20.20.20.1”
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description \
“ProtocolInterface - 16:01 - 1”
interfaceEntry config -macAddress “00 00 13 DE E3 F5”
interfaceEntry config -eui64Id “02 00 13 FF FE DE E3 F5”
interfaceEntry config -atmEncapsulation \
interfaceEntry config -atmMode atmBridged
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceTable addInterface
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
```

```
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress “16.1.0.2”
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress “16.1.0.1”
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description \
“ProtocolInterface - 05:01 - 2”
interfaceEntry config -macAddress “00 00 29 18 C7 46”
interfaceEntry config -eui64Id “02 00 29 FF FE 18 C7 46”
interfaceEntry config -atmEncapsulation \
atmEncapsulationLLCRoutedCLIP
interfaceEntry config -atmMode atmBridged
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 45
interfaceTable addInterface
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
ldpServer select $chassis $card $port
ldpServer clearAllRouters
ldpAtmLabelRange setDefault
ldpAtmLabelRange config -minVci 40
ldpAtmLabelRange config -minVpi 3
ldpAtmLabelRange config -maxVci 65535
ldpAtmLabelRange config -maxVpi 78
if {[ldpInterface addAtmLabelRange atmLabelRange1]} {
logMsg “ Error in adding Atm Label Range”
set retCode “FAIL”
}
ldpAtmLabelRange setDefault
ldpAtmLabelRange config -minVci 33
ldpAtmLabelRange config -minVpi 1
ldpAtmLabelRange config -maxVci 65535
ldpAtmLabelRange config -maxVpi 4095
if {[ldpInterface addAtmLabelRange atmLabelRange2]} {
logMsg " Error in adding Atm Label Range"
set retCode "FAIL"
}
ldpInterface setDefault
ldpInterface config -enable true
ldpInterface config -advertisingMode
ldpInterfaceDownstreamOnDemand
ldpInterface config -requestingMode ldpInterfaceIndependent
ldpInterface config -labelSpaceId 0
ldpInterface config -discoveryMode ldpInterfaceBasic
ldpInterface config -protocolInterfaceDescription \
"ProtocolInterface - 16:01 - 1"
ldpInterface config -enableAtmSession true
ldpInterface config -atmVcDirection atmVcBidirectional
if {[ldpRouter addInterface interface1]} {
logMsg "Error in adding interface"
set retCode "FAIL"
}
ldpAdvertiseFecRange setDefault
ldpAdvertiseFecRange config -enable true
ldpAdvertiseFecRange config -numRoutes 10
ldpAdvertiseFecRange config -maskWidth 24
ldpAdvertiseFecRange config -networkIpAddress "100.100.100.0"
ldpAdvertiseFecRange config -labelIncrementMode
ldpAdvertiseFecRangeIncrement
ldpAdvertiseFecRange config -labelValueStart 16
ldpAdvertiseFecRange config -enablePacking false
ldpRouter addAdvertiseFecRange advertiseFecRange1
ldpAdvertiseFecRange setDefault
ldpAdvertiseFecRange config -enable true
ldpAdvertiseFecRange config -numRoutes 10
ldpAdvertiseFecRange config -maskWidth 24
ldpAdvertiseFecRange config -networkIpAddress “45.45.2.0”
ldpAdvertiseFecRange config -labelIncrementMode
ldpAdvertiseFecRangeIncrement
ldpAdvertiseFecRange config -labelValueStart 16
ldpAdvertiseFecRange config -enablePacking false
ldpRouter addAdvertiseFecRange advertiseFecRange2
ldpRouter setDefault
ldpRouter config -routerId “20.20.20.1”
ldpRouter config -enable true
ldpRouter config -enableRemoteConnect true
ldpRouter config -enableL2VpnVcFecs true
ldpRouter config -enableVcGroupMatching false
ldpRouter config -enableExplicitIncludeIpFec false
if {[ldpServer addRouter router1]} {
logMsg “Error adding router1”
set retCode “FAIL”
}
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse true
protocolServer config -enableLdpService true
protocolServer set $chassis $card $port
lappend portList [list $chassis $card $port]
######### Chassis-test1600t Card-5 Port-2 #########
set port 2
port setFactoryDefaults $chassis $card $port
arpServer setDefault
arpServer config -rate 1412830
arpServer set $chassis $card $port
interfaceTable select $chassis $card $port
interfaceTable clearAllInterfaces
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress “20.20.20.1”
interfaceIpV4 config -maskWidth 24
```

```
interfaceIpV4 config -ipAddress “20.20.20.2”
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description \
“ProtocolInterface - 16:02 - 1”
interfaceEntry config -macAddress “00 00 13 DE DB B7”
interfaceEntry config -eui64Id “02 00 13 FF FE DE DB B7”
interfaceEntry config -atmEncapsulation \
atmEncapsulationLLCRoutedCLIP
interfaceEntry config -atmMode atmBridged
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceTable addInterface
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress “20.20.20.1”
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress “20.20.20.3”
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description \
“ProtocolInterface - 05:02 - 2”
interfaceEntry config -macAddress “00 00 29 18 C7 47”
interfaceEntry config -eui64Id “02 00 29 FF FE 18 C7 47”
interfaceEntry config -atmEncapsulation \
atmEncapsulationLLCRoutedCLIP
interfaceEntry config -atmMode atmBridged
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceTable addInterface
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress "16.1.0.1"
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress "16.1.0.2"
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description \
"ProtocolInterface - 05:02 - 3"
interfaceEntry config -macAddress "00 00 2E 46 40 DE"
interfaceEntry config -eui64Id "02 00 2E FF FE 46 40 DE"
interfaceEntry config -atmEncapsulation \
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmMode atmBridged
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceTable addInterface
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
ldpServer select $chassis $card $port
ldpServer clearAllRouters
ldpAtmLabelRange setDefault
ldpAtmLabelRange config -minVci 33
ldpAtmLabelRange config -minVpi 1
ldpAtmLabelRange config -maxVci 65535
ldpAtmLabelRange config -maxVpi 4095
ldpInterface addAtmLabelRange atmLabelRange1
ldpInterface setDefault
ldpInterface config -enable true
ldpInterface config -advertisingMode
ldpInterfaceDownstreamOnDemand
ldpInterface config -requestingMode ldpInterfaceIndependent
ldpInterface config -labelSpaceId 0
ldpInterface config -discoveryMode ldpInterfaceBasic
ldpInterface config -protocolInterfaceDescription \
“ProtocolInterface - 16:02 - 1”
ldpInterface config -enableAtmSession true
ldpInterface config -atmVcDirection atmVcBidirectional
ldpRouter addInterface interface1
ldpRequestFecRange setDefault
ldpRequestFecRange config -enable true
ldpRequestFecRange config -numRoutes 10
ldpRequestFecRange config -maskWidth 24
ldpRequestFecRange config -enableStaleTimer true
ldpRequestFecRange config -staleRequestTime 5
ldpRequestFecRange config -enableHopCountTlv true
ldpRequestFecRange config -hopCount 1
ldpRequestFecRange config -networkIpAddress “100.100.100.0”
ldpRequestFecRange config -nextHopPeerIp “20.20.20.1”
ldpRouter addRequestFecRange requestFecRange1
ldpRequestFecRange setDefault
ldpRequestFecRange config -enable true
ldpRequestFecRange config -numRoutes 10
ldpRequestFecRange config -maskWidth 24
ldpRequestFecRange config -enableStaleTimer true
ldpRequestFecRange config -staleRequestTime 5
ldpRequestFecRange config -enableHopCountTlv true
ldpRequestFecRange config -hopCount 1
ldpRequestFecRange config -networkIpAddress “45.45.2.0”
ldpRequestFecRange config -nextHopPeerIp “20.20.20.1”
ldpRouter addRequestFecRange requestFecRange2
ldpRouter setDefault
ldpRouter config -routerId “20.20.20.2”
ldpRouter config -enable true
ldpRouter config -enableRemoteConnect true
ldpRouter config -enableL2VpnVcFecs true
ldpRouter config -enableVcGroupMatching false
ldpRouter config -enableExplicitIncludeIpFec false
ldpServer addRouter router1
ldpInterface setDefault
ldpInterface config -enable true
ldpInterface config -advertisingMode
ldpInterfaceDownstreamOnDemand
ldpInterface config -requestingMode ldpInterfaceIndependent
ldpInterface config -labelSpaceId 0
ldpInterface config -discoveryMode ldpInterfaceBasic
ldpInterface config -protocolInterfaceDescription \
“ProtocolInterface - 05:02 - 2”
ldpInterface config -enableAtmSession true
ldpInterface config -atmVcDirection atmVcBidirectional
ldpRouter addInterface interface1
ldpRequestFecRange setDefault
ldpRequestFecRange config -enable true
ldpRequestFecRange config -numRoutes 10
ldpRequestFecRange config -maskWidth 24
ldpRequestFecRange config -enableStaleTimer true
ldpRequestFecRange config -staleRequestTime 5
ldpRequestFecRange config -enableHopCountTlv true
ldpRequestFecRange config -hopCount 1
ldpRequestFecRange config -networkIpAddress “45.45.2.0”
ldpRequestFecRange config -nextHopPeerIp “20.20.20.1”
ldpRouter addRequestFecRange requestFecRange1
ldpRouter setDefault
ldpRouter config -routerId “16.2.0.1”
ldpRouter config -enable true
ldpRouter config -enableRemoteConnect true
ldpRouter config -enableL2VpnVcFecs true
ldpRouter config -enableVcGroupMatching false
ldpRouter config -enableExplicitIncludeIpFec false
ldpServer addRouter router2
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse true
protocolServer config -enableLdpService true
protocolServer set $chassis $card $port
lappend portList [list $chassis $card $port]
ixWritePortsToHardware portList
logMsg "Starting LDP.."
ixStartLdp portList
after 10000
#Case #1 Get by local Id
set port 1
if {[ldpServer select $chassis $card $port]} {
logMsg "Error in selecting port $chassis $card $port"
set retCode "FAIL"
}
if {[ldpServer getRouter router1]} {
logMsg "Error in getting router1"
set retCode "FAIL"
}
if {[ldpRouter getInterface interface1]} {
logMsg "Error in getting interface"
set retCode "FAIL"
}
ldpAtmLabelRange setDefault
if {[ldpInterface getAtmLabelRange atmLabelRange1]} {
logMsg "Error in getting atmLabelRange1"
set retCode "FAIL"
}
ldpAtmLabelRange config -minVci 50
if {[ldpInterface setAtmLabelRange atmLabelRange1]} {
logMsg "Error in setting atmLabelRange1"
set retCode "FAIL"
}
ldpServer write
```

```
#Case #2 Getfirst/getNext
if {[ldpServer getFirstRouter]} {
logMsg "Error in getting router getFirst/getNext"
set retCode "FAIL"
}
if {[ldpRouter getFirstInterface]} {
logMsg "Error in getting interface getFirst/getNext"
set retCode "FAIL"
}
ldpAtmLabelRange setDefault
if {[ldpInterface getFirstAtmLabelRange]} {
logMsg "Error in getting atmLabelRange getFirst/getNext"
set retCode "FAIL"
}
ldpAtmLabelRange setDefault
if {[ldpInterface getNextAtmLabelRange]} {
logMsg "Error in getting atmLabelRange getFirst/getNext"
set retCode "FAIL"
}
ldpAtmLabelRange config -minVci 50
if {[ldpInterface setAtmLabelRange]} {
logMsg "Error in setting atmLabelRange getFirst/getNext"
set retCode "FAIL"
}
ldpServer write
if {[ldpServer getRouter router1]} {
logMsg "Error in getting router1"
set retCode "FAIL"
}
if {[ldpRouter getInterface interface1]} {
logMsg "Error in getting interface1"
set retCode "FAIL"
}
ldpAtmLabelRange setDefault
```

```
if {[ldpInterface getAtmLabelRange atmLabelRange2]} {
logMsg “Error in getting atmLabelRange1”
set retCode “FAIL”
}
if {[ldpAtmLabelRange cget -minVci] != 50 } {
logMsg “Wrong Values [ldpAtmLabelRange cget -minVci]”
set retCode “FAIL”
}
logMsg “getting the labels”
if {[ldpServer select $chassis $card 2]} {
logMsg “Error in selecting port $chassis $card 2”
set retCode “FAIL”
}
if {[ldpServer getFirstRouter]} {
logMsg “Error in getting router getFirst/getNext”
set retCode “FAIL”
}
if {[ldpRouter getFirstInterface]} {
logMsg “Error in getting interface getFirst/getNext”
set retCode “FAIL”
}
if {[ldpInterface requestLearnedLabels]} {
set retCode “FAIL”
}
after 5000
set timeout 10
while {[ldpInterface getLearnedLabelList] && $timeout 0} {
after 1000
incr timeout -1
}
if {![ldpInterface getLearnedLabelList]} {
if {[ldpInterface cget -numLearnedLabels] != 20} {
logMsg “Wrong number of labels”
set retCode “FAIL”
}
if {[ldpInterface getFirstLearnedIpV4AtmLabel]} {
logMsg “Error geting first ATM label”
set retCode “FAIL”
} else {
showCmd ldpLearnedIpV4AtmLabel
}
if {[ldpInterface getNextLearnedIpV4AtmLabel]} {
logMsg “Error geting next ATM label”
set retCode “FAIL”
} else {
showCmd ldpLearnedIpV4AtmLabel
}
}
logMsg “getting Assigned ATM labels”
if {[ldpServer select $chassis $card 1]} {
logMsg “Error in selecting port $chassis $card 2”
set retCode “FAIL”
}
if {[ldpServer getFirstRouter]} {
logMsg “Error in getting router getFirst/getNext”
set retCode “FAIL”
}
if {[ldpRouter getFirstInterface]} {
logMsg “Error in getting interface getFirst/getNext”
set retCode “FAIL”
}
if {[ldpInterface requestAssignedAtmLabels]} {
logMsg “Error in requestAssignedAtmLabels”
set retCode “FAIL”
}
after 5000
set timeout 10
while {[ldpInterface getAssignedAtmLabelList] && $timeout 0} {
```

```
after 1000
incr timeout -1
}
if {![ldpInterface getAssignedAtmLabelList]} {
set n 0
while {![ldpInterface getNextAssignedAtmLabel]} {
incr n
}
if {$n != 20} {
logMsg "Wrong number of assigned atm labels"
set retCode "FAIL"
}
} else {
set retCode "FAIL"
}
ixStopLdp portList
return $retCode
```

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpL2VplsMacRange

ldpL2VplsMacRange — configures a MAC range for an L2 VPN VC range.

SYNOPSIS

ldpL2VplsMacRange *subcommand options*

DESCRIPTION

The *ldpL2VplsMacRange* command holds a MAC address range, used in *ldpL2VpnVcRange*.

STANDARD OPTIONS

count

The number of MAC addresses to generate for the VC range. *(default = 1)*

enableGenerateUnique true | false

If *false*, the same MAC addresses will be associated with all of the VCIDs in the *ldpL2VpnVcRange* command. Else, each VCID generated in *ldpL2VpnVcRange* command will receive unique ascending MAC addresses. *(default = 1)*

macAddress

The first MAC address to be generated. *(default = {00 00 00 00 00 00})*

COMMANDS

The *ldpL2VplsMacRange* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands.

ldpL2VplsMacRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *ldpL2VplsMacRange* command.

ldpL2VplsMacRange config *option value*

Modify the configuration options of the ldpL2VplsMacRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ldpL2VplsMacRange.

ldpL2VplsMacRange purgeMac

Sends a set of MAC TLVs for all of the MAC addresses associated with this MAC range. This will cause the DUT to unlearn all the current MAC range.

ldpL2VplsMacRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpL2VpnInterface

ldpL2VpnInterface — configures an interface for an LDP router to be used for Layer 2 VPNs.

SYNOPSIS

ldpL2VpnInterface *subcommand options*

DESCRIPTION

The *ldpL2VpnInterface* command holds the information related to a single interface on the simulated router to be used in establishing Layer 2 VPNs. L2 VPN interfaces are added into the *ldpRouter* interface list using the *ldpRouter* *addL2VpnInterface* command. Refer to *LDP* for an overview.

This command holds a list of virtual circuit (VC) ranges.

STANDARD OPTIONS

count

The number of contiguous values of *groupId* that will be used in generating VC FECs. All FECs will be generated using the *vcId* and *count* options for each of the associated *ldpL2VpnVcRanges*, then the *groupId* will be incremented for *count*times and repeated. *(default = 1)*

enable true / false

If set, enables the use of this route range for the simulated router. *(default = false)*

groupId

The group ID associated with all VC FEC elements for this interface. *(default = 0)*

type

The type of virtual circuit. The options include:

Option	Value	Usage
l2VpnInterfaceFrameRelay	1	Frame Relay DLCI.
l2VpnInterfaceATMAAL5	2	ATM AAL5 VCC transport
l2VpnInterfaceATMXCell	3	ATM transparent cell transport
l2VpnInterfaceVLAN	4	*(default)* Ethernet VLAN
l2VpnInterfaceEthernet	5	Ethernet
l2VpnInterfaceHDLC	6	HDLC
l2VpnInterfacePPP	7	PPP
l2VpnInterfaceCEM	8	Circuit Emulation Mode
l2VpnInterfaceATMVCC	9	ATM VCC cell transport
l2VpnInterfaceATMVPC	10	ATM VPC cell transport
l2VpnInterfaceCEIP	11	

Option	Value	Usage
l2VpnInterfaceFrameRelayRFC4619	25	

COMMANDS

The **ldpL2VpnInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpL2VpnInterface addL2VpnVcRange *vcRangeLocalId*

Adds a VC range for the interface from the options in the *ldpL2VpnVcRange* command. The peer is given a local ID of *vcRangeLocalId*.

ldpL2VpnInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *ldpL2VpnInterface* command.

ldpL2VpnInterface clearAllL2VpnVcRanges

Removes all VC ranges from the interface.

ldpL2VpnInterface config *option value*

Modify the configuration options of the *ldpL2VpnInterface*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *ldpL2VpnInterface*.

ldpL2VpnInterface delL2VpnVcRange *vcRangeLocalId*

Deletes the VC range indicated by the value of *vcRangeLocalId.* Specific errors include:

- No VC range with the *vcRangeLocalId* label exists in the list.

ldpL2VpnInterface getFirstL2VpnVcRange

Finds the first VC range in the list and makes it available in the options of the *ldpTargetedPeer* command. Specific errors include:

- No items in the list.

ldpL2VpnInterface getNextL2VpnVcRange

Finds the next VC range in the list and makes it available in the options of the *ldpTargetedPeer* command. Specific errors include:

- *getFirstL2VpnVcRange*has not been called yet.

ldpL2VpnInterface getL2VpnVcRange *vcRangeLocalId*

Finds the VC range with the label *vcRangeLocalId* in the list and makes its data available through the use of the *ldpTargetedPeer* command. Specific errors include:

- No VC range with the *vcRangeLocalId* label exists in the list.

ldpL2VpnInterface sendEmptyMacTlv

For the current interface selected by use of a *getL2VpnVcRange;* i.e. getFirstL2VpnVcRange/getNextL2VpnVcRange may not be used in order to address the interface used in this command. Sends an empty TLVs for this interface. This will cause the DUT to unlearn all of its previous MACs from this interface. Specific errors include:

- No interface has been selected from the list.

ldpL2VpnInterface sendMacTlv

For the current interface selected by use of a *getL2VpnVcRange;* i.e. getFirstL2VpnVcRange/getNextL2VpnVcRange may not be used in order to address the interface used in this command. Sends a set of MAC TLVs for all of the MAC addresses associated with this interface. Specific errors include:

- No interface has been selected from the list.

ldpL2VpnInterface setL2VpnVcRange *vcRangeLocalId*

Allows the options associated with a VC range to be changed on the fly while the LDP protocol is running. The options in the *ldpTargetedPeer* command are set into the VC range with ID *vcRangeLocalId.* This must be followed with a call to the *ldpServer* *write* command in order to make the protocol server use the values. Specific errors include:

- No VC range with the *vcRangeLocalId* label exists in the list.

ldpL2VpnInterface setDefault

Sets default values for all configuration options.

DEPRECATED OPTIONS

type (ldpL2VpnInterfaceEthernetVPLS)

The ldpL2VpnInterfaceType for Ethernet Virtual Private LAN Service (VPLS).

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpL2VpnVcRange

ldpL2VpnVcRange — configures a targeted peer for LDP discovery.

SYNOPSIS

ldpL2VpnVcRange *subcommand options*

DESCRIPTION

The *ldpL2VpnVcRange* command holds information about a VC range to be associated with an LDP L2 VPN interface. VC ranges are added to a *ldpL2VpnInterface* using the *ldpL2VpnInterface* *addL2VpnVcRange* subcommand. A list of MAC ranges is associated with this command; individual elements are built using the *ldpL2VplsMacRange* command and added using the *addVplsMacRange* subcommand.

STANDARD OPTIONS

General Options

ceIpAddress

The IP address of attached CE endpoint. If IP Type is set to IPv4, then the default is 0.0.0.0, and if the IP type is set to IPv6, then the default is 0:0:0:0:0:0:0:0.

ceIpStep

The increment step to be added to each additional CE endpoints in the range of CE endpoints.

enablePwStatusTlv

Enables the use of PW status TLV in notification messages to notify the PW status.

enableSendPwStatus

If checked, it enables a notification message with a PW status for the corresponding PW.

downStartInterval

The duration in time after session becomes up and a notification message being sent to make the session down. *(default = 30 sec)*

downInterval

Time interval for which the PW status will remain down. *(default = 60 sec)*

upInterval

Time interval for which the PW status will remain up. *(default = 30 sec)*

repeatCount

The number of times to repeat Up Interval and Down Interval. *(default = 1)*

pwStatusCode

Editable dropdown to denote the PW status. This field is editable and the range is from 0x00000001 0xFFFFFFFF. The options are as follows:

Option	Value
PW not forwarding	0
AC Rx Fault	1
AC Tx Fault	2
PW Rx Fault	3
PW Tx Fault	4

count

The number of times that the *vcId* will be incremented in order to generated FECs. Also see the *groupId* and *count* options of the *ldpL2VpnInterface* to see how VC IDs are generated. *(default = 1)*

description

An interface description string, if *enableDescription* is *true*. *(default = "")*

enable true | false

Enables the use of this entry. *(default = false)*

enableCBit true | false

Enables the generation of a control word in the VC. *(default = false)*

enableDescription true | false

Enables the generation of an interface description in the VC. *(default = false)*

enableMtu true | false

Enables the generation of an MTU interface parameter field, using the value in *mtuSize. (default = false)*

enablePacking true | false

Enables the packing of multiple label mappings on sending labels and sending withdrawals into a single PDU*.* This is only applicable to sessions established in the downstream unsolicited mode. *(default = false)*

ipRange

This is applicable only if the L2 interface type is IP. This is used to denote that the IP addresses of the simulated hosts are the IP virtual circuit CE endpoint. This is used only for traffic generation on the IP virtual circuit.

labelMode

The manner in which labels are assigned to generated VCs. The options include:

Option	Value	Usage
ldpL2VpnVcFixedLabel	0	Use the same label for all VCs.
ldpL2VpnVcIncrementLabel	1	(default) Increment the label by one for each VC.

labelValueStart

The initial label value used in the generated VC. *(default = 16)*

macVlanRange

A VC range of MAC addresses for the L2 interface.

mtu

(in octets) The 2-octet value for the maximum Transmission Unit (MTU).

mtuSize

The value of the MTU to be included if *enableMtu* is set to *true. (default = 0)*

peerAddress

The IPv4 address of the LDP router which is the peer for this VC range. *(default = 0.0.0.0)*

vcId

The virtual circuit ID, together with the value of the type defined in the *ldpL2VpnInterface* command, identifies a unique VC. (default = 10)

vcIdStep

The step value applied between uses of vcId. (default = 1)

capableOfReassembly

If true, the VC range is capable of reassembly.

cas

This is available only for vc type 0x0017.The aoptions include:

Option	Value
an E1 trunk	01
a T1/ESF trunk	10
a T1 SF trunk	11

frequency

The frequency of the VC range.

includeSsrc

If true, the SSRC is enabled.

ssrc

The positive value for SSRC.

sp

Editable dropdown.

The options are:

Option	Value
ldpL2VpnVcHexVal1	0x00
ldpL2VpnVcHexVal2	0x01
ldpL2VpnVcHexVal3	0x02
ldpL2VpnVcHexVal4	0x03

timestampMode

Editable dropdown

The options are:

- Absolute
- Differential

includeTdmOption

If true, the TDM option is enabled.

includeRtpHeader

If true, the RTP header is enabled.

tdmBitrate

The integer value fro TDM bitrate.

includeTdmBitrate

If true, TDM bitrate option is included.

tdmDataSize

The integer value for the data size.

includeTdmPayload

If true, the TDM data size is enabled.

payloadType

The integer value for Payload type.

The acceptable range is 0x00 0x7F.

ATM Related Options

enableAtm true | false

Enables the generation of an interface parameter field with the maximum number of concatenated ATM cells. *(default = 0)*

maxNumAtmCells

The maximum number of concatenated ATM cells, if *enableAtm* is *true. (default = 1)*

CEM Related Options

cemOptions

The CEM options, if *enableCemOptions* is *true. (default = 0)*

cemPayloadBytes

The number of CEM payload bytes, if *enableCemPayload* is *true. (default = 48)*

enableCemOptions true | false

Enables the generation of an interface parameter field with CEM options. *(default = 0)*

enableCemPayload true | false

Enables the generation of an interface parameter field with the number of CEM payload bytes. *(default = 0)*

COMMANDS

The *ldpL2VpnVcRange* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpL2VpnVcRange addVcIpRange *ipRangeLocalId*

Adds the VC IP range described in the *Running H/F 2* command to the list associated with the port. The entry in the list is given an identifier of *ipRangeLocalId.*

ldpL2VpnVcRange addVcMacVlanRange *macVlanRangeLocalId*

Adds the VC MAC VLAN range described in the *Running H/F 2* command to the list associated with the port. The entry in the list is given an identifier of *macVlanRangeLocalId.*

ldpL2VpnVcRange addVplsMacRange *macRangeLocalId*

Adds the MAC range described in the *ldpL2VplsMacRange* command to the list associated with the port. The entry in the list is given an identifier of *macRangeLocalId*. Specific errors are:

- The parameters in *ldpL2VplsMacRange* are invalid.
- An entry with this *macRangeLocalId* exists already in the list.

ldpL2VpnVcRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *ldpL2VpnVcRange* command.

ldpL2VpnVcRange clearAllVcIpRanges

Deletes all the Vc IP ranges in the list.

ldpL2VpnVcRange clearAllVcMacVlanRanges

Deletes all the VC MAC VLAN ranges in the list.

ldpL2VpnVcRange clearAllVplsMacRanges

Deletes all the MAC ranges in the list.

ldpL2VpnVcRange config *option value*

Modify the configuration options of the ldpL2VpnVcRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ldpL2VpnVcRange.

ldpL2VpnVcRange delVcIpRange *ipRangeLocalId*

Deletes the VC IP range from the list associated with the port.

ldpL2VpnVcRange delVcMacVlanRange *macVlanRangeLocalId*

Deletes the VC MAC VLAN range from the list associated with the port.

ldpL2VpnVcRange delVplsMacRange *macRangeLocalId*

Deletes the MAC range that has an identifier of *macRangeLocalId*. Specific errors are:

- There is no entry with this *macRangeLocalId* in the list.

ldpL2VpnVcRange getFirstVcIpRange

Access the first VC IP range in the list. The results may be accessed using the *ldpL2VplsMacRange* command. Specific errors are:

- There are no elements in the list.

ldpL2VpnVcRange getFirstVcMacVlanRange

Access the first VC VLAN MAC range in the list The results may be accessed using the *Running H/F 2* command. Specific errors are:

- There are no elements in the list.

ldpL2VpnVcRange getFirstVplsMacRange

Access the first MAC range in the list. The results may be accessed using the *Running H/F 2* command. Specific errors are:

- There are no elements in the list.

ldpL2VpnVcRange getNextVcIpRange

Access the next VC IP range in the list. The results may be accessed using the *Running H/F 2* command.

ldpL2VpnVcRange getNextVcMacVlanRange

Access the next VC MAC VLAN range in the list The results may be accessed using the *Running H/F 2* command.

ldpL2VpnVcRange getNextVplsMacRange

Access the next MAC range in the list The results may be accessed using the *ldpL2VplsMacRange* command. Specific errors are:

- *ldpL2VpnVcRange getFirstVplsMacRange* has not been called.
- There is no more entries in the list.

ldpL2VpnVcRange getVcIpRange *ipRangeLocalId*

Accesses the VC IP range described in the *Running H/F 2* command to the list associated with the port. The entry in the list is given an identifier of *ipRangeLocalId.*

ldpL2VpnVcRange getVcMacVlanRange *macVlanRangeLocalId*

Accesses the VC MAC VLAN range described in the *Running H/F 2* command to the list associated with the port. The entry in the list is given an identifier of *macVlanRangeLocalId.*

ldpL2VpnVcRange purgeVc

Sends an empty TLV for the MAC addresses associated with this VC range. This will cause the DUT to unlearn all of its previous MACs from this interface. Specific errors include:

- No interface has been selected from the list.

ldpL2VpnVcRange setDefault

Sets default values for all configuration options.

ldpL2VpnVcRange setVcIpRange *ipRangeLocalId*

Sets the VC IP range described in the *Running H/F 2* command to the list associated with the port. The entry in the list is given an identifier of *ipRangeLocalId.*

ldpL2VpnVcRange setVcMacVlanRange *macVlanRangeLocalId*

Sets the VC MAC VLAN range described in the *Running H/F 2* command to the list associated with the port. The entry in the list is given an identifier of *macVlanRangeLocalId.*

DEPRECATED COMMANDS

ldpL2VpnVcRange sendEmptyMacTlv

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpLearnedIpV4AtmLabel

ldpLearnedIpV4AtmLabel — accesses learned ATM IPv4 LDP labels.

SYNOPSIS

ldpLearnedIpV4AtmLabel *subcommand options*

DESCRIPTION

The *ldpLearnedIpV4AtmLabel* command holds an element of the LDP ATM learned label list obtained in the *ldpInterface* command.

STANDARD OPTIONS

fec

(Read-only) The prefix associated with the label.

fecPrefixLength

(Read-only) The prefix length for the value in *fec*.

label

(Read-only) The value of the label itself.

labelSpaceId

(Read-only) Label space ID associated with the learned label.

peerIpAddress

(Read-only) The IP address of the peer that the label was learned from.

vci

(Read-only) The VCI of the learned label.

vpi

(Read-only) The VPI of the learned label.

COMMANDS

The *ldpLearnedIpV4AtmLabel* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpLearnedIpV4AtmLabel cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *discoveredList* command.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpLearnedIpV4Label

ldpLearnedIpV4Label — accesses learned IPv4 LDP labels.

SYNOPSIS

ldpLearnedIpV4Label *subcommand options*

DESCRIPTION

The *ldpLearnedIpV4Label* command holds an element of the LDP learned label list obtained in the *ldpInterface* command.

STANDARD OPTIONS

fec

(Read-only) The prefix associated with the label.

fecPrefixLength

(Read-only) The prefix length for the value in *fec*.

label

(Read-only) The value of the label itself.

labelSpaceId

(Read-only) Label space ID associated with the learned label.

peerIpAddress

(Read-only) The IP address of the peer that the label was learned from.

COMMANDS

The *ldpLearnedIpV4Label* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpLearnedIpV4Label cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *discoveredList* command.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpLearnedMartiniLabel

ldpLearnedMartiniLabel — accesses learned LDP Martini labels.

SYNOPSIS

ldpLearnedMartiniLabel *subcommand options*

DESCRIPTION

The *ldpLearnedMartiniLabel* command holds an element of the LDP learned Martini label list obtained in the *ldpInterface* command.

STANDARD OPTIONS

cBit

(Read-only) The C-bit state, indicating the presence of a control word in encapsulated packets.

cemOptions

(Read-only) The CEM options, if received. Otherwise, 0.

cemPayloadBytes

(Read-only) The number of CEM payload bytes, if received. Otherwise, 0.

description

(Read-only) The interface description, if received. Otherwise, "".

discoveredCeAddress

(Read-only) If the L2 interface type for the VC whose learned information is seen is IP, this field indicates the learned IP address of the remote end CE of the IP Virtual Circuit.

pwStatusReceived

(Read Only) Denotes the exact PW status for the PW state, as received in the notification message for PW state down.

For PW state up, field is 0.

The two options are of following:

Option	Value	Usage
Local PW Sub-Status	0	Reflects the status carried in the PW status notification received from the peer.
Peer PW Sub-Status	1	Reflects the status carried in the PW status last sent to the peer.

NOTE: Displayed for Extended Martini Discovery Mode only.

groupId

(Read-only) The group ID associated with the VC.

label

(Read-only) The value of the label itself.

labelSpaceId

(Read-only) Label space ID associated with the learned label.

localPwSubStatus

(Read-only) Reflects the status carried in the PW status notification received from the peer. If up, 0.

maxNumAtmCells

(Read-only) The maximum number of contiguous ATM cells, if received. Otherwise, 0.

mtuSize

(Read-only) The received MTU size, if received. Otherwise, 0.

peerIpAddress

(Read-only) The IP address of the peer that the label was learned from.

peerPwSubStatus

(Read-only) Reflects the status carried in the PW status last sent to the peer. If up, 0.

pseudoWireState

(Read-only) The pseudo-wire status of the connection. Either *true* or *false.*

vcId

(Read-only) The virtual circuit's ID.

vcType

(Read-only) The type of virtual circuit. One of the following options:

Option	Value	Usage
l2VpnInterfaceFrameRelay	1	Frame Relay DLCI.
l2VpnInterfaceATMAAL5	2	ATM AAL5 VCC transport
l2VpnInterfaceATMXCell	3	ATM transparent cell transport
l2VpnInterfaceVLAN	4	*(default)* Ethernet VLAN
l2VpnInterfaceEthernet	5	Ethernet
l2VpnInterfaceHDLC	6	HDLC
l2VpnInterfacePPP	7	PPP

Option	Value	Usage
l2VpnInterfaceCEM	8	Circuit Emulation Mode
l2VpnInterfaceATMVCC	9	ATM VCC cell transport
l2VpnInterfaceATMVPC	10	ATM VPC cell transport

cas

This is available only for vc type 0x0017.The aoptions include:

Option	Value
an E1 trunk	01
a T1/ESF trunk	10
a T1 SF trunk	11

frequency

The frequency of the VC range.

ssrc

The positive value for SSRC.

sp

Editable dropdown.

The options are:

Option	Value
ldpL2VpnVcHexVal1	0x00
ldpL2VpnVcHexVal2	0x01
ldpL2VpnVcHexVal3	0x02
ldpL2VpnVcHexVal4	0x03

timestampMode

Editable dropdown.

The options are:

- Absolute
- Differential

includeRtpHeader

If true, the RTP header is enabled.

tdmBitrate

The integer value fro TDM bitrate.

payloadType

The integer value for Payload type.

The acceptable range is 0x00 0x7F.

COMMANDS

p The *ldpLearnedMartiniLabel* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpLearnedMartiniLabel cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *discoveredList* command.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel*, *ldpLearnedIpV4Label*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpRequestFecRange

ldpRequestFecRange — configures an FEC range request for an LDP router.

SYNOPSIS

ldpRequestFecRange *subcommand options*

DESCRIPTION

The *ldpRequestFecRange* holds the information related to a single FEC range request associated with download on demand advertising mode. Requests are added into the *ldpRouter* request FEC list using the *ldpRouter addRequestFecRange* command. Refer to ldpInterface21 for an overview.

STANDARD OPTIONS

enable true / false

If set, enables the use of this request for the simulated router. *(default = false)*

enableHopCountTlv true / false

If set, the setup messages used to create an LSP will contain a Hop Count TLV. This TLV tracks the number of LSP hops during the LSP setup process. This TLV is optional except in the case of ATM links, which **require** this TLV. *(default = true)*

enableStaleTimer true / false

If set, enables the Stale Request Timer. *(default = true)*

hopCount

If *enableHopCountTlv* is *true*, then this is the number of hops along the path of the LSP. *(default = 1)*

nextHopPeerIp

The IPv4 address of the LDP peer that is the next hop router on this path. *(default = "0.0.0.0")*

networkIpAddress

The first FEC network address in the range. *(default = "0.0.0.0")*

numRoutes

The number of routes to request starting at the *networkIpAddress*. *(default = 1)*

maskWidth

The mask applied to *networkIpAddress*. *(default = 24)*

staleRequestTime

If *enableStaleTimer* is *true*, this is the value of the Stale Request Timer, in seconds. When the Ixia-emulated LDP peer sends Label Request messages to the DUT, this timer is set. If no response is received from the DUT within the specified time, the Ixia peer considers the requests timed out and "Stale", and deletes the records of the previously sent messages. The valid range is 1 to 65,535 seconds. *(default = 300)*

COMMANDS

The **ldpRequestFecRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpRequestFecRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ldpRequestFecRange** command.

ldpRequestFecRange config *option value*

Modify the configuration options of the *ldpRequestFecRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *ldpRequestFecRange*.

ldpRequestFecRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel, ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpRouter

ldpRouter — configures an LDP router.

SYNOPSIS

ldpRouter *subcommand options*

DESCRIPTION

The *ldpRouter* command represents a simulated router. In addition to some identifying options, it holds five lists for the router:

- Advertise FEC Range — FECs to be advertised by the simulated router, constructed in the *ldpAdvertiseFecRange* command.
- Request FEC Range — FECs to be requested of upstream peers, to be used in download on demand advertising mode.
- Interfaces — router interface, constructed in the *ldpInterface* command.
- Layer 2 VPN interfaces — router interfaces which participate in L2 VPNs, constructed in the *ldpL2VpnInterface* command.
- Explicit Include List — an optional list of IP FEC's used to filter received FECs. This allows the simulated router to ignore all other FECs.

Routers defined in this command are added to an *ldpServer* using the *ldpServer addRouter* command. Refer to *ldpRouter* for an overview of this command.

STANDARD OPTIONS

enable ***true / false***

Enables the use of this router in the simulated LDP network. *(default = false)*

enableExplicitInclude IpFec true / false

Enables the use of the explicit include IP FEC list which filters received labels. *(default = false)*

enableL2VpnVcFecs true / false

Enables the use of Layer 2 Virtual Circuit FECs for this router. *(default = true)*

enableRemoteConnect true / false

Enables LDP routers not part of the local multicast network to connect to the simualted router. *(default = true)*

enableVcGroup Matching true / false

This option enables the matching of received Martini labels by group ID as well as VCID. If this option is *true*, then the *pseudoWireState* option in the *ldpLearnedMartiniLabel* command will be *true* only if both the group ID and VCID of a learned label matches. *(default = false)*

routerId

The ID of the router, expressed as an IPv4 address. *(default = 0.0.0.0)*

COMMANDS

The **ldpRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpRouter addAdvertiseFecRange *advertiseFecRangeLocalId*

Adds the route range described in the *ldpAdvertiseFecRange* command to the list of route ranges associated with the router. The range's entry in the list is given an identifier of *advertiseFecRangeLocalId*. Specific errors are:

- The parameters in *ldpAdvertiseFecRange* are invalid.
- A router with this *advertiseFecRangeLocalId* exists already in the list.

ldpRouter addExplicitIncludeIpFec *explicitIncludeIpFecId*

Adds the explicit include FEC described in the *ldpExplicitIncludeIpFec* command to the explicit include FECs associated with the router. The entry in the list is given an identifier of *explicitIncludeIpFecId*. Specific errors are:

- The parameters in *ldpExplicitIncludeIpFec* are invalid.
- A router with this *explicitIncludeIpFecId* exists already in the list.

ldpRouter addInterface *interfaceLocalId*

Adds the router interface described in the *ldpInterface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *interfaceLocalId*. Specific errors are:

- The parameters in *ldpInterface* are invalid.
- An interface with this *interfaceLocalId* exists already in the list.

ldpRouter addL2VpnInterface *l2VpnInterfaceLocalId*

Adds the router interface described in the *ldpL2VpnInterface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *l2VpnInterfaceLocalId*. Specific errors are:

- The parameters in *ldpL2VpnInterface* are invalid.
- An interface with this *l2VpnInterfaceLocalId* exists already in the list.

ldpRouter addRequestFecRange *requestFecRangeLocalId*

Adds the FEC range request described in the *ldpRequestFecRange* command to the list of requests associated with the router. The request's entry in the list is given an identifier of *l2VpnInterfaceLocalId*. Specific errors are:

- The parameters in *ldpRequestFecRange* are invalid.
- An request with this *l2VpnInterfaceLocalId* exists already in the list.

ldpRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ldpRouter** command.

ldpRouter clearAllAdvertiseFecRanges

Deletes all of the FEC ranges.

ldpRouter clearAllExplicitIncludeIpFecs

Deletes all of the explicit include FECs.

ldpRouter clearAllInterfaces

Deletes all of the router interfaces.

ldpRouter clearAllL2VpnInterfaces

Deletes all of the L2 VPN interfaces.

ldpRouter clearAllRequestFecRanges

Deletes all of the request FEC ranges.

ldpRouter config *option value*

Modify the configuration options of the ldpRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ldpRouter.

ldpRouter delAdvertiseFecRange *AdvertiseFecRangeLocalId*

Deletes the FEC range with an identifier of *advertiseFecRangeLocalId*. Specific errors are:

- No FEC with this *advertiseFecRangeLocalId* exists in the list.

ldpRouter delExplicitIncludeIpFec *explicitIncludeIpFecId*

Deletes the explicit include FEC with an identifier of *explicitIncludeIpFecId*. Specific errors are:

- No item with this *explicitIncludeIpFecId* exists in the list.

ldpRouter ldpRouterLearnedIpV4Label requestLdpBgpAdVplsLearnedInfo

Requests the per router BgpAdVpls learned info.

If true, the value returned is 0.

ldpRouter ldpRouterLearnedIpV4Label getLdpBgpAdVplsLearnedInfo

Gets the router level BGP AD Learned Info after requestLdpBgpAdVplsLearnedInfo has been called.

If true, the value returned is 0.

ldpRouterldpRouterLearnedIpV4Label getFirstLdpBgpAdVplsLearnedInfo

Gets the first record of the retrieved learned info.

If true, the value returned is 0.

ldpRouter ldpRouterLearnedIpV4Label getNextLdpBgpAdVplsLearnedInfo

Gets the subsequent record of the retrieved learned info.

If true, the value returned is 0.

ldpRouter delInterface *interfaceLocalId*

Deletes the router interface with an identifier of *interfaceLocalId*. Specific errors are:

- No interface with this *interfaceLocalId* exists in the list.

ldpRouter delL2VpnInterface *l2VpnInterfaceLocalId*

Deletes the router interface with an identifier of *l2VpnInterfaceLocalId*. Specific errors are:

- An interface with this *l2VpnInterfaceLocalId* does not exist in the list.

ldpRouter delRequestFecRange *requestFecRangeLocalId*

Deletes the FEC request with an identifier of *requestFecRangeLocalId*. Specific errors are:

- An interface with this *requestFecRangeLocalId* does not exist in the list.

ldpRouter getAdvertiseFecRange *advertiseFecRangeLocalId*

Accesses the range's entry in the list with an identifier of *advertiseFecRangeLocalId*. The router range is accessed in the *ldpAdvertiseFecRange* command. Specific errors are:

- An FEC with this *advertiseFecRangeLocalId* does not exist in the list.

ldpRouter getExplicitIncludeIpFec *explicitIncludeIpFecId*

Accesses the entry in the list with an identifier of *explicitIncludeIpFecId*. The explicit include FEC is accessed in the *ldpExplicitIncludeIpFec* command. Specific errors are:

- An item with this *explicitIncludeIpFecId* does not exist in the list.

ldpRouter getFirstAdvertiseFecRange

Access the first FEC range in the list. The results may be accessed using the *ldpAdvertiseFecRange* command. Specific errors are:

- There are no FEC ranges in the list.

ldpRouter getFirstExplicitIncludeIpFec

Access the first item in the list. The results may be accessed using the *ldpExplicitIncludeIpFec* command. Specific errors are:

- There are no items in the list.

ldpRouter getFirstInterface

Access the first interface in the list. The results may be accessed using the *ldpInterface* command. Specific errors are:

- *ldpServer select* has not been called.
- There are no interfaces in the list.

ldpRouter getFirstL2VpnInterface

Access the first L2 VPN interface in the list. The results may be accessed using the *ldpL2VpnInterface* command. Specific errors are:

- *ldpServer select* has not been called.
- There are no L2 VPN interfaces in the list.

ldpRouter getFirstRequestFecRange

Access the first request FEC range in the list. The results may be accessed using the *ldpRequestFecRange* command. Specific errors are:

- *ldpServer select* has not been called.
- There are no items in the list.

ldpRouter getInterface *interfaceLocalId*

Accesses the interface's entry in the list with an identifier of *interfaceLocalId*. The router interface is accessed in the *ldpInterface* command. Specific errors are:

- An interface with this *interfaceLocalId* does not exist in the list.

ldpRouter getNextAdvertiseFecRange

Access the next FEC range in the list. The results may be accessed using the *ldpAdvertiseFecRange* command. Specific errors are:

- *ldpRouter getFirstAdvertiseFecRange* has not been called.
- There are no more FEC ranges in the list.

ldpRouter getNextExplicitIncludeIpFec

Access the next item in the list. The results may be accessed using the *ldpExplicitIncludeIpFec* command. Specific errors are:

- *ldpRouter getFirstExplicitIncludeIpFec* has not been called.
- There are no more items in the list.

ldpRouter getNextInterface

Access the next interface in the list. The results may be accessed using the *ldpInterface* command. Specific errors are:

- There is no more interfaces in the list.

ldpRouter getNextL2VpnInterface

Access the next L2 VPN interface in the list. The results may be accessed using the *ldpL2VpnInterface* command. Specific errors are:

- *ldpRouter select* has not been called.
- There are no more L2 VPN interfaces in the list.

ldpRouter getNextRequestFecRange

Access the next request FEC range in the list. The results may be accessed using the *ldpRequestFecRange* command. Specific errors are:

- *ldpRouter select* has not been called.
- There are no more items in the list.

ldpRouter getRequestFecRange *requestFecRangeLocalId*

Accesses the FEC request in the list with an identifier of *requestFecRangeLocalId*. The router interface is accessed in the *ldpRequestFecRange* command. Specific errors are:

- An interface with this *requestFecRangeLocalId* does not exist in the list.

ldpRouter setDefault

Sets default values for all configuration options.

ldpRouter setAdvertiseFecRange *interfaceLocalId*

Sets the values for the FEC range's entry in the list with an identifier of *interfaceLocalId* based on changes made through the *ldpAdvertiseFecRange* command. This command should be used to change a running configuration and must be followed by an *ldpServer write* command in order to send these changes to the protocol server. Specific errors are:

- An FEC with this *interfaceLocalId* does not exist in the list.

ldpRouter setInterface *interfaceLocalId*

Sets the values for the interface's entry in the list with an identifier of *interfaceLocalId* based on changes made through the *ldpInterface* command. This command can be used to change a running configuration and must be followed by an *ldpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A port has not been selected via the *ldpServer select* command.
- The port is owned by another user.
- Invalid interface configuration.
- There is no object with this ID.

ldpRouter setL2VpnInterface *l2VpnInterfaceLocalId*

Sets the values for the L2 VPN interface's entry in the list with an identifier of *l2VpnInterfaceLocalId* based on changes made through the *ldpL2VpnInterface* command. This command can be used to change a running configuration and must be followed by an *ldpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A port has not been selected via the *ldpServer select* command.
- The port is owned by another user.
- Invalid interface configuration.
- There is no object with this ID.

ldpRouter setRequestFecRange *requestFecRangeLocalId*

Sets the values for the request FEC range in the list with an identifier of *requestFecRangeLocalId* based on changes made through the *ldpRequestFecRange* command. This command can be used to change a running configuration and must be followed by an *ldpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A port has not been selected via the *ldpServer select* command.
- The port is owned by another user.
- Invalid interface configuration.
- There is no object with this ID.

ldpRouter addMulticastLeafRange multicastLeafRange1

Adds a new multicast leaf range to the ldp router.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel, ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpServer

ldpServer — accesses the LDP component of the protocol server for a particular port.

SYNOPSIS

ldpServer *subcommand options*

DESCRIPTION

The *ldpServer* command is necessary in order to access the LDP protocol server for a particular port. The *select* subcommand **must** be used before all other LDP commands. Refer to *ldpServer* for an overview.

STANDARD OPTIONS

enableDiscardSelf AdvertiseFecs true | false

Discard learned labels from the DUT that match any of the enabled configured IPv4 FEC ranges. This flag is only set when LDP is started. If it is to be changed later, LDP should be stopped, changed and set via *ldpServer set* and then LDP restarted. *(default = true)*

enableHelloJitter true | false

If *true*, Hello Jitter is enabled on this Ixia-emulated LDP port. When there are a large number of LDP adjacencies configured for the port, the Hello Jitter feature allows the LDP Hellos to be sent at slightly different times, to smooth out the traffic flow. The variation from the configured Hello Interval varies between +/- 15% of the configured value. *(default = true)*

enableLabel ExchangeOverLsp true | false

Enables protocol sessions to run over established LSPs.

If true, when a protocol packet is transmitted by an Ixia port and the IP details match an established LSP, the packet is MPLS encapsulated.

The MPLS label is set to the value learned from the LSP.

helloHoldTime

The amount of time, expressed in seconds, that an LDP adjacency will be maintained in the absence of a HELLO message. *(default = 15)*

helloInterval

The amount of time, expressed in seconds, between transmitted HELLO messages. *(default = 5)*

keepAliveHoldTime

The amount of time, expressed in seconds, that an LDP adjacency will be maintained in the absence of a PDU received from the adjacency. *(default = 30)*

keepAliveInterval

The amount of time, expressed in seconds, between keep-alive messages sent from simulated routers to their adjacency in the absence of other PDUs sent to the adjacency. *(default = 10)*

targetedHelloHoldTime

The amount of time, expressed in seconds, that an LDP adjacency will be maintained for a targeted peer in the absence of a HELLO message. *(default = 15)*

targetedHelloInterval

The amount of time, expressed in seconds, between transmitted HELLO messages to targeted peers. *(default = 45)*

COMMANDS

The **ldpServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpServer addRouter *routerLocalId*

Adds the LDP router described in the *ldpRouter* command to the list of routers associated with the port. The router's entry in the list is given an identifier of *routerLocalId*. Specific errors are:

- ldpServer select has not been called.
- The router parameters in *ldpRouter* are invalid.
- A router with this *routerLocalId* exists already in the list.

ldpServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ldpServer** command.

ldpServer clearAllRouters

Deletes all the LDP routers in the list. Specific errors are:

- ldpServer select has not been called.
- There is no router with this *routerLocalId* in the list.

ldpServer config *option value*

Modify the configuration options of the **ldpServer**. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for **ldpServer**.

ldpServer delRouter *routerLocalId*

Deletes the LDP router described that has an identifier of *routerLocalId*. Specific errors are:

- ldpServer select has not been called.
- There is no router with this *routerLocalId* in the list.

ldpServer generateIpV4Streams *chasID cardID portID action*

Generate streams creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each enabled route range associated with each LDP router; each stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

ldpServer get *chasID cardID portID*

Gets the current LDP server configuration of the port with ID *portID* on card *cardID*, chassis *chasID*. Call this command before calling *ldpServer* cget *option* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.
- Invalid port number.

ldpServer getFirstRouter

Access the first LDP routerin the list. The results may be accessed using the *ldpRouter* command. Specific errors are:

- *ldpServer* select has not been called.
- There are no routers in the list.

ldpServer getNextRouter

Access the next LDP router in the list. The results may be accessed using the *ldpRouter* command. Specific errors are:

- *ldpServer select* has not been called.
- *ldpServer getFirstRouter* has not been called.
- There is no more routers in the list.

ldpServer getRouter *routerLocalId*

Access the LDP router with an identifier of *routerLocalId.* The results may be accessed using the *ldpRouter* command. Specific errors are:

- *ldpServer select* has not been called.
- There is no router with this *routerLocalId* in the list.

ldpServer select *chasID cardID portID*

Accesses the LDP component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The LDP protocol package has not been installed.
- Invalid port specified.

ldpServer set

Sets the configuration of the LDP server in IxHAL for the port selected with the *select* subcommand by reading the configuration option values set by the *ldpServer configoption value* command and subsidiary commands.Specific errors are:

- No connection to a chassis.
- Invalid port number.

ldpServer setRouter *routerLocalId*

Sets the values for the router's entry in the list with an identifier of *routerLocalId* based on changes made through the *ldpRouter* command. This command should be used to change a running configuration and must be followed by an *ldpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *routerLocalId* does not exist in the list.

ldpServer write

Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card and port. Before using this command, use the *ldpServer select* command to select the port.

EXAMPLES

```
package req IxTclHal

set hostname loopback

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis
```

```
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts “Could not connect to $host”
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chassis [ixGetChassisID $host]
set card 4
set port 1
set streamId 1
set portList [list [list $chassis $card $port]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
# Make an interface table entry for the port
interfaceTable select $chassis $card $port
interfaceTable clearAllInterfaces
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress {192.168.18.1}
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress {192.168.18.2}
```

```
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable false
interfaceEntry config -description {1 - 04:01
ProtocolInterface}
interfaceEntry config -macAddress {00 00 00 93
BE 34}
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 0
interfaceTable addInterface
# Select the port for ldpServer to operate on
ldpServer select $chassis $card $port
ldpServer clearAllRouters
# Create a targeted peer
ldpTargetedPeer setDefault
ldpTargetedPeer config -enable true
ldpTargetedPeer config -ipAddress
{192.168.18.12}
# And attach it to an interface
ldpInterface addTargetedPeer targetedPeer1
# Specify the rest of the interface
ldpInterface setDefault
ldpInterface config -enable true
ldpInterface config -advertisingMode
ldpInterfaceDownstreamUnsolicited
ldpInterface config -requestingMode
ldpInterfaceIndependent
ldpInterface config -labelSpaceId 0
ldpInterface config -discoveryMode
ldpInterfaceExtendedMartini
ldpInterface config -protocolInterfaceDescription {1
- 04:01 ProtocolInterface}
# And add the interface to the router
ldpRouter addInterface interface1
```

```
# Create an advertised range
ldpAdvertiseFecRange setDefault
ldpAdvertiseFecRange config -enable true
ldpAdvertiseFecRange config -numRoutes 16
ldpAdvertiseFecRange config -maskWidth 16
ldpAdvertiseFecRange config -networkIpAddress {10.1.0.0}
ldpAdvertiseFecRange config -labelIncrementMode
ldpAdvertiseFecRangeIncrement
ldpAdvertiseFecRange config -labelValueStart 42
# And add the range to the router
ldpRouter addAdvertiseFecRange advertiseFecRange1
# Build up the VPN VC range
ldpL2VpnVcRange setDefault
ldpL2VpnVcRange config -enable true
ldpL2VpnVcRange config -peerAddress {192.168.18.2}
ldpL2VpnVcRange config -vcId 14
ldpL2VpnVcRange config -count 1
ldpL2VpnVcRange config -enableCBit false
ldpL2VpnVcRange config -enableMtu true
ldpL2VpnVcRange config -mtuSize 0
ldpL2VpnVcRange config -enableDescription true
ldpL2VpnVcRange config -description {Site a}
ldpL2VpnVcRange config -labelValueStart 16
ldpL2VpnVcRange config -labelMode
ldpL2VpnVcIncrementLabel
# And add the VC range to the VC interface
ldpL2VpnInterface addL2VpnVcRange l2VpnVcRange1
# And specify the rest of the VPN interface
ldpL2VpnInterface setDefault
ldpL2VpnInterface config -enable true
ldpL2VpnInterface config -groupId 100
ldpL2VpnInterface config -count 1
ldpL2VpnInterface config -type
l2VpnInterfaceVLAN
```

```
# And add the VPN interface to the router
ldpRouter addL2VpnInterface l2VpnInterface1
# Finish specifying the LDP router
ldpRouter setDefault
ldpRouter config -routerId {192.168.1.2}
ldpRouter config -enable true
ldpRouter config -enableRemoteConnect true
ldpRouter config -enableL2VpnVcFecs true
# And add the router to the server
ldpServer addRouter router1
# Finish setting up the server
ldpServer setDefault
ldpServer config -enableDiscardSelfAdvertiseFecs
true
ldpServer set
# And enable the protocol
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enableLdpService true
protocolServer set $chassis $card $port
ixWritePortsToHardware portList
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel, ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpTargetedPeer*

NAME - ldpTargetedPeer

ldpTargetedPeer — configures a targeted peer for LDP discovery.

SYNOPSIS

ldpTargetedPeer *subcommand options*

DESCRIPTION

The *ldpTargetedPeer* command holds information about a targeted peer to be associated with an LDP interface. Targeted peers are added to a *ldpInterface* using the *ldpInterface addTargetedPeer* subcommand. The optional LDP test package must be installed in order for this command to operate.

STANDARD OPTIONS

authenticationType

The cryptographic authentication type used for the targeted peer. One of

Option	Value	Usage
ldpTargetPeerNULL	0	(default) No cryptographic authentication will be used.
ldpTargetPeerMD5	1	The Message Digest 5 (MD5) algorithm will be used for cryptographic authentication. If selected, an MD5 key must be configured. See *md5Key* below.

enable

Enables the use of this targeted peer entry. *(default = false)*

initiateTargetedHello true / false

If true, a Targeted Hello will be sent to the LDP Peer specified by the IP address in this row.

ipAddress

The IPv4 address of the targeted peer.*(default = 0.0.0.0)*

md5Key

Used with MD5 Authentication. A user-defined string; maximum = 255 characters.

COMMANDS

The **ldpTargetedPeer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpTargetedPeer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ldpTargetedPeer** command.

ldpTargetedPeer config *option value*

Modify the configuration options of the ldpTargetedPeer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ldpTargetedPeer.

ldpTargetedPeer setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4AtmLabel, ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*

NAME - ldpMulticastLeafRange

ldpMulticastLeafRange — configures a rmulticast leaf range to be associated with an LDP Router.

SYNOPSIS

ldpMulticastLeafRange *subcommand options*

DESCRIPTION

The *ldpMulticastLeafRange* command holds a multicast leaf range.

STANDARD OPTIONS

enable

Enable use of this multicast leaf range.

lspType

The type of multicast LSP. Currently only P2MP is supported. Possible values include:

- p2mp

rootAddress

The root address of the multicast LSP.

rootAddressCount

The root address count for this Multicast leaf range.

rootAddressStep

The Root Address increment step. This is applicable only if Root Address Count is greater than 1.

lspCountPerRoot

This is to specify how many different LSPs are created per Root.

labelValueStart

The start label value for first leaf.

labelValueStep

The label value increment step for more than 1 range.

COMMANDS

The *ldpMulticastLeafRange* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands.

ldpMulticastLeafRange setDefault

Sets default values for all configuration options.

ldpMulticastLeafRange addOpaqueValueElement

Adds an opaque value element.

EXAMPLES

SEE ALSO

ldpRouter

NAME - ldpOpaqueValueElement

ldpOpaqueValueElement — configures opaque value tlvs.

SYNOPSIS

ldpOpaqueValueElement *subcommand options*

DESCRIPTION

The *ldpOpaqueValueElement* command holds opaque value tlvs.

STANDARD OPTIONS

type

The type of TLV.

length

The length of the TLV.

value

The value of the TLV.

increment

The increment value.

COMMANDS

The *ldpOpaqueValueElement* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands.

ldpMulticastLeafRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpRouter, *ldpMulticastLeafRange*

NAME - ldpLearnedMulticastLabel

ldpLearnedMulticastLabel — accesses learned multicast labels.

SYNOPSIS

ldpLearnedMulticastLabel *subcommand options*

DESCRIPTION

The *ldpLearnedMulticastLabel* command holds an element of the LDP Multicast learned label list obtained in the *ldpInterface* command.

STANDARD OPTIONS

label

(read only) Indicates the label value added to the packet(s) by the upstream LDP peer.

labelSpaceId

(read only) Part of the LSR Id. It forms the last 2 octets of the 6-octet LDP Identifier.

peerIpAddress

(read only) .The RID of the upstream LDP peer. Part of the LSR Id. It must be globally unique. It forms the first 4 octets of the 6-octet LDP Identifier.

rootAddress

(read only) Root Address of IPv4 P2MP labels learned.

COMMANDS

The *ldpLearnedMulticastLabel* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpLearnedMulticastLabel cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *discoveredList* command.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - ldpLearnedOpaqueValueElement

ldpLearnedOpaqueValueElement — accesses learned opaque value element.

SYNOPSIS

ldpLearnedOpaqueValueElement *subcommand options*

DESCRIPTION

The *ldpLearnedOpaqueValueElement* command holds an element of the LDP learned Opaque Value Element obtained in the *ldpInterface* command.

STANDARD OPTIONS

value

(read only) Indicates the value of the opaque element.

length

(read only) Indicates the length of the opaque element.

type

(read only) Indicates the type of the opaque element

COMMANDS

The *ldpLearnedMulticastLabel* command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ldpLearnedOpaqueValueElement cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *discoveredList* command.

EXAMPLES

See examples under *ldpServer* and *ldpInterface*.

SEE ALSO

ldpAdvertiseFecRange, *ldpAssignedAtmLabel, ldpAtmRange*, *ldpExplicitIncludeIpFec*, *ldpInterface*, *ldpL2VplsMacRange*, *ldpL2VpnInterface*, *ldpL2VpnVcRange*, *ldpLearnedIpV4Label*, *ldpLearnedMartiniLabel*, *ldpRequestFecRange*, *ldpRouter*, *ldpServer*, *ldpTargetedPeer*

NAME - mldGroupRange

mldGroupRange — configures a multicast group range for a simulated MLD host.

SYNOPSIS

mldGroupRange *subcommand options*

DESCRIPTION

Each port's MLD implementation includes a number of hosts, which are described in *mldHost*. Each host is interested in any number of multicast groups, described in this command. For each multicast group range, a set of source addresses may be specified in *mldSourceRange*. Each MLD source range is added to the group range using the *addSourceRange* subcommand.These source ranges constitute a set of IPV6 sources that are to be included or excluded from the group range.

Refer to *MLD* for an overview.

STANDARD OPTIONS

enable true / false

Enables the use of this group range in the MLD simulation. *(default = false)*

enablePacking true | false

If true, then *recordsPerFrame* multicast addresses groups are included in each transmitted listener response message. *sourcesPerRecord* source addresses are placed in each group record. *(default = false)*

groupCount

The number of IPV6 addresses in the group range. *(default = 1)*

groupIpFrom

The starting IPV6 dress for the group range. *(default = FF02:0:0:0:0:0:0:0)*

incrementStep

The increment applied between IPV6 addresses in the range, if *groupCount* is more than 1. *(default = 1)*

recordsPerFrame

If *enablePacking* is true, then this is the number of multicast addresses groups that will be included in each transmitted listener response message. *(default = 0)*

sourceMode

This option indicates the mode applied to the associated list of source ranges. One of:

Option	Value	Usage
multicastSourceModeInclude	0	Indicate that the source range

Option	Value	Usage
		addresses are to be included.
multicastSourceModeExclude	1	*(default)* Indicate that the source range addresses are to be excluded.

sourcesPerRecord

If *enablePacking* is true, then this is the number of source addresses that will be included in each group record. *(default = 0)*

COMMANDS

The **mldGroupRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mldGroupRange addSourceRange *sourceRangeId*

Adds the source range described in the *mldSourceRange* command to the list of source ranges associated with the host. The range's entry in the list is given an identifier of *sourceRangeId*. Specific errors are:

- The parameters in *mldSourceRange* are invalid.
- A host with this *sourceRangeId* exists already in the list.

mldGroupRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *mldGroupRange* command.

mldGroupRange clearAllSourceRanges

Deletes all of the group ranges.

mldGroupRange config *option value*

Modify the configuration options of the *mldGroupRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mldGroupRange.

mldGroupRange delSourceRange *sourceRangeId*

Deletes the group range with an identifier of *sourceRangeId*. Specific errors are:

- No host with this *sourceRangeId* exists in the list.

mldGroupRange generateStreams *chasID cardID portID [action]*

This command creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	*(default)* Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for this group range; each stream covers the set of IPV6 addresses associated with the group range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv6 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the group range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the group range; it should not be reprogrammed.

mldGroupRange getFirstSourceRange

Access the first source range in the list. The results may be accessed using the *mldSourceRange* command. Specific errors are:

- There are no source ranges in the list.

mldGroupRange getNextSourceRange

Access the next source range in the list. The results may be accessed using the *mldSourceRange* command. Specific errors are:

- mldGroupRange getFirstSourceRange has not been called.
- There is no more source ranges in the list.

mldGroupRange getSourceRange *sourceRangeId*

Accesses the source range's entry in the list with an identifier of *sourceRangeId*. The source range is accessed in the *mldSourceRange* command. Specific errors are:

A source range with this *sourceRangeId* does not exist in the list.

mldGroupRange setDefault

Sets default values for all configuration options.

mldGroupRange setSourceRange *[sourceRangeId]*

Sets the values for the source range's entry in the list with an identifier of *sourceRangeId*, or if omitted the source range accessed through the use of *getFirstSourceRange/-getNextSourceRange,* based on changes made through the *mldSourceRange* command. This command can be used to change a running configuration and must be followed by an *mldServer* *write* command in order to send these changes to the protocol server. Specific errors are:

- A source range with this *sourceRangeId* does not exist in the list.

EXAMPLES

See examples under *mldServer*.

SEE ALSO

NAME - mldHost, NAME - mldServer, NAME - mldSourceRange, NAME - mldQuerierLearnedInfo, NAME - mldQuerier

NAME - mldHost

mldHost — configures a simulated MLD host.

SYNOPSIS

mldHost *subcommand options*

DESCRIPTION

Each port's MLD implementation includes a number of hosts, which are described in this command. Each host is interested in any number of multicast groups, described in *mldGroupRange*. Each MLD group range is added to the host using the *addGroupRange* subcommand. For each multicast group range, a set of source addresses may be specified in *mldSourceRange*. These source ranges constitute a set of IPV6 sources that are to be included or excluded from the group range.

Refer to *MLD* for an overview.

STANDARD OPTIONS

enable true / false

Enables the use of this host in the MLD simulation. *(default = false)*

enableGeneralQuery true | false

Enables responses to general queries received on the interface described in *protocolInterfaceDescription.(default = true)*

enableGroupSpecific true | false

Enables responses to group specific queries received on the interface described in *protocolInterfaceDescription.(default = true)*

enableImmediate Response true | false

Causes the simulated host to immediately respond to a received Query message, rather than waiting a random amount of time between 0 and the *Maximum Response Delay* field value of the Query message. *(default = false)*

enableRouterAlert true | false

Sets the router alert bit in transmitted listener reports. *(default = true)*

enableSupressReports true | false

If true, will cause the host to suppress the transmission of a listener report that duplicates one received on the interface. *(default = false)*

enableUnsolicited true | false

If true, will cause the host to transmit unsolicited listener reports at the interval specified in *reportFrequency*. *(default = false)*

protocolInterface Description

The *description* option associated with an *interfaceEntry* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

reportFrequency

If *enableUnsolicited* is set to true, then this is the frequency with which unsolicited listener reports will be sent, expressed in seconds. *(default = 120*)

version

The version of MLD to be used. *(default = 1)*

COMMANDS

The **mldHost** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mldHost addGroupRange *groupRangeId*

Adds the group range described in the *mldGroupRange* command to the list of group ranges associated with the host. The range's entry in the list is given an identifier of *groupRangeId*. This command can be used to add a group range to a running configuration and must be followed by *mldServer* *setHost* and *mldServer* *write* commands in order to send these changes to the protocol server. Specific errors are:

- The parameters in *mldGroupRange* are invalid.
- A host with this *groupRangeId* exists already in the list.

mldHost cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *mldHost* command.

mldHost clearAllGroupRanges

Deletes all of the group ranges.

mldHost config *option value*

Modify the configuration options of the *mldHost*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mldHost.

mldHost delGroupRange *groupRangeId*

Deletes the group range with an identifier of *groupRangeId*. Specific errors are:

- No host with this *groupRangeId* exists in the list.

mldHost generateStreams *chasID cardID portID [action]*

This command creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Valu	Usage
protocolServerStreamReplace	0	*(default)* Replace the port's current

Option	Valu	Usage
		streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each group range associated with the host; each stream covers the set of IPV6 addresses associated with each group range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv6 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the group range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the group range; it should not be reprogrammed.

mldHost getFirstGroupRange

Access the first group range in the list. The results may be accessed using the *mldGroupRange* command. Specific errors are:

- There are no group ranges in the list.

mldHost getNextGroupRange

Access the next group range in the list. The results may be accessed using the *mldGroupRange* command. Specific errors are:

- mldHost getFirstGroupRange has not been called.
- There is no more group ranges in the list.

mldHost getGroupRange *groupRangeId*

Accesses the group range's entry in the list with an identifier of *groupRangeId*. The group range is accessed in the *mldGroupRange* command. Specific errors are:

- A group range with this *groupRangeId* does not exist in the list.

mldHost setDefault

Sets default values for all configuration options.

mldHost setGroupRange *[groupRangeId]*

Sets the values for the group range's entry in the list with an identifier of *groupRangeId*, or if omitted the group range accessed through the use of *getFirstGroupRange/getNextGroupRange,* based on changes made through the *mldGroupRange* command. This command can be used to change a running configuration and must be followed by an *mldServer* *write* command in order to send these changes to the protocol server. (A call to *mldServer* *setHost*should not be used). Specific errors are:

- A group range with this *groupRangeId* does not exist in the list.
- Too many groups defined.

EXAMPLES

See examples under *mldServer*.

SEE ALSO

NAME - mldGroupRange, NAME - mldServer, NAME - mldSourceRange, NAME - mldQuerierLearnedInfo, NAME - mldQuerier

NAME - mldQuerierLearnedInfo

mldQuerierLearnedInfo — views retrieved Learned MLD information.

SYNOPSIS

mldQuerierLearnedInfo *subcommand options*

DESCRIPTION

The *mldQuerierLearnedInfo* command is used to look at the information retrieved when using the *requestLearnedInfo* and *getLearnedInfoList* subcommands of the *mldQuerier* command.

Refer to *mldQuerierLearnedInfo* for an overview of this command. The optional MLD test package must be installed in order for this command to operate.

STANDARD OPTIONS

compatibilityMode

(Read-only.) What version of MLD this group address is. One of:

Option	Value	Usage
MLDV1		Uses MLD version 1.
MLDV2		Uses MLD version 2.

compatibilityTimer

(Read-only.) The number of seconds remaining in the compatibility timer. (Integer)

filterMode

(Read-only.) Whether this group address is included or excluded. One of:

Option	Value	Usage
MLD_GROUPMODE_ INCLUDE	0	
MLD_GROUPMODE_ EXCLUDE	1	

groupAddress

(Read-only.) The IPv4 address for the router group. (IPv4 address)

groupTimer

(Read-only.) The number of seconds remaining in the group address timer. (Integer)

sourceAddress

(Read-only.) The IPv4 address for the group source. (IPv4 address)

sourceTimer

(Read-only.) The number of seconds remaining in the group address timer. (Integer)

COMMANDS

The *mldQuerierLearnedInfo* command is invoked with the following subcommand. If no subcommand is specified, returns a list of all subcommands available.

mldQuerierLearnedInfo setDefault

Sets the options to default values.

EXAMPLES

See examples under *mldServer*.

SEE ALSO

NAME - mldGroupRange, NAME - mldHost, NAME - mldServer, NAME - mldSourceRange, NAME - mldQuerier

NAME - mldQuerier

mldQuerier — configures an MLD Querier.

SYNOPSIS

mldQuerier *subcommand options*

DESCRIPTION

Each port's MLD implementation includes a number of Queriers, which are included in *mldServer*.

Refer to *MLD* for an overview of MLD.

STANDARD OPTIONS

discardLearnedInfo true / false

When disabled, the emulated Querier maintains a complete record state for received reports and sent queries (based on the timer expiry for received groups and sources. (Default = disabled)

When enabled, the Querier does not maintain any database and only sends periodic General Queries. The Specific Query group/source record information is not calculated based on any earlier received report, but solely based on the last received report.

enable true / false

If set to True, enables this MLD Querier.

enableRouterAlert true / false

If enabled, sets the "Send Router Alert" bit in the IP header.

generalQueryInterval

The amount of time (in seconds) between MLD General Query messages sent by the querier. (Integer) (Default = 125)

genQueryResponseInterval

The maximum amount of time (in seconds) that the MLD querier waits to receive a response to a General Query message. (Integer) (Default = 10 seconds, and must be less than the Query Interval)

robustnessVariable

Defines the subnet vulnerability to lost packets. MLD can recover from robustness variable minus 1 lost packets. The robustness variable should be set to a value of 2 or greater. (Integer) (Default = 2)

specQueryResponseInterval

The maximum amount of time (in seconds) that the MLD querier waits to receive a response to a Specific Query message. (Integer) (Default = 10 seconds, and must be less than the Query Interval).

specQueryTransmissionCount

Indicates the total number of Specific Query messages sent every Specific Query Response Interval (in seconds) before assuming that there is no interested listener for the particular group/source. (Integer)

startupQueryCount

The number of MLD General Query messages sent at startup. (Integer) (Default = 2)

supportElection true / false

Indicates whether the Querier participates in Querier election or not. If disabled, then all incoming Query messages are discarded.

supportOlderVersionHost true / false

Indicates whether the Querier will comply with RFC 3376 Section 7.3.2 and RFC 3810 Section 8.3.2. If disabled, all membership reports with version less than the current version are discarded.

supportOlderVersionQuerier

Indicates whether the Querier downgrades to the lowest version of received Query messages. If disabled, all Query messages with version less than the current version are discarded.

version

Indicates the MLD protocol version to be used. One of:

Option	Value	Usage
mldQuerierVersion1	1	Uses MLD Version 1
mldQuerierVersion2	2	Uses MLD Version 2

Learned Info

isQuerier true / false

(Read-only) If true, indicates that the currently-elected querier is self. If false, indicates that the currently-elected querier is other.

querierAddress

(Read-only) Indicates the IPv6 address of the currently-elected querier. (String)

querierWorkingVersion

(Read-only) Indicates the working version of the MLD querier at that point in time. (Integer)

COMMANDS

The **mldQuerier** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mldQuerier getFirstLearnedInfo

Retrieves the first entry of MLD learned info from the list.

mldQuerier getLearnedInfoList

Populates the Learned info list for the MLD Querier. When it returns TCL_OK, it means that learned info is returned.

mldQuerier getNextLearnedInfo

Retrieves the next entry of MLD learned info from the list.

mldQuerier requestLearnedInfo

Requests the learned MLD information for the respective MLD Querier.

mldQuerier setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *mldServer*.

SEE ALSO

NAME - mldGroupRange, NAME - mldHost, NAME - mldServer, NAME - mldSourceRange, NAME - mldQuerierLearnedInfo

NAME - mldServer

mldServer — accesses the MLD component of the protocol server for a particular port.

SYNOPSIS

mldServer *subcommand options*

DESCRIPTION

The *mldServer* command is necessary in order to access the MLD protocol server for a particular port. The *select* subcommand **must** be used before all other MLD commands. The MLD simulation covers both MLDv1 and MLDv2.

Each port's MLD implementation includes a number of hosts, which are described in *mldHost*. A host is added to the server with the *addHost* subcommand. Each host is interested in any number of multicast groups, described in *mldGroupRange*. For each multicast group range, a set of source addresses may be specified in *mldSourceRange*. These source ranges constitute a set of IPV6 sources that are to be included or excluded from the group range.

Refer to *MLD* for an overview.

STANDARD OPTIONS

enableSendDoneOnStop true / false

If true, enables the Send Done's on Stop feature.

mldv2ReportType

The type of MLD Multicast Listener Report to generate. (Integer)

numGroups

The number of multicast groups to transmit every *timePeriod* milliseconds. A value of 0 disables this feature and transmits all groups immediately for all updates. *(default = 0)*

timePeriod

The time period to use for throttling updates, expressed in milliseconds. A value of 0 disables this feature and transmits all groups immediately for all updates. *(default = 0)*

COMMANDS

The **mldServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mldServer addHost *hostId*

Adds the MLD host described in the *mldHost* command to the list of hosts associated with the port. The host's entry in the list is given an identifier of *hostId*. Specific errors are:

- mldServer select has not been called.
- The host parameters in *mldHost* are invalid.

- A host with this *hostId* exists already in the list.
- Too many groups defined.
- Too many hosts defined.

mldServer addQuerier *routerId*

Adds the MLD querier described in the *mldQuerier*command to the list of queriers associated with the port. The querier's entry in the list is given an identifier of *routerId*.

mldServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **mldServer** command.

mldServer clearAllHosts

Deletes all the MLD hosts in the list. Specific errors are:

- mldServer select has not been called.

mldServer clearAllQueriers

Deletes all the MLD queriers in the list.

mldServer config *option value*

Modify the configuration options of the *mldServer.* If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mldServer.

mldServer delHost *hostId*

Deletes the MLD host described that has an identifier of *hostId*. Specific errors are:

- mldServer select has not been called.
- There is no host with this *hostId* in the list.

mldServer delQuerier *routerId*

Deletes the MLD querier described that has an identifier of *routerId*.

mldServer generateStreams *chasID cardID portID [action]*

This command creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	*(default)* Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each simulated host and included group range; each stream covers the set of IPV6 addresses associated with the group range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv6 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the group range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the group range; it should not be reprogrammed.

mldServer get

Gets the current MLD server configuration for the last port selected with the *select* subcommand. Call this command before calling the *cget* subcommand to get the value of the configuration option.

mldServer getFirstHost

Access the first MLD host in the list. The results may be accessed using the *mldHost* command. Specific errors are:

- mldServer select has not been called.
- There are no hosts in the list.

mldServer getFirstQuerier

Access the first MLD querier in the list. The results may be accessed using the *mldQuerier* command.

mldServer getNextHost

Access the next MLD host in the list. The results may be accessed using the *mldHost* command. Specific errors are:

- mldServer select has not been called.
- mldServer getFirstHost has not been called.
- There are no more hosts in the list.

mldServer getNextQuerier

Access the next MLD querier in the list. The results may be accessed using the *mldQuerier* command.

mldServer getHost *hostId*

Access the MLD host with an identifier of *hostId.* The results may be accessed using the *mldHost* command. Specific errors are:

- mldServer select has not been called.
- There is no host with this *hostId* in the list.

mldServer getQuerier *routerId*

Access the MLD querier with an identifier of *routerId.* The results may be accessed using the *mldQuerier* command.

mldServer select *chasID cardID portID*

Accesses the MLD component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The MLD protocol package has not been installed.
- Invalid port specified.
- mld is not supported on this older port type.

mldServer set

Sets the configuration of the MLD server in IxHAL for the port last selected with the *select* subcommand by reading the configuration option values set by the *configsubcommand*. Specific errors are:

- No connection to a chassis.
- The port is being used by another user.
- Configured parameters are not valid for this setting.

mldServer setDefault

Sets default values for all configuration options.

mldServer setHost *hostId*

Sets the values for the host's entry in the list with an identifier of *hostId* based on changes made through the *mldHost* command. This command should be used to change a running configuration and must be followed by an *mldServer write* command in order to send these changes to the protocol server. Specific errors are:

- A host with this *hostId* does not exist in the list.
- Too many groups defined.
- Too many hosts defined.

mldServer setQuerier *routerId*

Sets the values for the querier's entry in the list with an identifier of *routerId* based on changes made through the *mldQuerier* command. This command should be used to change a running configuration and must be followed by an *mldServer write* command in order to send these changes to the protocol server.

mldServer write

Sends any changes made with the MLD suite of commands to the protocol server for immediate application. This command **must** be used in order for the changes to have an effect.

EXAMPLES

```
package req IxTclHal
set localhost astro
set username user
# Check if we're running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set ch [ixGetChassisID $host]
set card 12
set port 1
set portList [list [list $ch $card $port]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
# Example for configure MLD protocol and get back the
configuration using IDs.
```

```
# In this method users add each object by an id and they get/set by
the same id
# in the reverse order of adding.
# These ids don’t apear in the GUI and if users close the wish
consol they can NOT
# get back the configuration by those Ids again.
# Make sure be consistent in using Ids. If you are using Ids, use
it all the way
# and don’t mix it with getFirst/getNext
# Configure a version2 MLD Host with one group range and one
source range.
port setFactoryDefaults $ch $card $port
mldServer select $ch $card $port
mldServer clearAllHosts
#configure source range
mldSourceRange config -sourceIpFrom 100::2
mldSourceRange config -count 1
# Add the source range to the group range
if [mldGroupRange addSourceRange source1] {
logMsg “Error in adding sourceRange”
}
# Configure groupRange
mldGroupRange config -enable true
mldGroupRange config -groupIpFrom 12::100
# Add the group range to the host
if [mldHost addGroupRange group1] {
logMsg “Error adding groupRange group1”
}
mldGroupRange config -enable true
mldGroupRange config -groupIpFrom FF00::100
# And another one
if [mldHost addGroupRange group2] {
logMsg “Error adding groupRange group2”
}
```

```
# Configure host - assume interface exists
mldHost config -enable true
mldHost config -protocolInterfaceDescription “$card:0$port”
mldHost config -version mldVersion2
# Add the host to the server
if [mldServer addHost host1] {
logMsg “Error adding host”
}
# Send to the hardware
mldServer set
if [mldServer write] {
logMsg “Error writing”
}
# To get an object:
# Make sure you apply the hierarchy to get objects
# Be consistent in using Ids. If you are using Ids, use it all the
way
# and don’t mix it with getFirst/getNext methods.
# Example of disabling host on the fly (when MLD server is
running)
mldServer select $ch $card $port
mldServer getHost host1
mldHost config -enable
0
mldServer setHost host1
mldServer write
# Example of modifying group Range on the fly
mldServer select $ch $card $port
mldServer getHost host1
mldHost getGroupRange group2
mldGroupRange config -groupIpFrom FFC0::200
if [mldHost setGroupRange group2 ] {
logMsg “Error in setting group range group2”
}
```

```
mldServer write
# Example of modifying source Range on fly
mldServer select $ch $card $port
mldServer getHost host1
mldHost getGroupRange group2
mldGroupRange getSourceRange source1
mldSourceRange config -count 20
if [mldGroupRange setSourceRange source1] {
logMsg “Error in setting source range”
}
mldServer write
# Example of generating streams at server level for enabled hosts
and group ranges.
set targetCh 1
set targetCard 12
set targetPort 2
set targetPortList [list [list $targetCh $targetCard $targetPort]]
mldServer select $ch $card $port
mldServer generateStreams $targetCh $targetCard $targetPort
ixWriteConfigToHardware targetPortList
# Example of generating streams at group range level for enabled
group range.
# You can get the group range by name too. Here is an example of
using getFirst/getNext.
mldServer select $ch $card $port
mldServer getFirstHost
mldHost getFirstGroupRange
mldGroupRange generateStreams $targetCh $targetCard $targetPort
ixWriteConfigToHardware targetPortList
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
```

```
if [isUNIX] {

ixDisconnectTclServer $host

}
```

SEE ALSO

NAME - mldGroupRange, NAME - mldHost, NAME - mldSourceRange, NAME - mldQuerier, NAME - mldQuerierLearnedInfo

NAME - mldSourceRange

mldSourceRange — configures a multicast source range for an MLD group range.

SYNOPSIS

mldSourceRange *subcommand options*

DESCRIPTION

Each port's MLD implementation includes a number of hosts, which are described in *mldHost*. Each host is interested in any number of multicast groups, described in *mldGroupRange*. For each multicast group range, a set of source addresses may be specified in this command.These source ranges constitute a set of IPV6 sources that are to be included or excluded from the group range.

Refer to *MLD* for an overview.

STANDARD OPTIONS

count

The number of IPV6 addresses in the source range. *(default =1)*

sourceIpFrom

The starting IPV6 dress for the source range. *(default = 0:0:0:0:0:0:0:0)*

COMMANDS

The **mldSourceRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mldSourceRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *mldSourceRange* command.

mldSourceRange config *option value*

Modify the configuration options of the *mldSourceRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mldSourceRange.

mldSourceRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *mldServer*.

SEE ALSO

NAME - mldGroupRange, NAME - mldHost, NAME - mldServer

NAME - ospfInterface

ospfInterface — configures an interface for an OSPF router.

SYNOPSIS

ospfInterface *subcommand options*

DESCRIPTION

The *ospfInterface* command holds the information related to a single interface on the simulated router. Interfaces are added into the *ospfRouter* interface list using the *ospfRouter addInterface* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfInterface* for an overview.

Authentication is handled by the *authenticationMethod, password, mk5KeyId* and *md5key* options.

Note that this command only applies to the OSPFv2 implementation; the commands related to OSPFv3 all begin with **ospfV3...**.

STANDARD OPTIONS

adminGroup

A 4-octet bit mask used to assign administrative group numbers to the interface, for use in assigning "colors" and resource classes. Each set bit corresponds to a single administrative group for this interface. The settings translate into Group numbers which range from 0 to 31 (integers). *(default = "00 00 00 00")*

areaId

The OSPF area ID associated with the interface. *(default = 0)*

authenticationMethod

The type of authentication to be used for the interface. One of the following options:

Option	Value	Usage
ospfInterfaceAuthenticationNull	0	*(default)* No authentication.
ospfInterfaceAuthenticationPassword	1	Clear text 64-bit password.
ospfInterfaceAuthenticationMD5	2	MD5 key authentication.

connectToDut

Indicates that this interface is directly connected to the DUT. *(default = 0)*

deadInterval

The time after which the DUT router is considered dead if it does not send HELLO messages. *(default = 40)*

enable true | false

Enables the use of the simulated interface. *(default = false)*

enableAdvertise NetworkRange true | false

Enables the advertisement of a range of OSPF routers expressed as a matrix of n x m routers. A network range is defined in the *ospfNetworkRange* command. This may only be used when *connectToDut* is *false. (default = false)*

enableBfdRegistration true | false

Indicates if a BFD session is to be created to the OSPF peer IP address once the OSPF session is established. This allows OSPF to use BFD to maintain IPv4 connectivity the OSPF peer.

enableTraffic Engineering true / false

Enables the use of the *linkMetric, maxBandwidth, maxReservableBandwidth,* and *unreservedBandwidthPriority0-7* for traffic engineering purposes. These values are used to generate two LSAs: a router LSA and a link LSA with an opaque TLV containing sub-TLV for the link metric, max bandwidth, max reservable bandwidth and unreserved bandwidth priorities. *(default = false)*

enableValidateMtu true / false

Enables validation on incoming database entries received by the simulated router. If this is set to 1, then received database entries which advertise an MTU larger than the value in the *mtuSize* option are ignored. If this is set to 0, then the advertised MTU size is 0 and the MTU size in received database entries is ignored. *(default = true)*

helloInterval

The time between HELLO messages sent over the interface. *(default = 10)*

ipAddress

The IP address for this interface. Only used if *protocolInterfaceDescription* is empty. *(default = 0.0.0.0)*

ipMask

The IP mask associated with the IP address for this interface. Only used if *protocolInterfaceDescription* is empty. *(default = 255.255.255.0)*

linkMetric

If *enableTrafficEngineering* is *true*, then this indicates the traffic engineering metric associated with the interface. *(default = 0)*

linkType

The Link Type advertised in the Router LSA interface list. One of the following:

Option	Value	Usage
ospfInterfaceLinkPointToPoint	1	A point-to-point network.
ospfInterfaceLinkTransit	2	*(default)* A transit network.
ospfInterfaceLinkStub	3	A stub network.

maxBandwidth

If *enableTrafficEngineering* is 1, then this indicates the maximum bandwidth that can be used on the link between this interface and its neighbors in the outbound direction. *(default = 0.0)*

maxReservable Bandwidth

If *enableTrafficEngineering* is 1, then this indicates the maximum bandwidth, in bytes per second, that can be reserved on the link between this interface and its neighbors in the outbound direction. *(default = 0.0)*

md5Key

If *authenticationMethod* is set to *ospfInterfaceAuthenticationMD5*, then this is secret MD5 key used for authentication. *(default = "")*

md5KeyId

If *authenticationMethod* is set to *ospfInterfaceAuthenticationMD5*, then this is MD5 key ID used for authentication. *(default = 1)*

metric

The metric associated with the interface. *(default = 10)*

mtuSize

The advertised MTU value in database entries sent to other routers. The *enableValidateMTU* option must be set to *true* in order for the MTU size to be transmitted. *(default = 1500)*

neighborRouterId

When the *linkType* option is set to *ospfInterfaceLinkPointToPoint,* then this option should be set to the ID of the router on the other end of the point-to-point connection. *(default = 0.0.0.0)*

networkType

Indicates the type of network for the interface. One of the following options:

Option	Value	Usage
ospfPointToPoint	1	Indicates that the network is point to point, as in a PPP connection.
ospfBroadcast	2	*(default)* Indicates that the network is a broadcast network, as in an Ethernet connection.
ospfPointToMultipoint	3	Indicates that the network is point to multipoint.

numberOfLearnedLsas

(Read-only.) The number of learned LSAs obtained through a call to *ospfInterface getLearnedLsaList.*

options

Options related to the interface. Multiple options may be or'd together. The available options are:

Option	Value	Usage
ospfOptionBitTypeOfService	0x01	
ospfOptionBitExternalRouting	0x02	default
ospfOptionBitMulticast	0x04	
ospfOptionBitNSSACapability	0x08	
ospfOptionBitExternalAttributes	0x10	
ospfOptionBitDemandCircuit	0x20	
ospfOptionBitLSANoForward	0x40	
ospfOptionBitUnused	0x80	

password

If *authenticationMethod* is set to *ospfInterfaceAuthenticationPassword*, then this is the 64-bit plaintext password used for authentication. *(default = {00 00 00 00 00 00 00 00}*

priority

The priority of the interface, for use in election of the designated or backup master. *(default = 0)*

protocolInterface Description

The *description* option associated with an *interfaceEntry* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

unreservedBandwidth Priority0-7

If *enableTrafficEngineering* is true, then these eight values indicate the amount of bandwidth, in bytes per second, not yet reserved at each of the eight priority levels. These values correspond to the bandwidth that can be reserved with a setup priority of 0 through 7. Each value must be less than the *maxReservableBandwidth* option. *(default = 0.0)*

COMMANDS

The **ospfInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfInterface** command.

ospfInterface config *option value*

Modify the configuration options of the ospfInterface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfInterface.

ospfInterface getFirstLearnedLsa

This command must be preceded by use of the *ospfInterface getLearnedLsaList* command and followed by multiple uses of *ospfInterface getNextLearnedLsa.* This command fetches the first of the learned LSAs in the list into memory. The data associated with the individual LSA may be read through the use of the *ospfUserLsa* command.

ospfInterface getLearnedLsaList

This command must be preceded by use of the *ospfInterface requestLearnedLsaList* and followed by a call to *ospfInterface getFirstLearnedLsa.* This command determines whether the reading of learned LSAs from the protocol server has completed. This command should be called until it returns a `0', or until some suitable period of time has elapsed. The number of learned LSAs is available in the *numberOfLearnedLsas* option.

ospfInterface getNextLearnedLsa

This command must be preceded by use of the *ospfInterface getFirstLsa* command and repeated multiple times to obtain all of the learned LSAs*.* This command fetches the next of the learned LSAs in the list into memory. The data associated with the individual LSA may be read through the use of the *ospfUserLsa* command.

ospfInterface requestLearnedLsa

Requests that the learned LSAs associated with this interface be retrieved from the protocol server. This command must be followed by call to *ospfInterface getLearnedLsaList.*

ospfInterface setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfServer*

SEE ALSO

ospfServer, *ospfRouter*, *ospfRouteRange*, *ospfUserLsaGroup*, *ospfUserLsa*, *ospfRouterLsaInterface*, *ospfNetworkRange*

NAME - ospfNetworkRange

ospfNetworkRange — configures an OSPF network range.

SYNOPSIS

ospfNetworkRange *subcommand options*

DESCRIPTION

The *ospfNetworkRange* command allows a matrix of simulated routers to be defined in the form of a number of columns and rows. Each router is connected to its immediate row and column neighbors. The entry point to the matrix is defined as a row and column location. There are provisions for varying the router ID and IP address of the simulated router.

Network ranges defined in this command are added to an *ospfInterface* using the *ospfRouter addInterface* command, given that the *ospfInterface* command is configured as **not** connected to the DUT and *enableAdvertiseNetworkRange* is set to true. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment.

STANDARD OPTIONS

enableAdvertise RouterLsaLoopback true | false

If true, advertises the router's LSA loopback address. *(default = false)*

enableBBit true | false

If true, advertises the range as a border router. *(default = false)*

enableEBit true | false

If true, advertises the range as an edge router. *(default = false)*

enableIncrementIp Mask true | false

If *true,* the IP address assigned to the simulated routers is incremented based on the value of the *maskWidth* parameter. For example, if *maskWidth = 24,* then each new router will be incremented by a value of 0.0.1.0 as they are assigned addresses.

If *false*, then IP addresses are incremented by the value in the *subnetIpIncrementBy* option. *(default = true)*

enableTe true | false

If true, advertises the Traffic Engineering data for the simulated grid, as detailed in the *maxBandwidth, maxReservableBandwidth, linkMetric* and *unreservedBandwidthPriority0-7* options. *(default = false)*

entryPointColumn

The column number for the entry point into the matrix of simulated routers. Column numbers start at 1. *(default = 1)*

entryPointRow

The row number for the entry point into the matrix of simulated routers. Row numbers start at 1. *(default = 1)*

firstRouterId

The ID associated with the first router. *(default = 0.0.0.0)*

firstSubnetIpAddress

The IP subnet address associated with the first router. *(default = 0.0.0.0)*

linkMetric

If *enableTe* is *true*, then this indicates the traffic engineering metric associated with the link to the grid. *(default = 0)*

linkType

The type of links used between the elements. One of:

Option	Value	Usage
ospfNetworkRangeLinkBroadcast	0	*(default)* Broadcast network
ospfNetworkRangeLinkPointToPoint	1	Point to point network

maskWidth

The length of the mask associated with the *firstSubnetIpAddress.(default = 24)*

maxBandwidth

If *enableTe* is *true*, then this indicates the maximum bandwidth that can be used on the link between this interface and the simulated grid at its entry point. *(default = 0.0)*

maxReservable Bandwidth

If *enableTe* is *true*, then this indicates the maximum bandwidth, in bytes per second, that can be reserved on the link between this interface and the simulated grid at its entry point. *(default = 0.0)*

numColumns

The number of columns in the generated matrix of routers. *(default = 1)*

numGeneratedLsas

(Read-only.) The number of LSAs that were generated to cover the simulated network range.

numRouters

(Read-only.) The total number of routers in the matrix.

numRows

The number of rows in the generated matrix of routers. *(default = 1)*

numSubnets

(Read-only.) The total number of subnets in the matrix.

routerIdIncrementBy

The value that the router ID of simulated routers will be incremented as new routers are generated. *(default = 0.0.0.1)*

subnetIpIncrementBy

If *enableIncrementIpFromMask* is *false,* this is the value used to increment the subnet address by between successively generated routers. *(default = 0.0.1.0)*

unreservedBandwidth Priority0-7

If *enableTe* is *true*, then these eight values indicate the amount of bandwidth, in bytes per second, not yet reserved at each of the eight priority levels. These values correspond to the bandwidth that can be reserved with a setup priority of 0 through 7. Each value must be less than the *maxReservableBandwidth* option. *(default = 0.0)*

COMMANDS

The **ospfNetworkRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfNetworkRange config *option value*

Modify the configuration options of the ospfNetworkRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfNetworkRange.

EXAMPLES

See examples under *ospfServer*.

ospfServer, *ospfRouter*, *ospfRouteRange*, *ospfUserLsaGroup*, *ospfUserLsa*, *ospfRouterLsaInterface*, *ospfNetworkRange*

NAME - ospfRouter

ospfRouter — configures an OSPF router.

SYNOPSIS

ospfRouter *subcommand options*

DESCRIPTION

The *ospfRouter* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *ospfRouteRange* command.
- Interfaces — router interface, constructed in the *ospfInterface* command.
- LSA Groups — Link State Advertising Groups which will be associated with advertised routes, constructed in the *ospfUserLSAGroup* command and subsidiary commands.

Routers defined in this command are added to an *ospfServer* using the *ospfServer addRouter* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to ospfRouter for an overview of this command.

Note that this command only applies to the OSPFv2 implementation; the commands related to OSPFv3 all begin with **ospfV3....**

STANDARD OPTIONS

autoGenerateRouterLsa true / false

If enabled, the router will automatically generate a router LSA including all of the interfaces added with the *ospfRouter addInterface* command. This should be turned off if you are building OSPF topologies with *ospfUserLsa* commands. *(default = true)*

enable true / false

Enables the use of this router in the simulated OSPF network. *(default = false)*

enableDiscard LearnedLsas true / false

When this option is true, this simulated OSPF router (RID) will not learn any LSAs from the neighbor. *(default = false)*

enableGracefulRestart true | false

Enables the graceful restart Helper Mode function, per the IETF drafts, for the emulated OSPF router. *(default = false)*

enableReasonRedundantProcessor true | false

(Available ONLY for use with OSPF Graceful Restart RFC 3623 support)

Enables Graceful Restart Helper Mode for the emulated OSPF router when the restart reason is an unplanned switchover to a redundant control processor on the restarting router (an unplanned outage). *(default = true)*

enableReasonSoftReloadUpgrade true | false

(Available ONLY for use with OSPF Graceful Restart RFC 3623 support)

Enables Graceful Restart Helper Mode for the emulated OSPF router when the restart reason is a software reload or upgrade on the restarting router (a planned outage). *(default = true)*

enableReasonSoftRestart true | false

(Available ONLY for use with OSPF Graceful Restart RFC 3623 support)

Enables Graceful Restart Helper Mode for the emulated OSPF router when the restart reason is a software restart on the restarting router (for a planned or unplanned outage). *(default = true)*

enableReasonUnknown true | false

(Available ONLY for use with OSPF Graceful Restart RFC 3623 support)

Enables Graceful Restart Helper Mode for the emulated OSPF router when the restart reason is unknown and unplanned (an unplanned outage). *(default = true)*

enableRebuild Adjacency true | false

The *enableGracefulRestart* option must be *true.* If this option is *true*, Database Description (DBD) packets will have the "R" bit set and the DBD packets will also have the "LR" (LSDB Resynchronization) bit set in the LLS Extended Options TLV. Out-of-Band Link State Database (OOB LSDB) resynchronization will be used instead of normal LSDB resynchronization, in order to preserve the OSPF adjacency with the neighbor router across OSPF Graceful Restart. *(default = false)*

enableStrictLSAChecking true | false

(Available ONLY for use with OSPF Graceful Restart RFC 3623 support)

If enabled, the OSPF Restart Helper will terminate Graceful Restart when there are changes to an LSA that would be flooded to, or retransmitted by, the restarting router. *(default = true)*

enableSupportForRFC3623 true | false

(Available ONLY for use with OSPF Graceful Restart RFC 3623 support)

Enables Graceful Restart Helper Mode per RFC 3623 on the emulated OSPF router. *(default = false)*

routerId

ID of the router, usually the lowest IP address on the router. *(default = 0.0.0.0)*

COMMANDS

The **ospfRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfRouter addInterface *interfaceLocalId*

Adds the router interface described in the *ospfInterface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *interfaceLocalId*. Specific errors are:

- The parameters in *ospfInterface* are invalid.
- A router with this *interfaceLocalId* exists already in the list.

ospfRouter addRouteRange *routeRangeLocalId*

Adds the route range described in the *ospfRouteRange* command to the list of route ranges associated with the router. The range's entry in the list is given an identifier of *routeRangeLocalId*. Specific errors are:

- The parameters in *ospfRouteRange* are invalid.
- A router with this *routeRangeLocalId* exists already in the list.

ospfRouter addUserLsaGroup *userLsaGroupLocalId*

Adds the user LSA group described in the *ospfUserLsaGroup* command to the list of user LSAs associated with the router. The LSA's entry in the list is given an identifier of *userLsaGroupLocalId*. Specific errors are:

- The parameters in *ospfUserLsaGroup* are invalid.
- A router with this *UserLsaGroupLocalId* exists already in the list

ospfRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfRouter** command.

ospfRouter clearAllInterfaces

Deletes all of the router interfaces.

ospfRouter clearAllLsaGroups

Deletes all of the user LSA groups.

ospfRouter clearAllRouteRanges

Deletes all of the route ranges.

ospfRouter config *option value*

Modify the configuration options of the ospfRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfRouter.

ospfRouter delInterface *interfaceLocalId*

Deletes the router interface with an identifier of *interfaceLocalId*. Specific errors are:

- No router with this *interfaceLocalId* exists in the list.

ospfRouter delRouteRange *routeRangeLocalId*

Deletes the route range with an identifier of *routeRangeLocalId*. Specific errors are:

- No router with this *routeRangeLocalId* exists in the list.

ospfRouter delUserLsaGroup *userLsaGroupLocalId*

Deletes the user LSA group with an identifier of *userLsaGroupLocalId*. Specific errors are:

- No router with this *UserLsaGroupLocalId* exists in the list.

ospfRouter **getFirstInterface**

Access the first interfacein the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- There are no interfaces in the list.

ospfRouter getFirstRouteRange

Access the first route range in the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- There are no route ranges in the list.

ospfRouter getFirstUserLsaGroup

Access the first user LSA group in the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- There are no user LSA groups in the list.

ospfRouter getInterface *interfaceLocalId*

Accesses the interface's entry in the list with an identifier of *interfaceLocalId*. The router interface is accessed in the *ospfInterface* command. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

ospfRouter getNextInterface

Access the next interface in the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- ospfRouter getFirstInterface has not been called.
- There is no more interfaces in the list.

ospfRouter getNextRouteRange

Access the next route range in the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- ospfRouter getFirstRouteRange has not been called.
- There is no more route ranges in the list.

ospfRouter getNextUserLsaGroup

Access the next user LSA group in the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- ospfRouter getFirstUserLsaGroup has not been called.
- There is no more user LSA groups in the list.

ospfRouter getRouteRange *routeRangeLocalId*

Accesses the range's entry in the list with an identifier of *routeRangeLocalId*. The router range is accessed in the *ospfRouteRange* command. Specific errors are:

- A route range with this *routeRangeLocalId* does not exist in the list.

ospfRouter getUserLsaGroup *userLsaGroupLocalId*

Accesses the user LSA group's entry in the list with an identifier of *userLsaGroupLocalId*. The group is accessed in the *ospfUserLsaGroup* command. Specific errors are:

- A router with this *userLsaGroupLocalId* does not exist in the list.

ospfRouter setDefault

Sets default values for all configuration options.

ospfRouter setInterface *interfaceLocalId*

Sets the values for the interface's entry in the list with an identifier of *interfaceLocalId* based on changes made through the *ospfInterface* command. This command can be used to change a running configuration and must be followed by an *ospfServer write* command in order to send these changes to the protocol server. Specific errors are:

- An interface with this *interfaceLocalId* does not exist in the list.

ospfRouter setRouteRange *routeRangeLocalId*

Sets the values for the route range's entry in the list with an identifier of *routeRangeLocalId* based on changes made through the *ospfRouteRange* command. This command can be used to change a running configuration and must be followed by an *ospfServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router range with this *routeRangeLocalId* does not exist in the list.

ospfRouter setUserLsaGroup *userLsaGroupLocalId*

Sets the values for the user LSA group's entry in the list with an identifier of *userLsaGroupLocalId* based on changes made through the *ospfUserLsaGroup* command. This command can be used to change a running configuration and must be followed by an *ospfServer write* command in order to send these changes to the protocol server. Specific errors are:

- A user LSA group with this *userLsaGroupLocalId* does not exist in the list.

EXAMPLES

See examples under *ospfServer*.

ospfServer, *ospfInterface*, *ospfRouteRange*, *ospfUserLsaGroup*, *ospfUserLsa*, *ospfRouterLsaInterface*, *ospfNetworkRange*

NAME - ospfRouteRange

ospfRouteRange — sets up the parameters associated with an OSPF route range.

SYNOPSIS

ospfRouteRange *subcommand options*

DESCRIPTION

The *ospfRouteRange* command describes an individual set of routes. Route ranges are added into *ospfRouter* lists using the *ospfRouter addRouteRange* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfRouteRange* for an overview. Note that this command only applies to the OSPFv2 implementation; the commands related to OSPFv3 all begin with **ospfV3...**.

STANDARD OPTIONS

enable true / false

Enables the use of this route range for the simulated router. *(default = false)*

metric

The cost metric associated with the route. *(default = 0)*

networkIpAddress

The IP address of the routes to be advertised. *(default = 0.0.0.0)*

numberOfNetworks

The number of prefixes to be advertised. *(default = 1)*

prefix

The number of bits in the prefixes to be advertised. For example, a value of 24 is equivalent to a network mask of 255.255.255.0. *(default = 24)*

routeOrigin

Whether the route originated within the area or externally. One of:

Option	Value	Usage
ospfRouteOriginArea	0	*(default)* within the area
ospfRouteOriginExternal	1	from outside the area
ospfRouteOriginExternalType2	2	from outside the area, but with metrics which are larger than any internal metric
ospfRouteOriginnssa	3	route originated in an Not So Subby Area (NSSA)
ospfRouteOriginsameArea	4	route originated in the same area

COMMANDS

The **ospfRouteRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfRouteRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfRouteRange** command.

ospfRouteRange config *option value*

Modify the configuration options of the ospfRouteRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfRouteRange.

ospfRouteRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under ospfServer.

SEE ALSO

ospfServer, *ospfInterface*, *ospfRouter*, *ospfUserLsaGroup*, *ospfUserLsa*, *ospfRouterLsaInterface*

NAME - ospfRouterLsaInterface

ospfRouterLsaInterface — configures a single Router LSA Interface entry.

SYNOPSIS

ospfRouterLsaInterface *subcommand options*

DESCRIPTION

The *ospfRouterLSAInterface* command describes a single Router LSA Interface entry. The data from this entry is added to an *ospfUserLsa* list for a RouterLSA entry using the *ospfUserLsa addInterfaceDescriptionToRouterLsaIdentifier* or *addInterfaceDescriptionToRouterLsa* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfRouterLSAInterface* for an overview of this command.

STANDARD OPTIONS

linkData

Depends on the *linkType* field: *(default = 0.0.0.0)*

linkType	Value
ospfLinkPointToPoint	For unnumbered connections, the interface's MIB-II ifIndex value.
ospfLinkTransit	The router interface's IP address.
ospfLinkStub	The network's IP address mask.
ospfLink	The router interface's IP address.

linkId

Identifies the object that this router link connects to, depending on the *linkType* field: *(default = 0.0.0.0)*

linkType	Value
ospfLinkPointToPoint	The neighboring router's router ID.
ospfLinkTransit	The IP address of the Designated Router.
ospfLinkStub	The IP network/subnet number.
ospfLink	The neighboring router's router ID.

linkType

The type of the router link. One of:

Option	Value	Usage
ospfLinkPointToPoint	1	A point-to-point connection to another router.
ospfLinkTransit	2	*(default)* A connection to a transit network.
ospfLinkStub	3	A connection a stub network.
ospfLink	4	A virtual link.

metric

The cost of using the router link, applied to all TOS values.*(default = 0)*

COMMANDS

The **ospfRouterLsaInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfRouterLsaInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfRouterLsaInterface** command.

ospfRouterLsaInterface config *option value*

Modify the configuration options of the ospfRouterLsaInterface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfRouterLsaInterface.

ospfRouterLsaInterface setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfServer*.

SEE ALSO

ospfServer, *ospfInterface*, *ospfRouter*, *ospfRouteRange*, *ospfUserLsaGroup*, *ospfUserLsa*

NAME - ospfServer

ospfServer — accesses the OSPFv2 component of the protocol server for a particular port.

SYNOPSIS

ospfServer *subcommand options*

DESCRIPTION

The *ospfServer* command is necessary in order to access the OSPF protocol server for a particular port. The *select* subcommand **must** be used before all other OSPF commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfServer* for an overview.

Note that this command only applies to the OSPFv2 implementation; the commands related to OSPFv3 all begin with **ospfV3...**.

STANDARD OPTIONS

None

COMMANDS

The **ospfServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfServer addRouter *routerLocalId*

Adds the OSPF router described in the *ospfRouter* command to the list of routers associated with the port. The router's entry in the list is given an identifier of *routerLocalId*. Specific errors are:

- ospfServer select has not been called.
- The router parameters in *ospfRouter* are invalid.
- A router with this *routerLocalId* exists already in the list.

ospfServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfServer** command.

ospfServer clearAllRouters

Deletes all the OSPF routers in the list. Specific errors are:

- ospfServer select has not been called.

ospfServer config *option value*

Modify the configuration options of the ospfServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfServer.

ospfServer delRouter *routerLocalId*

Deletes the OSPF router described that has an identifier of *routerLocalId*. Specific errors are:

- ospfServer select has not been called.
- There is no router with this *routerLocalId* in the list.

ospfServer generateStreams *chasID cardID portID action*

This command creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each enabled route range associated with each router; each stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.

- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

ospfServer getFirstRouter

Access the first OSPF routerin the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- ospfServer select has not been called.
- There are no routers in the list.

ospfServer getNextRouter

Access the next OSPF router in the list. The results may be accessed using the *ospfRouter* command. Specific errors are:

- ospfServer select has not been called.
- ospfServer getFirstRouter has not been called.
- There are no more routers in the list.

ospfServer getRouter *routerLocalId*

Access the OSPF router with an identifier of *routerLocalId.* The results may be accessed using the *ospfRouter* command. Specific errors are:

- ospfServer select has not been called.
- There is no router with this *routerLocalId* in the list.

ospfServer select *chasID cardID portID*

Accesses the OSPF component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The OSPF protocol package has not been installed.
- Invalid port specified.

ospfServer setRouter *routerLocalId*

Sets the values for the router's entry in the list with an identifier of *routerLocalId* based on changes made through the *ospfRouter* command. This command should be used to change a running configuration and must be followed by an *ospfServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *routerLocalId* does not exist in the list.

ospfServer write

Sends any changes made with *ospfRouter setInterface, ospfRouter setRouteRange, ospfRouter setUserLsaGroup* or *ospfServer setRouter* to the protocol server for immediate application. This command **must** be used after those mentioned above in order for their changes to have an effect.

EXAMPLES

```
package req IxTclHal

# Define parameters used by OSPF router

set host localhost

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts “Could not connect to $host”

return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set ch [ixGetChassisID $host]
# Port is: card 4, port 1
set ca 4
set po 1
set pl [list [list $ch $ca $po]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we’ll use
if [ixTakeOwnership $pl] {
ixPuts $::ixErrorInfo
return 1
}
set myMac {00 0a de 01 01 01}
set router 101.101.9.2
set router2 101.101.10.2
set neighbor 101.101.9.1
set interfaceIpMask 255.255.255.0
set ospfInterfaceNetworkType ospfBroadcast
set areaId 0
set numberOfRoute 1650
# Set up the interface table for IPv4 and IPv6 interfaces
# on the port
interfaceTable select $ch $ca $po
interfaceTable clearAllInterfaces

interfaceIpV6 setDefault

interfaceIpV6 config -ipAddress

{0:0:0:0:0:0:0:1}

interfaceIpV6 config -maskWidth 64

interfaceEntry addItem addressTypeIpV6

interfaceIpV4 setDefault

interfaceIpV4 config -ipAddress $router

interfaceIpV4 config -gatewayIpAddress

$neighbor

interfaceIpV4 config -maskWidth 24

interfaceEntry addItem addressTypeIpV4

interfaceEntry setDefault

interfaceEntry config -enable

true

interfaceEntry config -description

{Port 04:01 Interface}

interfaceEntry config -macAddress

$myMac

interfaceTable addInterface

interfaceEntry clearAllItems addressTypeIpV4

interfaceEntry clearAllItems addressTypeIpV6

interfaceIpV6 setDefault

interfaceIpV6 config -ipAddress

{0:0:0:0:0:0:0:2}

interfaceIpV6 config -maskWidth 64

interfaceEntry addItem addressTypeIpV6

interfaceIpV4 setDefault

interfaceIpV4 config -ipAddress

$router2

interfaceIpV4 config -gatewayIpAddress

$neighbor

interfaceIpV4 config -maskWidth 24

interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable
true
interfaceEntry config -description
{Port 04:01 Interface-2}
interfaceEntry config -macAddress
$myMac
interfaceTable addInterface
interfaceTable write
# Select port to operate
ospfServer select $ch $ca $po
# Clear all routers
ospfServer clearAllRouters
# Configure the interface
ospfInterface setDefault
ospfInterface config -enable true
ospfInterface config -connectToDut true
ospfInterface config -protocolInterfaceDescription {Port 04:01
Interface}
ospfInterface config -areaId $areaId
ospfInterface config -networkType ospfBroadcast
ospfInterface config -metric 10
# Add the ospf interface to the router
if [ospfRouter addInterface interface1] {
logMsg “Error in adding ospfInterface interface1”
}
# Configure an interface not connected to the DUT and using a
# Network Range
ospfInterface setDefault
ospfInterface config -enable true
ospfInterface config -connectToDut false
ospfInterface config -protocolInterfaceDescription {Port 04:01
Interface-2}
ospfInterface config -areaId $areaId
ospfInterface config -networkType ospfBroadcast
ospfInterface config -metric 10
ospfInterface config -enableAdvertiseNetworkRange true
# Now the route range
ospfNetworkRange config -entryPointRow 2
ospfNetworkRange config -entryPointColumn 3
ospfNetworkRange config -numRows 10
ospfNetworkRange config -numColumns 10
ospfNetworkRange config -firstRouterId 0.0.1.0
ospfNetworkRange config -routerIdIncrementBy 0.0.0.1
ospfNetworkRange config -firstSubnetIpAddress 192.168.1.0
ospfNetworkRange config -maskWidth 24
# Add the ospf interface to the router
if [ospfRouter addInterface interface2] {
logMsg "Error in adding ospfInterface interface2"
}
# Configure the routeRange
ospfRouteRange setDefault
ospfRouteRange config -enable true
ospfRouteRange config -metric 1
ospfRouteRange config -numberOfNetworks $numberOfRoute
ospfRouteRange config -prefix 24
ospfRouteRange config -networkIpAddress {14.0.0.0}
# Add the ospf routeRange to the router
if [ospfRouter addRouteRange routeRange1] {
logMsg "Error in adding routeRange"
}
# Configure ospf router
ospfRouter setDefault
ospfRouter config -routerId $ca.$po.0.0
ospfRouter config -enable true
# Add the router to the server
if [ospfServer addRouter router1] {
logMsg "Error in adding router"
}
# Let the protocol server respond to ARP, OSPF
protocolServer config -enableArpResponse true
protocolServer config -enableOspfService true
protocolServer config -enablePingResponse false
protocolServer set $ch $ca $po
# Send the data to the hardware
ixWriteConfigToHardware pl
# And start ospf on the port
ixStartOspf pl
# Disable routeRange1 while ospf server is runnung.
# This is the same as removing the route range from router
ospfServer select $ch $ca $po
if [ospfServer getRouter router1] {
logMsg “Error getting router1”
}
if [ospfRouter getRouteRange routeRange1] {
logMsg “Error getting routeRange1”
}
# Disable the route range (You can also change other configuration
if you want)
ospfRouteRange config -enable false
if [ospfRouter setRouteRange routeRange1] {
logMsg “Error setting routeRange1”
}
if [ospfServer write] {
logMsg “Error writing ospfServer”
}
after 10000
#Stop the server at the end
ixStopOspf pl
# If you wanted to add a route range while ospf server is running,
# -Configure it disabled before starting ospf server and then
# enable it
# The same thing is also possible on ospfInterface,
ospfUserLsaGroup
# and ospfUserLsa.
# You just need to get the item that you want and change the
configuration
# and set that item. Then write the changes to hardware by
ospfServer write
# Let go of the ports that we reserved
ixClearOwnership $pl
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

ospfInterface, *ospfRouter*, *ospfRouteRange*, *ospfUserLsaGroup*, *ospfUserLsa*, *ospfRouterLsaInterface*, *ospfServer*

NAME - ospfUserLsa

ospfUserLsa — configures an individual User LSA for use in OSPF.

SYNOPSIS

ospfUserLsa *subcommand options*

DESCRIPTION

The *ospfUserLSA* describes an individual LSA. The types supported are:

- Router LSA — describes router's interfaces with state and cost. This consists of a list of *ospfRouterLSAInterface* elements added via the *ospfUserLSA addInterfaceDescriptionToRouterLsaIdentifier*.
- Network LSA — generated by a designated router and lists all attached routers.
- Summary IP LSA — describes destinations outside of an area.
- Summary ASBR LSA — generated by AS Border Routers and list the ASBR itself.
- Opaque LSA — used to carry information for other protocols and functions, notably OSPF.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfUserLSA* for an overview of this command.

STANDARD OPTIONS

Each of the supported LSA types has a different set of applicable standard options. The common options and the LSA specific options are described in the multiple tables below. All values must be expressed as integers (not hex values, for example). The LSA type is described when the LSA is added to the group using *ospfUserLsaGroup*.

Common

advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

age

Read only. Only available when this command is used to access a learned LSA (See *ospfInterface*). This value holds the age of the LSA extracted from the LSA header.

enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

lsaType

Read-only. The current LSA type. (default = 0)

options

Multiple options may be or'd together. The available options are:

Option	Value	Usage
	0	(default)
ospfOptionBitTypeOfService	0x01	
ospfOptionBitExternalRouting	0x02	
ospfOptionBitMulticast	0x04	
ospfOptionBitNSSACapability	0x08	
ospfOptionBitExternalAttributes	0x10	
ospfOptionBitDemandCircuit	0x20	
ospfOptionBitLSANoForward	0x40	
ospfOptionBitUnused	0x80	

sequenceNumber

Read only. Only available when this command is used to access a learned LSA (see *ospfInterface*). This value holds the sequence number of the LSA extracted from the LSA header.

Router LSA

linkStateId

The router ID of the originating router. *(default = 0.0.0.0)*

routerCapabilityBits

The router's capability bits. Multiple bits are defined; to set more than one bit use the TCL or `|' operator. *(default = 0).* The bits defined are:

Option	Value	Usage
ospfBBit	1	B bit
ospfEBit	2	E bit
ospfVBit	4	V bit

Network LSA

linkStateId

The IP address of the network's Designated Router. *(default = 0.0.0.0)*

neighborId

A list of router IDs in the area, in IP address format separated by spaces. *(default = { })*

Summary IP LSA

incrementLink StateIdBy

If *numberOfLSAs 1*, the value to increment the *linkStateId* by for each LSA.IP format. *(default = 0.0.0.0)*

linkStateId

The IP address of the network. *(default = 0.0.0.0)*

metric

The cost of the route for all TOS levels. *(default = 0)*

networkMask

The destination network's IP address mask. *(default = 0.0.0.0)*

numberOfLSAs

The number of Summary IP LSAs to generate. *(default = 0)*

Summary ASBR LSA

linkStateId

The router ID of the AS Boundary router. *(default = 0.0.0.0)*

External LSA

externalMetricEBit

The value of the external metric's E-bit. *(default = 0)*

externalRouteTag

The type of external metric. A value of 1 implies type 2 metric, in which case the *metric* value is larger than any internal metric. A value of 0 implies type 1 metrics, in which case the *metric* value is expressed in the same units as internal metrics. *(default = 0.0.0.0)*

forwardingAddress

Data traffic for the advertised destination will be forwarded to this address. If set to 0.0.0.0, then traffic will be forwarded to the AS boundary router itself. *(default = 0.0.0.0)*

incrementLink StateIdBy

If *numberOfLSAs 1*, the value to increment the *linkStateId* by for each LSA.IP format. *(default = 0.0.0.0)*

linkStateId

The IP network number of the external destination. 0.0.0.0 is used to indicate a default route. *(default = 0.0.0.0)*

metric

The cost of the route, applied to all TOS values. *(default = 0)*

networkMask

The IP address mask for the advertised destination. 0.0.0.0 is used to indicate a default route. *(default = 0.0.0.0)*

numberOfLSAs

The number of External LSAs to generate. *(default = 0)*

Opaque Local, Opaque Area and Opaque Domain LSAs

linkStateId

The high-order 8 bits indicate the Opaque type as defined in RFC 2370. The low-order 24 bits indicate the type-specific ID. *(default = 0.0.0.0)*

tlvType

Indicates the type of TLV (type-length-value) to be generated. This should be set to one of:

Option	Value	Usage
ospfRouterTlv	0	(default)
ospfLinkTlv	1	

For Router TLVs

tlvRouterIpAddress

The stable IP address of the advertising router that is always reachable if there is any connectivity to it. This is typically implemented as a "loopback address". *(default = 0.0.0.0)*

For Link TLVs

enableTlvLinkId true / false

Enables the generation of a sub-TLV for the Link ID, based on the value of the *tlvLinkId. (default = false)*

enableTlvLinkMetric true / false

Enables the generation of a sub-TLV for the Link Metric, based on the value of the *tlvLinkMetric. (default = false)*

enableTlvLinkType true / false

Enables the generation of a sub-TLV for the Link Type, based on the value of the *tlvLinkType. (default = false)*

enableTlvLocalIp Address true / false

Enables the generation of a sub-TLV for the Local IP address, based on the value of the *tlvLocalIpAddress. (default = false)*

enableTlvMax Bandwidth true / false

Enables the generation of a sub-TLV for the Max Bandwidth, based on the value of the *tlvMaxBandwidth. (default = false)*

enabletlvMax ReservableBandwidth true / false

Enables the generation of a sub-TLV for the Max Reservable Bandwidth, based on the value of the *tlvMaxReservableBandwidth. (default = false)*

enableTlvRemoteIp Address true / false

Enables the generation of a sub-TLV for the Remote IP Address, based on the value of the *tlvRemoteIpAddress. (default = false)*

enableTlvResource Class true / false

Enables the generation of a sub-TLV for the Resrouce Class, based on the value of the *tlvResourceClass. (default = false)*

enableTlvUnreserved Bandwidth true / false

Enables the generation of a sub-TLV for the Unreserved Bandwidth, based on the value of the *tlvUnreservedBandwidthPriority0-7. (default = false)*

tlvLinkId

Identifies the other end of the link. For point-to-point links, thie is the ROuter ID of the neighbor. For multiaccess link, sthis is the interface address of the designated router. *(default = 0.0.0.0)*

tlvLinkMetric

The link metric for traffic engineering purposes. *(default = 0)*

tlvLinkType

The type of the link, one of:

Option	Value	Usage
ospfTlvLinkPointToPoint	1	(default)
ospfTlvLinkMultiAccess	2	

tlvLocalIpAddress

The IP address(es) of the interface corresponding to this link. If there are multiple local addresses on the link, they are all listed in this Tlv. *(default = 0.0.0.0)*

tlvMaxBandwidth

The maximum bandwidth that can be used on this link in this direction (from the system originating the LSA to its neighbor). This is the true link capacity expressed in bytes per second. *(default = 0.0)*

tlvMaxReservable Bandwidth

The maximum bandwidth that may be reserved on this link in this direction, (from the system originating the LSA to its neighbor). Note that this may be greater than the maximum bandwidth (in which case the link may be oversubscribed). Units are in bytes per second. *(default = 0.0)*

tlvRemoteIpAddress

The IP address(es) of the neighbor's interface corresponding to this link. This and the local address are used to discern multiple parallel links between systems. *(default = 0.0.0.0)*

tlvResourceClass

Specifies the administrative group membership for this link, in terms of a bit mask. A link that is a member of multiple groups will have multiple bits sent. *(default = {00 00 00 00})*

tlvUnreserved BandwidthPriority0-7

The amount of bandwidth not yet reserved, at each of eight priority levels. Each value must be less than or equal to the maximum reservable bandwidth. Units are in bytes per second. *(default = 0.0)*

COMMANDS

The **ospfUserLsa** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfUserLsa addInterfaceDescriptionToRouterLsa

This command is for use with Router LSAs only. Adds an interface to the list of interfaces for the Router LSA described in the *ospfRouterLsaInterface* command. Specific errors are:

- The parameters in *ospfRouterLsaInterface* are invalid

ospfUserLsa addInterfaceDescriptionToRouterLsaIdentifier *interface name*

This command is for use with Router LSAs only. Adds an interface, specified by interface name, to the list of interfaces for the Router LSA described in the *ospf RouterLsaInterface* command.

ospfUserLsa cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfUserLsa** command.

ospfUserLsa clearAllRouterLsaInterface

This command is for use with Router LSAs only. Deletes all of the interfaces from the user Router LSAs.

ospfUserLsa config *option value*

Modify the configuration options of the ospfUserLsa. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfUserLsa.

ospfUserLsa delInterfaceDescriptionToRouterLsa *interface name*

This command is for use with Router LSAs only. Deletes a particular interface, specified by interface name (and added with the *addInterfaceDescriptionToRouterLsaIdentifier command*), from the list of interfaces for the Router LSA described in the *ospfRouterLsaInterface* command.

ospfUserLsa getFirstRouterLsaInterface

This command is for use with Router Interface LSAs only. It accesses the first Router Interface LSA entry in the list. The user LSA is accessed in the *ospfRouterLsaInterface* command. Specific errors are:

- The type of this *ospfUserLsa* entry is not *ospfLsaRouter*
- There are no items in the list.

ospfUserLsa getNextRouterLsaInterface

This command is for use with Router Interface LSAs only. It accesses the next Router Interface LSA entry in the list. *getFirstRouterLsaInterface* must be called before any calls to this command. The user LSA is accessed in the *ospfRouterLsaInterface* command. Specific errors are:

- There are no more items in the list.

ospfUserLsa setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfServer*.

SEE ALSO

ospfServer, *ospfInterface*, *ospfRouter*, *ospfRouteRange*, *ospfUserLsaGroup*, *ospfRouterLsaInterface*

NAME - ospfUserLsaGroup

ospfUserLsaGroup — configures a User LSA group for use in OSPF.

SYNOPSIS

ospfUserLsaGroup *subcommand options*

DESCRIPTION

The *ospfUserLSAGroup* describes a list of LSAs to be associated with advertised routes. The list consists of elements constructed through the use of the *ospfUserLsa* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfUserLSAGroup* for an overview of this command.

STANDARD OPTIONS

areaId

The area ID for the LSA group. *(default = 0)*

description

A commentary description for the user LSA group. *(default = "")*

enabletrue / false

Enables the use of this router in the simulated OSPF network. *(default = false)*

COMMANDS

The **ospfUserLsaGroup** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfUserLsaGroup addUserLsa *userLsaLocalId lsaType*

Adds the user LSA described in the *ospfUserLsa* command and of type *lsaType* to the list. The LSA's entry in the list is given an identifier of *userLsaLocalId*. The choices of *lsaType* are:

Option	Value	Usage
ospfLsaRouter	1	
ospfLsaNetwork	2	
ospfLsaSummaryIp	3	
ospfLsaSummaryAs	4	
ospfLsaExternal	5	
ospfLsaOpaqueLocal	9	
ospfLsaOpaqueArea	10	
ospfLsaOpaqueDomain	11	

Specific errors are:

- The LSA type in *ospfUserLsa* is not valid.
- The parameters in *ospfUserLsa* are invalid.
- A router with this *userLsaLocalId* exists already in the list.

ospfUserLsaGroup cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfUserLsaGroup** command.

ospfUserLsaGroup clearAllUserLsas

Deletes all of the user LSAs.

ospfUserLsaGroup config *option value*

Modify the configuration options of the ospfUserLsaGroup. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfUserLsaGroup.

ospfUserLsaGroup delUserLsa *userLsaLocalId*

Deletes the user LSA with an identifier of *userLsaLocalId*. Specific errors are:

- No LSA with this *userLsaLocalId* exists in the list.

ospfUserLsaGroup getFirstUserLsa

Access the first user LSA in the list. The results may be accessed using the *ospfUserLsa* command. Specific errors are:

- There are no routers in the list.

ospfUserLsaGroup getNextUserLsa

Access the next user LSA in the list*.* The results may be accessed using the *ospfUserLsa* command. Specific errors are:

- ospfUserLsaGroup getFirstUserLsa has not been called.
- There is no more LSAs in the list.

ospfUserLsaGroup getUserLsa *userLsaLocalId*

Accesses the LSA's entry in the list with an identifier of *userLsaLocalId*. The user LSA is accessed in the *ospfUserLsa* command. Specific errors are:

- No LSA with this *userLsaLocalId* exists in the list.

ospfUserLsaGroup setDefault

Sets default values for all configuration options.

ospfUserLsaGroup setInterface *userLsaLocalId*

Sets the values for the user LSA's entry in the list with an identifier of *userLsaLocalId* based on changes made through the *ospfUserLsa* command. This command can be used to change a running configuration and must be followed by an *ospfServer write* command in order to send these changes to the protocol server. Specific errors are:

- A user LSA with this *ospfUserLsa* does not exist in the list.

EXAMPLES

See examples under *ospfServer*.

SEE ALSO

ospfServer, *ospfInterface*, *ospfRouter*, *ospfRouteRange*, *ospfUserLsa*, *ospfRouterLsaInterface*

NAME - ospfV3Interface

ospfV3Interface — configures an interface for an OSPFv3 router.

SYNOPSIS

ospfV3Interface *subcommand options*

DESCRIPTION

The *ospfV3Interface* command holds the information related to a single interface on the simulated router. Interfaces are added into the *ospfV3Router* interface list using the *ospfV3Router addInterface* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3Interface* for an overview.

Note that this command only applies to the OSPFv3 implementation; the commands related to OSPF all omit the **V3** letters in their name.

STANDARD OPTIONS

areaId

The OSPF area ID associated with the interface. *(default = 0)*

deadInterval

The time after which the DUT router is considered dead if it does not send HELLO messages. *(default = 40)*

enable true | false

Enables the use of the simulated interface. *(default = false)*

enableBfdRegistration true | false

Indicates if a BFD session is to be created to the OSPFv3 peer IP address once the OSPFv3 session is established. This allows OSPFv3 to use BFD to maintain IPv4 connectivity the OSPFv3 peer.

enableIgnoreDbDescMTU true | false

When true, enbles the ability of OSPFv3 to ignore the database set maximum transmission unit (MTU). *(default = false)*

helloInterval

The time between HELLO messages sent over the interface. *(default = 10)*

instanceId

The instance ID for the interface. *(default = 0)*

options

Options related to the interface. Multiple options may be or'd together. *(default = 0x13)* The available options are:

Option	Value	Usage
ospfV3InterfaceOptionDCBit	0x20	
ospfV3InterfaceOptionRBit	0x10	
ospfV3InterfaceOptionNBit	0x08	
ospfV3InterfaceOptionMCBit	0x04	
ospfV3InterfaceOptionEBit	0x02	
ospfV3InterfaceOptionV6Bit	0x01	

protocolInterface Description

The *description* option associated with an *interfaceEntry* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

type

The type of the interface. One of the following options

Option	Value	Usge
ospfV3InterfacePointToPoint	0x01	
ospfV3InterfaceBroadcast	0x02	(default)

COMMANDS

The **ospfV3Interface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3Interface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3Interface** command.

ospfV3Interface config *option value*

Modify the configuration options of the ospfV3Interface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3Interface.

ospfV3Interface setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3IpV6Prefix, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3IpV6Prefix

ospfV3IpV6Prefix — configures an address prefix to be used.

SYNOPSIS

ospfV3IpV6Prefix *subcommand options*

DESCRIPTION

The *ospfV3IpV6Prefix* command is used to set up and read address prefixes for use in the *ospfV3LsaIntraAreaPrefix*and *ospfV3LsaLink* commands.

STANDARD OPTIONS

address

The prefix address to be advertised in the LSA. Although only *length* bits of the IPv6 address are meaningful, a full IPv6 address should be specified. The *ipV6Address* command can be used to construct the address. *(default = 0:0:0:0:0:0:0:0)*

incrementBy

If *numLsaToGenerate* in the *ospfV3LsaIntraAreaPrefix*/*ospfV3LsaLink* command is greater than 1, this is the value that will be added to the most significant *length* bits of *address* between generated LSAs. *(default = 1)*

length

The number of high-order bits of *address* that are significant. *(default = 64)*

options

An 8-bit quantity with options related to the *address. (default = 0)* Multiple bits may be or'd together:

Option	Value	Usage
ospfV3PrefixOptionPBit	0x08	The *propagate* bit, which is set on NSSA area prefixes that should be re-advertised at the NSSA area border.
ospfV3PrefixOptionMCBit	0x04	The multicast capability bit, which should be set if the prefix should be included in IPv6 multicast routing calculations.
ospfV3PrefixOptionLABit	0x02	The *local address capability* bit, which should be set if the prefix is actually an IPv6 interface address of the advertising router.
ospfV3PrefixOptionNUBit	0x01	The *no unicast* bit, which should be set if the prefix should be excluded from IPv6 unicast calculations.

COMMANDS

The **ospfV3IpV6Prefix** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3IpV6Prefix cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3IpV6Prefix** command.

ospfV3IpV6Prefix config *option value*

Modify the configuration options of the ospfV3IpV6Prefix. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *ospfV3IpV6Prefix*.

ospfV3IpV6Prefix setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaAsExternal

ospfV3LsaAsExternal — configures an AS External LSA for use in OSPFv3.

SYNOPSIS

ospfV3LsaAsExternal *subcommand options*

DESCRIPTION

The *ospfV3LsaAsExternal* command is used to set up an AS External LSA. These LSAs are originated by ASBRs to describe the IPv6 address prefixes that are external to the local AS, with one LSA per address prefix.

Refer to *the Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaAsExternal* for an overview of this command.

STANDARD OPTIONS

The standard options marked with a * are common to all of the *ospfV3Lsa* commands.

*advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

*enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

enableEBit true / false

The type of external metric. A value of 1 implies type 2 metric, in which case the *metric* value is larger than any internal metric. A value of 0 implies type 1 metrics, in which case the *metric* value is expressed in the same units as internal metrics. *(default = false)*

enableFBit true / false

The value of the external metric's F-bit. If *true,* then the *forwardingAddress* field is to be included in the LSA. *(default = false)*

enableTBit true / false

The value of the external metric's T-bit. If *true,* then the *externalRouteTag* field will be included in the LSA. *(default = false)*

externalRouteTag

If the *enableTBit* is *true,* an additional value to be used for external routes between AS boundary routers. This field is not used within OSPF. *(default = 0.0.0.0)*

forwardingAddress

If the *enableFBit*is *true,* data traffic for the advertised destination will be forwarded to this fully qualified IPv6 address. *(default = 0:0:0:0:0:0:0:0)*

*incrementLink StateIdBy

If *numberLsaToGenerate* is greater than 1, the value to increment the *linkStateId* by for each LSA. The value is expressed in IPv4 format. *(default = 0.0.0.0)*

incrementPrefixBy

If *numLsaToGenerate* is greater than 1, this is the value that will be added to the most significant *prefixLength* bits of *prefixAddress* between generated LSAs. *(default = 1)*

*linkStateId

A unique value to be associated with the LSA. Each of the generated LSAs may have a unique value, as determined by the *incrementLinkStateIdBy* and *numLsaToGenerate* options. *(default = 0.0.0.0)*

metric

The cost of the route. *(default = 0)*

*numLsaToGenerate

The number of LSAs to generate, each with potentially different Link State IDs determined by the value of the *incrementLinkStateIdBy* value. *(default = 1)*

prefixAddress

The prefix address to be advertised in the LSA. Although only *prefixLength* bits of the IPv6 address are meaningful, a full IPv6 address should be specified. The *ipV6Address* command can be used to construct the address. *(default = 0:0:0:0:0:0:0:0)*

prefixLength

The number of high-order bits of *prefixAddress* that are significant. *(default = 64)*

prefixOptions

An 8-bit quantity with options related to the *prefixAddress. (default = 0)* Multiple bits may be or'd together:

Option	Value	Usage
ospfV3PrefixOptionPBit	0x08	The *propagate* bit, which is set on NSSA area prefixes that should be re-advertised at the NSSA area border.
ospfV3PrefixOptionMCBit	0x04	The multicast capability bit, which should be set if the prefix should be included in IPv6 multicast routing calculations.
ospfV3PrefixOptionLABit	0x02	The *local address capability* bit, which should be set if the prefix is actually an IPv6 interface address of the advertising router.
ospfV3PrefixOptionNUBit	0x01	The *no unicast* bit, should be set if the prefix should be excluded from IPv6 unicast calculations.

referencedLinkStateId

If *referencedType* is non-zero, this is the link state ID of the type *referencedType* that is referenced by this LSA. *(default = 0.0.0.0)*

referencedType

If non-zero, this is the type of a different LSA referenced by this LSA. The value of *referencedLinkStateId* indicates which particular LSA of that type is referenced. *(default = 0)*

*type

Read-only. The type of LSA; always *$::ospfV3LsaAsExternal*.

COMMANDS

The **ospfV3LsaAsExternal** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaAsExternal cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaAsExternal** command.

ospfV3LsaAsExternal config *option value*

Modify the configuration options of the ospfV3LsaAsExternal. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaAsExternal.

ospfV3LsaAsExternal setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaInterAreaPrefix

ospfV3LsaInterAreaPrefix — configures an Inter Area Prefix LSA for use in OSPFv3.

SYNOPSIS

ospfV3LsaInterAreaPrefix *subcommand options*

DESCRIPTION

The *ospfV3LsaInterAreaPrefix* command is used to build an Inter Area Prefix LSA. This type of LSA is originated by area border routers to describe routes to IPv6 address prefixes in other areas, with one LSA per prefix.

Refer to *the Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaInterAreaPrefix* for an overview of this command.

STANDARD OPTIONS

The standard options marked with a * are common to all of the *ospfV3Lsa* commands.

*advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

*enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

*incrementLink StateIdBy

If *numberLsaToGenerate* is greater than 1, the value to increment the *linkStateId* by for each LSA. The value is expressed in IPv4 format. *(default = 0.0.0.0)*

*linkStateId

A unique value to be associated with the LSA. Each of the generated LSAs may have a unique value, as determined by the *incrementLinkStateIdBy* and *numLsaToGenerate* options. *(default = 0.0.0.0)*

*numLsaToGenerate

The number of LSAs to generate, each with potentially different Link State IDs determined by the value of the *incrementLinkStateIdBy* value. *(default = 1)*

prefixAddress

The prefix address to be advertised in the LSA. Although only *prefixLength* bits of the IPv6 address are meaningful, a full IPv6 address should be specified. The *ipV6Address* command can be used to construct the address. *(default = 0:0:0:0:0:0:0:0)*

prefixLength

The number of high-order bits of *prefixAddress* that are significant. *(default = 64)*

prefixOptions

An 8-bit quantity with options related to the *prefixAddress. (default = 0)* Multiple bits may be or'd together:

Option	Value	Usage
ospfV3PrefixOptionPBit	0x08	The *propagate* bit, which is set on NSSA area prefixes that should be re-advertised at the NSSA area border.
ospfV3PrefixOptionMCBit	0x04	The multicastcapability bit, which should be set if the prefix should be included in IPv6 multicast routing calculations.
ospfV3PrefixOptionLABit	0x02	The *local address capability* bit, which should be set if the prefix is actually an IPv6 interface address of the advertising router.
ospfV3PrefixOptionNUBit	0x01	The *no unicast* bit, which should be set if the prefix should be excluded from IPv6 unicast calculations.

*type

(Read-only.) The type of LSA; always *$::ospfV3LsaInterAreaPrefix*.

COMMANDS

The **ospfV3LsaInterAreaPrefix** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaInterAreaPrefix cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaInterAreaPrefix** command.

ospfV3LsaInterAreaPrefix config *option value*

Modify the configuration options of the ospfV3LsaInterAreaPrefix. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaInterAreaPrefix.

ospfV3LsaInterAreaPrefix setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaInterAreaRouter

ospfV3LsaInterAreaRouter — configures an Inter Area Router LSA for use in OSPFv3.

SYNOPSIS

ospfV3LsaInterAreaRouter *subcommand options*

DESCRIPTION

The *ospfV3LsaInterAreaRouter* command is used to build an Inter Area Router LSA. This type of LSA is generated by area border routers to describe the OSPFv3 routers (ASBRs) in other areas, with one LSA per router.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaInterAreaRouter* for an overview of this command.

STANDARD OPTIONS

The standard options marked with a * are common to all of the *ospfV3Lsa* commands.

*advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

destinationRouterId

The router ID of the destination router. *(default = 0.0.0.0)*

*enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

incrementDest RouterIdBy

If *numLsaToGenerate* is greater than 1, this is the value that will be added to *destinationRouterId* between generated LSAs. *(default = 0.0.0.1)*

*incrementLink StateIdBy

If *numberLsaToGenerate* is greater than 1, the value to increment the *linkStateId* by for each LSA. The value is expressed in IPv4 format. *(default = 0.0.0.0)*

*linkStateId

A unique value to be associated with the LSA. Each of the generated LSAs may have a unique value, as determined by the *incrementLinkStateIdBy* and *numLsaToGenerate* options. *(default = 0.0.0.0)*

metric

The metric cost of this route. *(default = 0)*

*numLsaToGenerate

The number of LSAs to generate, each with potentially different Link State IDs determined by the value of the *incrementLinkStateIdBy* value. *(default = 1)*

options

The 24-bit options field associated with the destination router. *(default = 0)* Multiple bits may be or'd together.

Option	Value	Usage
ospfV3LsaOptionV6Bit	0x01	Indicates whether to include this router/link for IPv6 routing calculations (1) or not (0).
ospfV3LsaOptionEBit	0x02	Indicates that the originating router is capable of receiving AS External LSAs (1) or not (0).
ospfV3LsaOptionMCBit	0x04	Indicates that multicast datagrams are to be forwarded (1) or not (0).
ospfV3LsaOptionNBit	0x08	Indicates bit handling of Type 7 (LSAs). Not yet supported.
ospfV3LsaOptionRBit	0x10	Indicates that the originator is an active router (1) or not (0).
ospfV3LsaOptionDCBit	0x20	Demand circuit handling. Not yet supported.

*type

(Read-only.) The type of LSA; always *$::ospfV3LsaInterAreaRouter*.

COMMANDS

The **ospfV3LsaInterAreaRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaInterAreaRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaInterAreaRouter** command.

ospfV3LsaInterAreaRouter config *option value*

Modify the configuration options of the ospfV3LsaInterAreaRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaInterAreaRouter.

ospfV3LsaInterAreaRouter setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaIntraAreaPrefix

ospfV3LsaIntraAreaPrefix — configures an Inter Area Router LSA for use in OSPFv3.

SYNOPSIS

ospfV3LsaIntraAreaPrefix *subcommand options*

DESCRIPTION

The *ospfV3LsaIntraAreaPrefix* command is used to build a Inter Area Router LSA. This type of LSA is used to advertise one or more IPv6 address prefixes that are associated with the router itself, an attached stub network segment or an attached transit network segment. A prefix is defined with the *ospfV3IpV6Prefix* command and then added into the LSA with the *ospfV3LsaIntraAreaPrefix addPrefix* subcommand.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaIntraAreaPrefix* for an overview of this command.

STANDARD OPTIONS

The standard options marked with a * are common to all of the *ospfV3Lsa* commands.

*advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

*enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

*incrementLink StateIdBy

If *numberLsaToGenerate* is greater than 1, the value to increment the *linkStateId* by for each LSA. The value is expressed in IPv4 format. *(default = 0.0.0.0)*

*linkStateId

A unique value to be associated with the LSA. Each of the generated LSAs may have a unique value, as determined by the *incrementLinkStateIdBy* and *numLsaToGenerate* options. *(default = 0.0.0.0)*

*numLsaToGenerate

The number of LSAs to generate, each with potentially different Link State IDs determined by the value of the *incrementLinkStateIdBy* value. *(default = 1)*

referencedLinkStateId

If *referencedType* is non-zero, this is the link state ID of the type *referencedType* that is referenced by this LSA. *(default = 0.0.0.0)*

referencedRouterId

If *referencedType* is non-zero, this is the router ID that is referenced by this LSA. *(default = 0.0.0.0)*

referencedType

If non-zero, this is the type of a different LSA referenced by this LSA. The value of *referencedLinkStateId* indicates which particular LSA of that type is referenced. *(default = 1)*

*type

(Read-only) The type of LSA; always *$::ospfV3LsaIntraAreaPrefix*.

COMMANDS

The **ospfV3LsaIntraAreaPrefix** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaIntraAreaPrefix addPrefix

Adds an IPv6 address prefix to the list of prefixes, using the data from the *ospfV3IpV6Prefix* command.

ospfV3LsaIntraAreaPrefix cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaIntraAreaPrefix** command.

ospfV3LsaIntraAreaPrefix config *option value*

Modify the configuration options of the ospfV3LsaIntraAreaPrefix. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaIntraAreaPrefix.

ospfV3LsaIntraAreaPrefix clearPrefixList

Removes all of the IPv6 addresses from the LSA's list.

ospfV3LsaIntraAreaPrefix delPrefix

Deletes the currently accessed address prefix, as determined by use of *getFirstPrefix* and *getNextPrefix.* Errors include:

- *getFirstPrefix* has not been called.
- *getNextPrefix* has run off the end of the list.

ospfV3LsaIntraAreaPrefix getFirstPrefix

Accesses the first prefix in the list. The options for the prefix can be accessed through the use of the *ospfV3IpV6Prefix* command. Errors include:

- There are no prefixes in the list.

ospfV3LsaIntraAreaPrefix getNextPrefix

Accesses the next prefix in the list. The options for the prefix can be accessed through the use of the *ospfV3IpV6Prefix* command. Errors include:

- *getFirstPrefix* has not been called.
- There are no more prefixes in the list.

ospfV3LsaIntraAreaPrefix setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaLink

ospfV3LsaLink — configures a Link LSA for use in OSPFv3.

SYNOPSIS

ospfV3LsaLink *subcommand options*

DESCRIPTION

The *ospfV3LsaLink* command is used to build a Link LSA. This type of LSA is generated by routers to describe an attached link, one LSA per link. A prefix is defined with the *ospfV3IpV6Prefix* command and then added into the LSA with the *ospfV3LsaLink addPrefix* subcommand.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaLink* for an overview of this command.

STANDARD OPTIONS

The standard options marked with a * are common to all of the *ospfV3Lsa* commands.

*advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

*enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

*incrementLink StateIdBy

If *numberLsaToGenerate* is greater than 1, the value to increment the *linkStateId* by for each LSA. The value is expressed in IPv4 format. *(default = 0.0.0.0)*

linkLocalAddress

The IPv6 link local address for the interface. *(default = 0:0:0:0:0:0:0:0)*

*linkStateId

The interface ID associated with the link. *(default = 0.0.0.0)*

*numLsaToGenerate

The number of LSAs to generate, each with potentially different Link State IDs determined by the value of the *incrementLinkStateIdBy* value. *(default = 1)*

options

The 24-bit options field associated with the destination router.*(default = 0)* Multiple bits may be or'd together.

Option	Value	Usage
ospfV3LsaOptionV6Bit	0x01	Indicates whether to include this router/link for IPv6

Option	Value	Usage
		routing calculations (1) or not (0).
ospfV3LsaOptionEBit	0x02	Indicates that the originating router is capable of receiving AS External LSAs (1) or not (0).
ospfV3LsaOptionMCBit	0x04	Indicates that multicast datagrams are to be forwarded (1) or not (0).
ospfV3LsaOptionNBit	0x08	Indicates bit handling of Type 7 (LSAs). Not yet supported.
ospfV3LsaOptionRBit	0x10	Indicates that the originator is an active router (1) or not (0).
ospfV3LsaOptionDCBit	0x20	Demand circuit handling. Not yet supported.

priority

The router's priority for the interface to be used in Designated Router election. *(default = 1)*

*type

(Read-only) The type of LSA; always *$::ospfV3LsaLink*.

COMMANDS

The **ospfV3LsaLink** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaLink addPrefix

Adds an IPv6 address prefix to the list of prefixes, using the data from the *ospfV3IpV6Prefix* command.

ospfV3LsaLink cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaLink** command.

ospfV3LsaLink config *option value*

Modify the configuration options of the ospfV3LsaLink. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaLink.

ospfV3LsaLink clearPrefixList

Removes all of the IPv6 addresses from the LSA's list.

ospfV3LsaLink delPrefix

Deletes the currently accessed address prefix, as determined by use of *getFirstPrefix* and *getNextPrefix.* Errors include:

- *getFirstPrefix* has not been called.
- *getNextPrefix* has run off the end of the list.

ospfV3LsaLink getFirstPrefix

Accesses the first prefix in the list. The options for the prefix can be accessed through the use of the *ospfV3IpV6Prefix* command. Errors include:

- There are no prefixes in the list.

ospfV3LsaLink getNextPrefix

Accesses the next prefix in the list. The options for the prefix can be accessed through the use of the *ospfV3IpV6Prefix* command. Errors include:

- *getFirstPrefix* has not been called.
- There are no more prefixes in the list.

ospfV3LsaLink setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaNetwork

ospfV3LsaNetwork — configures a Network LSA for use in OSPFv3.

SYNOPSIS

ospfV3LsaNetwork *subcommand options*

DESCRIPTION

The *ospfV3LsaNetwork* command is used to build a Network LSA. This type of LSA is sent by the Designated Router for an area that supports two or more routers. It describes all routers attached to the network, including the Designated Router.

Refer to *the Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaNetwork* for an overview of this command.

STANDARD OPTIONS

The standard options marked with a * are common to all of the *ospfV3Lsa* commands.

*advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

*enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

*linkStateId

The interface ID for the link that was previously advertised by the Designated Router in its Hello message. *(default =0.0.0.0)*

neighborRouterIdList

A space separated list of router IDs for all the routers on the link. Each router is in IPv4 format. For example, {10.1.0.1 192.168.36.2}. *(default = {})*

options

The 24-bit options field associated with the destination router. *(default = 0)* Multiple bits may be or'd together.

Option	Value	Usage
ospfV3LsaOptionV6Bit	0x01	Indicates whether to include this router/link for IPv6 routing calculations (1) or not (0).
ospfV3LsaOptionEBit	0x02	Indicates that the originating router is capable of receiving AS External LSAs (1) or not (0).
ospfV3LsaOptionMCBit	0x04	Indicates that multicast datagrams are to be forwarded (1) or not (0).
ospfV3LsaOptionNBit	0x08	Indicates bit handling of Type 7 (LSAs). Not yet supported.

Option	Value	Usage
ospfV3LsaOptionRBit	0x10	Indicates that the originator is an active router (1) or not (0).
ospfV3LsaOptionDCBit	0x20	Demand circuit handling. Not yet supported.

*type

(Read-only) The type of LSA; always *$::ospfV3LsaNetwork*.

COMMANDS

The **ospfV3LsaNetwork** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaNetwork cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaNetwork** command.

ospfV3LsaNetwork config *option value*

Modify the configuration options of the ospfV3LsaNetwork. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaNetwork.

ospfV3LsaNetwork setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaRouter

ospfV3LsaRouter — configures a Router LSA for use in OSPFv3.

SYNOPSIS

ospfV3LsaRouter *subcommand options*

DESCRIPTION

The *ospfV3LsaRouter* command is used to build a Router LSA. This type of LSA is generated by every router to describe the state and cost of the router's interfaces to the area. Each interface is constructed using the *ospfV3LsaRouterInterface* command and then added to this LSA's interface list using the *addInterface* subcommand.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaRouter* for an overview of this command.

STANDARD OPTIONS

The standard options marked with an * are common to all of the *ospfV3Lsa* commands.

*advertisingRouterId

The router ID of the router that is originating the LSA. *(default = 0.0.0.0)*

*enable true / false

Indicates whether the LSA is to be used in the simulation. *(default = false)*

enableAutoPopulate NeighborInfo true | false

If *true,* and the value of an *interfaceId* in an *ospfV3LsaRouterInterface* object added via *addInterface* is **1**, then the *neighborId* and *neighborRouterId* of that object will be ignored and replaced with the DUT's actual values as run time. *(default = false)*

enableBBit true / false

Indicates that the router is an area border router. *(default = false)*

enableEBit true / false

Indicates that the router is an AS boundary router (ASBR). *(default = false)*

enableVBit true / false

Indicates that the router is an endpoint of one or more fully adjacent virtual links. *(default = false)*

enableWBit true / false

Indicates that the router is a wild-card multicast receiver and will receive multicast datagrams regardless of destination. *(default = false)*

*linkStateId

A unique value to be associated with the LSA. Each of the generated LSAs may have a unique value, as determined by the *incrementLinkStateIdBy* and *numLsaToGenerate* options. *(default* = 0.0.0.0*)*

options

The 24-bit options field associated with the destination router. *(default = 0)* Multiple bits may be or'd together.

Option	Value	Usage
ospfV3LsaOptionV6Bit	0x01	Indicates whether to include this router/link for IPv6 routing calculations (1) or not (0).
ospfV3LsaOptionEBit	0x02	Indicates that the originating router is capable of receiving AS External LSAs (1) or not (0).
ospfV3LsaOptionMCBit	0x04	Indicates that multicast datagrams are to be forwarded (1) or not (0).
ospfV3LsaOptionNBit	0x08	Indicates bit handling of Type 7 (LSAs). Not yet supported.
ospfV3LsaOptionRBit	0x10	Indicates that the originator is an active router (1) or not (0).
ospfV3LsaOptionDCBit	0x20	Demand circuit handling. Not yet supported.

*type

(Read-only) The type of LSA; always *$::ospfV3LsaRouter*.

COMMANDS

The **ospfV3LsaRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaRouter addInterface

Adds an interface to the list of interfaces, using the data from the *ospfV3LsaRouterInterface* command.

ospfV3LsaRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaRouter** command.

ospfV3LsaRouter config *option value*

Modify the configuration options of the ospfV3LsaRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaRouter.

ospfV3LsaRouter clearAllInterfaces

Removes all of the interfaces from the LSA's list.

ospfV3LsaRouter delInterface

Deletes the currently accessed interface, as determined by use of *getFirstInterface* and *getNextInterface.* Errors include:

- *getFirstInterface* has not been called.
- *getNextInterface* has run off the end of the list.

ospfV3LsaRouter getFirstInterface

Accesses the first interface in the list. The options for the interface can be accessed through the use of the *ospfV3LsaRouterInterface* command. Errors include:

- There are no interfaces in the list.

ospfV3LsaRouter getNextInterface

Accesses the next interface in the list. The options for the interface can be accessed through the use of the *ospfV3LsaRouterInterface* command. Errors include:

- *getFirstInterface* has not been called.
- There are no more interfaces in the list.

ospfV3LsaRouter setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3LsaRouterInterface

ospfV3LsaRouterInterface — configures an interface for use in a Router LSA.

SYNOPSIS

ospfV3LsaRouterInterface *subcommand options*

DESCRIPTION

The *ospfV3LsaRouterInterface* command is used to construct an interface descriptor for use in the *ospfV3LsaRouter* command.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPF testing with Ixia equipment. Refer to *ospfV3LsaRouterInterface* for an overview of this command.

STANDARD OPTIONS

interfaceId

The router defined interface ID for the interface. *(default = 0)*

metric

The cost of using this router interface for outbound traffic. *(default = 0)*

neighborInterfaceId

The interface ID that the neighbor router (or the attached link's Designated Router when *type* is *ospfV3LsaRouterInterfaceBroadcast*) has been advertising in Hello packets sent on the attached link. *(default = 0)*

neighborRouterId

The router ID of the neighbor router, or the attached link's Designated Router when *type* is *ospfV3LsaRouterInterfaceBroadcast*. *(default = 0.0.0.0)*

type

The type of the interface. One of:

Option	Value	Usage
ospfV3LsaRouterInterfacePointToPoint	1	Point to point connection to another router.
ospfV3LsaRouterInterfaceTransit	2	Connection to a transit network (broadcast or NMBA).
ospfV3LsaRouterInterfaceVirtual	4	*(default)* A virtual link.

COMMANDS

The **ospfV3LsaRouterInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3LsaRouterInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3LsaRouterInterface** command.

ospfV3LsaRouterInterface config *option value*

Modify the configuration options of the ospfV3LsaRouterInterface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3LsaRouterInterface.

ospfV3LsaRouterInterface setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3NetworkRange

ospfV3NetworkRange — sets up the parameters associated with an OSPFv3 network range.

SYNOPSIS

ospfV3NetworkRange *subcommand options*

DESCRIPTION

The *ospfV3NetworkRange* command describes an individual set of routes. Network ranges are added into *ospfV3Router* lists using the *ospfV3Router addNetworkRange* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPFv3 testing with Ixia equipment. Refer to *ospfV3NetworkRange* for an overview.

Note that this command only applies to the OSPFv3 implementation; the commands related to OSPF all omit the **V3** letters in their name.

STANDARD OPTIONS

BBit

If true, BBit is used to determine whether the router is on the border.

EBit

If true, EBit is used to determine whether the router is on the AS boundary.

enable

If checked, enables the OSPFv3 network range grid.

entryAddress

(IPv6 address) The IPv6 address of the entry point emulated OSPFv3 router in the grid.

entryAddressMaskLength

(integer, range = 1 to 128) The length of the mask used with the IPv6 address of the entry point emulated OSPFv3 router in the grid. The default is 64.

entryLinkMetric

(integer) The metric of the link connecting the grid with the emulated OSPFv3 router.

entryPointColumn

(integer) The column where the entry point router is located in the OSPFv3 network range grid.

entryPointRow

(integer) The row where the entry point router is located in the OSPFv3 network range grid.

firstRouterId

(Dotted decimal 4-byte number, in IPv4 address format.) The identifier for the first emulated OSPFv3 router in the grid.

firstSubnetIpAddress

(IPv6 address) The IPv6 prefix address of the first subnet in the grid.

linkType

Choose one of:

Option	Value	Usage
ospfV3NetworkRangeLinkBroadcast		The OSPFv3 network range link type is Broadcast.
ospfV3NetworkRangeLinkPointToPoint		The OSPFv3 network range link Type is Point-to-Point.

numColumns

(integer) The number of columns in a grid.

numRows

(integer) The number of rows in a grid.

prefix

(integer, range = 1 to 128) The length of the mask used with the IPv6 addresses of the first subnet in the grid. The default is 64.

routerIdIncrementBy

(Dotted decimal 4-byte number, in IPv4 address format.) The identifier for the first emulated OSPFv3 router in the grid.

COMMANDS

The **ospfV3NetworkRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands.

ospfV3NetworkRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3NetworkRange** command.

ospfV3NetworkRange config *option value*

Modify the configuration options of the ospfV3NetworkRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3NetworkRange.

ospfV3NetworkRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Router*

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3Router

ospfV3Router — configures an OSPFv3 router

SYNOPSIS

ospfV3Router *subcommand options*

DESCRIPTION

The *ospfV3Router* command represents a simulated router. In addition to some identifying options, it holds three lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *ospfV3RouteRange* command.
- Interfaces — router interface, constructed in the *ospfV3Interface* command.
- LSA Groups — Link State Advertising Groups which will be associated with advertised routes, constructed in the *ospfV3UserLSAGroup* command and subsidiary commands.

Routers defined in this command are added to an *ospfV3Server* using the *ospfV3Server addRouter* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPFv3 testing with Ixia equipment. Refer to ospfV3Router for an overview of this command.

Note that this command only applies to the OSPFv3 implementation; the commands related to OSPF all omit the **V3** letters in their name.

STANDARD OPTIONS

disableAutoGenerate RouterLsa true / false

If ***true***, the router will **not** automatically generate a router LSA including all of the interfaces added with the *ospfV3Router addInterface* command. This should be turned off if you are building OSPF topologies with *ospfV3UserLsa* commands. *(default = false)*

disableAutoGenerate LinkLsa true / false

If ***true***, the router will **not** automatically generate a link LSA including all of the links added with the *ospfV3Router addInterface* command. This should be turned off if you are building OSPF topologies with *ospfV3UserLsa* commands. *(default = false)*

enable true / false

Enables the use of this router in the simulated OSPF network. *(default = false).*

enableDiscard LearnedLsas true / false

When this option is true, this simulated OSPF router (RID) will not learn any LSAs from the neighbor. *(default = false).*

enableSupportRfc5838

If true, enables Support RFC 5838.

learnedLsaCmd

Read-only. The ospfV3Router *getFirstLearnedLsa* and *getNextLearnedLsa* commands set the *learnedLsaCmd* (learned LSA command) to one of the following learned LSA types if there is any learned LSA; otherwise, *learnedLsaCmd* is set to NULL.

Option	Value	Usage
ospfV3LsaRouter		OSPFv3 Router LSA
ospfV3LsaNetwork		OSPFv3 Network LSA
ospfV3LsaInterAreaPrefix		OSPFv3 InterArea Prefix LSA
ospfV3LsaInterAreaRouter		OSPFv3 InterArea Router LSA
ospfV3LsaLink		OSPFv3 Link LSA
ospfV3LsaIntraAreaPrefix		OSPFv3 IntraArea Prefix LSA

maxNumLsaPerSecond

The maximum number of LSAs that will be generated each second for the router. *(default = 1,000)*

routerId

ID of the router, usually the lowest IP address on the router. *(default = 0.0.0.0)*

enableGracefulRestartHelperMode

If enabled, the router will act as restarting router's neighbors, which must cooperate in order for the restart to be graceful.

enableStrictLsaChecking

If enabled, the helping router continues to help the restarting router even if there is a topology change detected. Enabled only if 'enableGracefulRestartHelperMode' check box is selected.

enableSupportReasonSwRestart

If enabled, the router will support reason for SW restart. Enabled only if 'enableGracefulRestartHelperMode' check box is selected.

enableSupportReasonSwReloadOrUpgrade

If enabled, the router will support reason for SW reload or upgrade. Enabled only if 'enableGracefulRestartHelperMode' check box is selected.

enableSupportReasonSwitchToRedundantControlProcessor

If enabled, the router will support reason for switch to redunndant control processor. Enabled only if 'enableGracefulRestartHelperMode' check box is selected.

enableSupportReasonUnknown

If enabled, the router will support reason unknown. Enabled only if 'enableGracefulRestartHelperMode' check box is selected.

enableDesignatedRouter true / false

Enables the use of the designated router in the simulated OSPF network. *(default = false).*

COMMANDS

The **ospfV3Router** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3Router addInterface *interfaceLocalId*

Adds the router interface described in the *ospfV3Interface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *interfaceLocalId*. Specific errors are:

- The parameters in *ospfV3Interface* are invalid.
- A router with this *interfaceLocalId* exists already in the list.

ospfV3Router addNetworkRange *ospfV3NetworkRangeName*

Adds a ospfV3NetworkRange to the ospfV3NetworkRange list.

ospfV3Router addRouteRange *routeRangeLocalId*

Adds the route range described in the *ospfV3RouteRange* command to the list of route ranges associated with the router. The range's entry in the list is given an identifier of *routeRangeLocalId*. Specific errors are:

- The parameters in *ospfV3RouteRange* are invalid.
- A router with this *routeRangeLocalId* exists already in the list.

ospfV3Router addUserLsaGroup *userLsaGroupLocalId*

Adds the user LSA group described in the *ospfV3UserLsaGroup* command to the list of user LSAs associated with the router. The LSA's entry in the list is given an identifier of *userLsaGroupLocalId*. Specific errors are:

- The parameters in *ospfV3UserLsaGroup* are invalid.
- A router with this *userLsaGroupLocalId* exists already in the list.

ospfV3Router cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3Router** command.

ospfV3Router clearAllInterfaces

Deletes all of the router interfaces.

ospfV3Router clearAllLsaGroups

Deletes all of the user LSA groups.

ospfV3Router clearAllNetworkRange

Clears the ospfV3 NetworkRange list.

ospfV3Router clearAllRouteRanges

Deletes all of the route ranges.

ospfV3Router config *option value*

Modify the configuration options of the ospfV3Router. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3Router.

ospfV3Router delInterface *interfaceLocalId*

Deletes the router interface with an identifier of *interfaceLocalId*. Specific errors are:

- No router with this *interfaceLocalId* exists in the list.

ospfV3Router delNetworkRange *[ospfV3NetworkRangeName]*

Deletes a ospfV3NetworkRange from the ospfV3NetworkRange list. If no ospfV3NetworkRangeName is specified, it deletes the current one.

ospfV3Router delRouteRange *routeRangeLocalId*

Deletes the route range with an identifier of *routeRangeLocalId*. Specific errors are:

- No router with this *routeRangeLocalId* exists in the list.

ospfV3Router delUserLsaGroup *userLsaGroupLocalId*

Deletes the user LSA group with an identifier of *userLsaGroupLocalId*. Specific errors are:

- No router with this *userLsaGroupLocalId* exists in the list.

ospfV3Router getFirstInterface

Access the first interface in the list. The results may be accessed using the *ospfV3Interface* command. Specific errors are:

- There are no interfaces in the list.

ospfV3Router getFirstLearnedLsa

Gets the first learned LSA information record.

ospfV3Router getFirstNetworkRange

Gets the first ospfV3NetworkRange from ospfV3NetworkRange list and refresh the options.

ospfV3Router getFirstRouteRange

Access the first route rangein the list. The results may be accessed using the *ospfV3RouteRange* command. Specific errors are:

- There are no route ranges in the list.

ospfV3Router getFirstUserLsaGroup

Access the first user LSA groupin the list*.* The results may be accessed using the *ospfV3UserLsaGroup* command. Specific errors are:

- There are no user LSA groups in the list.

ospfV3Router getInterface *interfaceLocalId*

Accesses the interface's entry in the list with an identifier of *interfaceLocalId*. The router interface is accessed in the *ospfV3Interface* command. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

ospfV3Router getLearnedLsaList

Gets the list of learned information records populated by *protocolLearnedConfig* upon the completion of receiving the learned information messages.

ospfV3Router getNextInterface

Access the next interface in the list. The results may be accessed using the *ospfV3Interface* command. Specific errors are:

- ospfV3Router getFirstInterface has not been called.
- There is no more interfaces in the list.

ospfV3Router getNextLearnedLsa

Gets the succeeding learned LSA information record.

ospfV3Router getNextNetworkRange

Gets the next ospfV3NetworkRange from ospfV3NetworkRange list and refresh the options.

ospfV3Router getNextRouteRange

Access the next route range in the list*.* The results may be accessed using the *ospfV3RouteRange* command. Specific errors are:

- ospfV3Router getFirstRouteRange has not been called.
- There is no more route ranges in the list.

ospfV3Router getNextUserLsaGroup

Access the next user LSA group in the list*.* The results may be accessed using the *ospfV3UserLsaGroup* command. Specific errors are:

- ospfV3Router getFirstUserLsaGroup has not been called.
- There is no more user LSA groups in the list.

ospfV3Router getRouteRange *routeRangeLocalId*

Accesses the range's entry in the list with an identifier of *routeRangeLocalId*. The router range is accessed in the *ospfV3RouteRange* command. Specific errors are:

- A route range with this *routeRangeLocalId* does not exist in the list.

ospfV3Router getUserLsaGroup *userLsaGroupLocalId*

Accesses the user LSA group's entry in the list with an identifier of *userLsaGroupLocalId*. The group is accessed in the ospfV3UserLsaGroup command. Specific errors are:

- A router with this *userLsaGroupLocalId* does not exist in the list.

ospfV3Router requestLearnedLsa

Sends a request to the protocol server for the learned information and registers a callback function for receiving the learned information from the protocol server. Upon receiving the learned information the callback checks whether the complete learned information is received and sets a flag accordingly.

ospfV3Router setDefault

Sets default values for all configuration options.

ospfV3Router setInterface *interfaceLocalId*

Sets the values for the interface's entry in the list with an identifier of *interfaceLocalId* based on changes made through the *ospfV3Interface* command. This command can be used to change a running configuration and must be followed by an *ospfV3Server write* command in order to send these changes to the protocol server. Specific errors are:

- An interface with this *interfaceLocalId* does not exist in the list.

spfV3Router setNetworkRange *[ospfV3NetworkRangeName]*

Edit on the fly "ospfV3NetworkRangeName." If there is no ospfV3NetworkRangeName specified, then the current one will be modified.

ospfV3Router setRouteRange *routeRangeLocalId*

Sets the values for the route range's entry in the list with an identifier of *routeRangeLocalId* based on changes made through the *ospfV3RouteRange* command. This command can be used to change a running configuration and must be followed by an *ospfV3Server write* command in order to send these changes to the protocol server. Specific errors are:

- A router range with this *routeRangeLocalId* does not exist in the list.

ospfV3Router setUserLsaGroup *userLsaGroupLocalId*

Sets the values for the user LSA group's entry in the list with an identifier of *userLsaGroupLocalId* based on changes made through the *ospfV3UserLsaGroup* command. This command can be used to change a running configuration and must be followed by an *ospfV3Server write* command in order to send these changes to the protocol server. Specific errors are:

- A user LSA group with this *userLsaGroupLocalId* does not exist in the list.

EXAMPLES

```
# To add an ospfV3NetworkRange under an ospfV3Router :

ospfV3Server select $ chId $ cardId $ portId
```

```
ospfV3Server getFirstRouter
<configure parameters of ospfV3NetworkRange object , say
ospfV3NetworkRange config –enable true e.t.c)
ospfV3NetworkRange config -enable false
ospfV3NetworkRange config -entryPointRow 1
ospfV3NetworkRange config -entryPointColumn 1
ospfV3NetworkRange config -numRows 1
ospfV3NetworkRange config -numColumns 1
ospfV3NetworkRange config -prefix 64
ospfV3NetworkRange config -entryAddressMaskLength 64
ospfV3NetworkRange config -linkType 0
ospfV3NetworkRange config -entryLinkMetric 1
ospfV3NetworkRange config -BBit 0
ospfV3NetworkRange config -EBit 0
ospfV3NetworkRange config -routerIdIncrementBy
“1.0.0.0”
ospfV3NetworkRange config -firstRouterId
“0.0.0.0”
ospfV3NetworkRange config -firstSubnetIpAddress
“EE:0:0:0:0:0:0:2”
ospfV3NetworkRange config -entryAddress
“EE:0:0:0:0:0:0:1”
ospfV3Router addNetworkRange n1
ospfV3Server write
# To set an existing ospfV3NetworkRange :
ospfV3Server select $ chId $ cardId $ portId
ospfV3Server getFirstRouter
<change the parameter values of an ospfV3NetworkRange, say the
first in the list of networkRanges)
ospfV3Router getFirstNetworkRange
ospfV3NetworkRange config -enable true
ospfV3NetworkRange config -entryPointRow 2
ospfV3NetworkRange config -entryPointColumn 2
ospfV3NetworkRange config -routerIdIncrementBy
```

```
"2.3.4.5"

ospfV3NetworkRange config -firstSubnetIpAddress

"DD:0:0:0:0:0:0:2"

ospfV3Router setNetworkRange

ospfV3Server write

#To get the first ospfV3NetworkRange

ospfV3Server select $ chId $ cardId $ portId

ospfV3Server getFirstRouter

ospfV3Router getFirstNetworkRange

#To get the next ospfV3NetworkRange

ospfV3Server select $ chId $ cardId $ portId

ospfV3Server getFirstRouter

ospfV3Router getFirstNetworkRange

ospfV3Router getNextNetworkRange

#To delete an existing ospfV3NetworkRange

ospfV3Server select $ chId $ cardId $ portId

ospfV3Server getFirstRouter

ospfV3Router getFirstNetworkRange

ospfV3Router getNextNetworkRange

ospfV3Router delNetworkRange // deletes the current

ospfV3NetworkRange

ospfV3server write

# To delete all existing ospfV3NetworkRanges from under a router

ospfV3Server select $ chId $ cardId $ portId

ospfV3Server getFirstRouter

ospfV3Router clearAllNetworkRanges

ospfV3Server write
```

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3RouteRange

ospfV3RouteRange — sets up the parameters associated with an OSPF router range.

SYNOPSIS

ospfV3RouteRange *subcommand options*

DESCRIPTION

The *ospfV3RouteRange* command describes an individual set of routes. Route ranges are added into *ospfV3Router* lists using the *ospfV3Router addRouteRange* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPFv3 testing with Ixia equipment. Refer to *ospfV3RouteRange* for an overview.

Note that this command only applies to the OSPFv3 implementation; the commands related to OSPF all omit the **V3** letters in their name.

STANDARD OPTIONS

enable true / false

Enables the use of this route range for the simulated router. *(default = false)*

iterationStep

The increment used to generate multiple addresses. *(default = 1)*

maskWidth

The number of bits in the address to be advertised. *(default = 64)*

metric

The cost metric associated with the route. *(default = 0)*

networkIpAddress

The IP address of the routes to be advertised, in IPv6 format. *(default = 0:0:0:0:0:0:0:0)*

numRoutes

The number of routes to be advertised. *(default = 1)*

ipType

Signifies the IP type. It can be address type IPv4 or IPv6.

addressFamily

Signifies the address family. It can be either unicast or multicast.

routeOrigin

Whether the route originated within the area or externally. One of:

Option	Value	Usage
ospfV3RouteOriginAnotherArea	0	*(default)* within the area
ospfV3RouteOriginExternalType1	1	from outside the area
ospfV3RouteOriginExternalType2	2	from outside the area, but with metrics which are larger than any internal metric

COMMANDS

The **ospfV3RouteRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands.

ospfV3RouteRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3RouteRange** command.

ospfV3RouteRange config *option value*

Modify the configuration options of the ospfV3RouteRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3RouteRange.

ospfV3RouteRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ospfV3Server*.

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3RouteRange*, *ospfV3Server*, *ospfV3UserLsaGroup*

NAME - ospfV3Server

ospfV3Server — accesses the OSPFv3 component of the protocol server for a particular port.

SYNOPSIS

ospfV3Server *subcommand options*

DESCRIPTION

The *ospfV3Server* command is necessary in order to access the OSPFv3 protocol server for a particular port. The *select* subcommand **must** be used before all other OSPFv3 commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPFv3 testing with Ixia equipment. Refer to *ospfV3Server* for an overview.

Note that this command only applies to the OSPFv3 implementation; the commands related to OSPF all omit the **V3** letters in their name.

COMMANDS

The **ospfV3Server** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3Server addRouter *routerLocalId*

Adds the OSPF router described in the *ospfV3Router* command to the list of routers associated with the port. The router's entry in the list is given an identifier of *routerLocalId*. Specific errors are:

- ospfV3Server select has not been called.
- The router parameters in *ospfV3Router* are invalid.
- A router with this *routerLocalId* exists already in the list.

ospfV3Server cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3Server** command.

ospfV3Server clearAllRouters

Deletes all the OSPF routers in the list. Specific errors are:

- ospfV3Server select has not been called.

ospfV3Server config *option value*

Modify the configuration options of the ospfV3Server. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ospfV3Server.

ospfV3Server delRouter *routerLocalId*

Deletes the OSPF router described that has an identifier of *routerLocalId*. Specific errors are:

- ospfV3Server select has not been called.
- There is no router with this *routerLocalId* in the list.

ospfV3Server getFirstRouter

Access the first OSPFv3 router in the list. The results may be accessed using the *ospfV3Router* command. Specific errors are:

- ospfV3Server select has not been called.
- There are no routers in the list.

ospfV3Server getNextRouter

Access the next OSPFv3 router in the list. The results may be accessed using the *ospfV3Router* command. Specific errors are:

- ospfV3Server select has not been called.
- ospfV3Server getFirstRouter has not been called.
- There are no more routers in the list.

ospfV3Server getRouter *routerLocalId*

Access the OSPFv3 router with an identifier of *routerLocalId.* The results may be accessed using the *ospfV3Router* command. Specific errors are:

- ospfV3Server select has not been called.
- There is no router with this *routerLocalId* in the list.

ospfV3Server select *chasID cardID portID*

Accesses the OSPFv3 component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The OSPFv3 protocol package has not been installed.
- Invalid port specified.

ospfV3Server setRouter *routerLocalId*

Sets the values for the router's entry in the list with an identifier of *routerLocalId* based on changes made through the *ospfV3Router* command. This command should be used to change a running configuration and must be followed by an *ospfV3Server write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *routerLocalId* does not exist in the list.

ospfV3Server write

Sends any changes made with *ospfV3Router setInterface, ospfV3Router setRouteRange, ospfV3Router setUserLsaGroup* or *ospfV3Server setRouter* to the protocol server for immediate application. This command **must** be used after those mentioned above in order for their changes to have an effect.

EXAMPLES

```
package req IxTclHal
set hostname localhost
set username user
# Check if we're running on UNIX - connect to the TCL Server
# which must be running on the chassis
if [isUNIX] {
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
set card 67
set port 1
set streamId 1
set portList [list [list $chassis $card $port]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
# Reset port to factory defaults
```

```
port setFactoryDefaults $chassis $card $port
# Set up an interface entry for a single IPv6 address
interfaceTable select $chassis $card $port
interfaceTable clearAllInterfaces
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description {1 - 67:01 -
EFC1:0:0:0:0:0:0:0/64}
interfaceEntry config -macAddress {00 00 0A 54 65 35}
interfaceTable addInterface
ixWritePortsToHardware portList
# Time to set up the OSPFv3 Server
ospfV3Server select $chassis $card $port
ospfV3Server clearAllRouters
# Start by defining an interface for a router
ospfV3Interface setDefault
ospfV3Interface config -enable true
ospfV3Interface config -areaId 8
ospfV3Interface config -options [expr
$::ospfV3InterfaceOptionEBit | \
$::ospfV3InterfaceOptionMCBit| \
$::ospfV3InterfaceOptionRBit]
ospfV3Interface config -helloInterval 10
ospfV3Interface config -deadInterval 40
ospfV3Interface config -protocolInterfaceDescription \
{1 - 67:01 -
EFC1:0:0:0:0:0:0:0/64}
ospfV3Interface config -interfaceId 0
# Add the interface to the router
ospfV3Router addInterface interface1
# Now define a route range for the router
ospfV3RouteRange setDefault
```

```
ospfV3RouteRange config -enable true
ospfV3RouteRange config -routeOrigin
ospfV3RouteOriginExternalType1
ospfV3RouteRange config -metric 4
ospfV3RouteRange config -numRoutes 1
ospfV3RouteRange config -maskWidth 128
ospfV3RouteRange config -networkIpAddress {785:0:0:0:0:0:0:0}
# Add the route range to the router
ospfV3Router addRouteRange routeRange1
# Create one each of the different LSA types
# Create a Router LSA - start with defining an interface
ospfV3LsaRouterInterface setDefault
ospfV3LsaRouterInterface config -interfaceId {10.0.0.0}
ospfV3LsaRouterInterface config -neighborInterfaceId {2.0.0.0}
ospfV3LsaRouterInterface config -neighborId {8.0.0.0}
ospfV3LsaRouterInterface config -type
ospfV3LsaRouterInterfaceVirtual
ospfV3LsaRouterInterface config -metric 9
# Add the interface to the Router LSA
ospfV3LsaRouter addInterface
# And another
ospfV3LsaRouterInterface setDefault
ospfV3LsaRouterInterface config -interfaceId {22.0.0.0}
ospfV3LsaRouterInterface config -neighborInterfaceId {3.0.0.0}
ospfV3LsaRouterInterface config -neighborId {9.0.0.0}
ospfV3LsaRouterInterface config -type
ospfV3LsaRouterInterfaceTransit
ospfV3LsaRouterInterface config -metric 45
ospfV3LsaRouter addInterface
# Now define the remaining bits of the Router LSA
ospfV3LsaRouter setDefault
ospfV3LsaRouter config -enable true
ospfV3LsaRouter config -linkStateId {10.0.0.0}
ospfV3LsaRouter config -advertisingRouterId {0.0.0.90}
```

```
ospfV3LsaRouter config -options [expr $::ospfV3LsaOptionNBit
| \
$::ospfV3LsaOptionRBit | \
$::ospfV3LsaOptionDCBit]
ospfV3LsaRouter config -enableVBit false
ospfV3LsaRouter config -enableEBit true
ospfV3LsaRouter config -enableBBit true
ospfV3LsaRouter config -enableWBit false
# Add the Router LSA to the user LSA group
ospfV3UserLsaGroup addUserLsa userLsa1 ospfV3LsaRouter
# Now create a Network LSA
ospfV3LsaNetwork setDefault
ospfV3LsaNetwork config -enable true
ospfV3LsaNetwork config -linkStateId {10.0.0.2}
ospfV3LsaNetwork config -advertisingRouterId {2.0.0.92}
ospfV3LsaNetwork config -options $::ospfV3LsaOptionDCBit
ospfV3LsaNetwork config -neighborRouterIdList \
{ 9.0.0.0 13.0.0.1 17.0.0.2 21.0.0.3
\
25.0.0.4 29.0.0.5 33.0.0.6 37.0.0.7
\
41.0.0.8 45.0.0.9}
# Add the Network LSA to the user LSA group
ospfV3UserLsaGroup addUserLsa userLsa2 ospfV3LsaNetwork
# Create an Inter Area Prefix LSA
ospfV3LsaInterAreaPrefix setDefault
ospfV3LsaInterAreaPrefix config -enable true
ospfV3LsaInterAreaPrefix config -linkStateId {10.0.0.4}
ospfV3LsaInterAreaPrefix config -advertisingRouterId {4.0.0.94}
ospfV3LsaInterAreaPrefix config -numLsaToGenerate 1
ospfV3LsaInterAreaPrefix config -incrementLinkStateIdBy {8.0.0.0}
ospfV3LsaInterAreaPrefix config -prefixLength 45
ospfV3LsaInterAreaPrefix config -prefixOptions [expr
$::ospfV3PrefixOptionMCBit | \
$::ospfV3PrefixOptionPBit]
ospfV3LsaInterAreaPrefix config -incrementPrefixBy 78
ospfV3LsaInterAreaPrefix config -prefixAddress
{100:0:0:0:0:0:0:0}
# Add the Inter Area Prefix LSA to the User LSA group
ospfV3UserLsaGroup addUserLsa userLsa3
ospfV3LsaInterAreaPrefix
# Create an Inter Area Router LSA
ospfV3LsaInterAreaRouter setDefault
ospfV3LsaInterAreaRouter config -enable true
ospfV3LsaInterAreaRouter config -linkStateId {10.0.0.6}
ospfV3LsaInterAreaRouter config -advertisingRouterId {6.0.0.96}
ospfV3LsaInterAreaRouter config -numLsaToGenerate 1
ospfV3LsaInterAreaRouter config -incrementLinkStateIdBy {8.0.0.0}
ospfV3LsaInterAreaRouter config -destinationRouterId {7.6.0.0}
ospfV3LsaInterAreaRouter config -incrementDestRouterIdBy
{0.9.0.1}
ospfV3LsaInterAreaRouter config -options [expr
$::ospfV3LsaOptionV6Bit | \
$::ospfV3LsaOptionNBit | \
$::ospfV3LsaOptionRBit]
ospfV3LsaInterAreaRouter config -metric 78
# Add the Inter Area Router LSA to the user group
ospfV3UserLsaGroup addUserLsa userLsa4 ospfV3LsaInterAreaRouter
# Create an AS External LSA
ospfV3LsaAsExternal setDefault
ospfV3LsaAsExternal config -enable true
ospfV3LsaAsExternal config -linkStateId {10.0.0.8}
ospfV3LsaAsExternal config -advertisingRouterId {8.0.0.98}
ospfV3LsaAsExternal config -numLsaToGenerate 78
ospfV3LsaAsExternal config -incrementLinkStateIdBy {8.0.0.0}
ospfV3LsaAsExternal config -incrementPrefixBy 25
ospfV3LsaAsExternal config -prefixLength 128
ospfV3LsaAsExternal config -prefixOptions [expr
```

```
$::ospfV3PrefixOptionLABit | \
$::ospfV3PrefixOptionPBit]
ospfV3LsaAsExternal config -prefixAddress
{545:0:0:0:0:0:0:0}
ospfV3LsaAsExternal config -metric 10
ospfV3LsaAsExternal config -enableEBit false
ospfV3LsaAsExternal config -enableTBit false
ospfV3LsaAsExternal config -enableFBit true
ospfV3LsaAsExternal config -referencedLinkStateId {9.3.0.0}
ospfV3LsaAsExternal config -externalRouteTag {7.8.0.0}
ospfV3LsaAsExternal config -referencedType 1
ospfV3LsaAsExternal config -forwardingAddress
{245:0:0:0:0:0:0:0}
# Add the AS External LSA to the user LSA group
ospfV3UserLsaGroup addUserLsa userLsa5 ospfV3LsaAsExternal
# Create a Link LSA, start by adding two address prefixes
# First prefix
ospfV3IpV6Prefix setDefault
ospfV3IpV6Prefix config -incrementBy 8
ospfV3IpV6Prefix config -length 19
ospfV3IpV6Prefix config -options [expr
$::ospfV3PrefixOptionMCBit | \
$::ospfV3PrefixOptionPBit]
ospfV3IpV6Prefix config -address {2352:3:0:0:0:0:0:0}
# Add the prefix to the Link LSA
ospfV3LsaLink addPrefix
# Second prefix
ospfV3IpV6Prefix setDefault
ospfV3IpV6Prefix config -incrementBy 9
ospfV3IpV6Prefix config -length 7
ospfV3IpV6Prefix config -options
$::ospfV3PrefixOptionNUBit
ospfV3IpV6Prefix config -address {35:0:0:0:0:0:0:0}
# Add the prefix to the Link LSA
```

```
ospfV3LsaLink addPrefix

# Now the rest of the Link LSA contents

ospfV3LsaLink setDefault

ospfV3LsaLink config -enable true

ospfV3LsaLink config -linkStateId {10.0.0.10}

ospfV3LsaLink config -advertisingRouterId

{10.0.0.100}

ospfV3LsaLink config -numLsaToGenerate 65

ospfV3LsaLink config -incrementLinkStateIdBy {0.8.0.0}

ospfV3LsaLink config -options [expr

$::ospfV3LsaOptionRBit | \

$::ospfV3LsaOptionDCBit]

ospfV3LsaLink config -linkLocalAddress

{0:77:0:0:0:0:0:0}

ospfV3LsaLink config -priority 8

# Add the Link LSA to the user LSA group

ospfV3UserLsaGroup addUserLsa userLsa6 ospfV3LsaLink

# Add an Intra Area Prefix LSA

# Start with defining two address prefixes

ospfV3IpV6Prefix setDefault

ospfV3IpV6Prefix config -incrementBy 9

ospfV3IpV6Prefix config -length 12

ospfV3IpV6Prefix config -options [expr

$::ospfV3PrefixOptionMCBit | \

$::ospfV3PrefixOptionPBit]

ospfV3IpV6Prefix config -address {55:0:0:0:0:0:0:0}

# Add the prefix to the LSA

ospfV3LsaIntraAreaPrefix addPrefix

ospfV3IpV6Prefix setDefault

ospfV3IpV6Prefix config -incrementBy 78

ospfV3IpV6Prefix config -length 25

ospfV3IpV6Prefix config -options 0

ospfV3IpV6Prefix config -address {5668:0:0:0:0:0:0:0}

# Add the prefix to the LSA
```

```
ospfV3LsaIntraAreaPrefix addPrefix
# The rest of the options for the Intra Area Prefix LSA
ospfV3LsaIntraAreaPrefix setDefault
ospfV3LsaIntraAreaPrefix config -enable true
ospfV3LsaIntraAreaPrefix config -linkStateId {10.0.0.12}
ospfV3LsaIntraAreaPrefix config -advertisingRouterId
{12.0.0.102}
ospfV3LsaIntraAreaPrefix config -numLsaToGenerate 1
ospfV3LsaIntraAreaPrefix config -incrementLinkStateIdBy {0.0.0.0}
ospfV3LsaIntraAreaPrefix config -referencedType 1
ospfV3LsaIntraAreaPrefix config -referencedLinkStateId {0.0.0.0}
ospfV3LsaIntraAreaPrefix config -referencedRouterId {0.0.0.0}
# Add the Intra Area Prefix LSA to the user LSA group
ospfV3UserLsaGroup addUserLsa userLsa7 ospfV3LsaIntraAreaPrefix
# Now finalize the details about the user LSA group
ospfV3UserLsaGroup setDefault
ospfV3UserLsaGroup config -enable true
ospfV3UserLsaGroup config -areaId 10
ospfV3UserLsaGroup config -description {}
# And add the group to the router
ospfV3Router addUserLsaGroup userLsaGroup1
# Finalize the OSPF router details
ospfV3Router setDefault
ospfV3Router config -routerId {11.2.0.0}
ospfV3Router config -enable true
ospfV3Router config -autoGenerateRouterLsa 0
ospfV3Router config -enableDiscardLearnedLsas false
# And add the router to the server
ospfV3Server addRouter router1
# Make sure to enable the protocol with the Protocol Server
protocolServer setDefault
protocolServer config -enableOspfV3Service true
protocolServer set $chassis $card $port
ixWritePortsToHardware portList
```

```
ixCheckLinkState portList
# Now start the OSPFv3 simulation
ixStartOspfV3
# While the simulation is running, all facets of the simulation
may
# be modified, using the following procedures
# Editing “Router” on the fly, using getFirst/getNext.
ospfV3Server select $chassis $card $port
ospfV3Server getFirstRouter
ospfV3Router config -routerId {20.2.0.0}
ospfV3Router config -enable true
ospfV3Server setRouter
ospfV3Server write
# Editing “Router” on the fly, using “get” by name.
ospfV3Server select $chassis $card $port
ospfV3Server getRouter router1
ospfV3Router config -routerId {30.2.0.0}
ospfV3Router config -enable false
ospfV3Server setRouter router1
ospfV3Server write
# Editing “Interface”, “RouteRange”, “userLsaGroup” on the fly,
# using getFirst/getNext.
ospfV3Server select $chassis $card $port
ospfV3Server getFirstRouter
ospfV3Router getFirstInterface
ospfV3Interface config -enable true
ospfV3Interface config -areaId 10
ospfV3Router setInterface
ospfV3Router getFirstRouteRange
ospfV3RouteRange config -enable true
ospfV3RouteRange config -routeOrigin 1
ospfV3RouteRange config -metric 90
ospfV3RouteRange config -numRoutes 100
ospfV3Router setRouteRange
```

```
ospfV3Router getFirstUserLsaGroup
ospfV3UserLsaGroup config -enable true
ospfV3UserLsaGroup config -areaId 450
ospfV3Router setUserLsaGroup
ospfV3Server write
# Editing “Interface”, “RouteRange”, “userLsaGroup” on the fly,
# using “get” by name.
ospfV3Server select $chassis $card $port
ospfV3Server getRouter router1
ospfV3Router getInterface interface1
ospfV3Interface config -enable true
ospfV3Interface config -areaId 80
ospfV3Router setInterface interface1
ospfV3Router getRouteRange routeRange1
ospfV3RouteRange config -enable true
ospfV3RouteRange config -routeOrigin 1
ospfV3RouteRange config -metric 20
ospfV3RouteRange config -numRoutes 800
ospfV3Router setRouteRange routeRange1
ospfV3Router getUserLsaGroup userLsaGroup1
ospfV3UserLsaGroup config -enable true
ospfV3UserLsaGroup config -areaId 500
ospfV3Router setUserLsaGroup userLsaGroup1
ospfV3Server write
# Editing “UserLsa” on the fly, using getFirst/getNext.
ospfV3Server select $chassis $card $port
ospfV3Server getFirstRouter
ospfV3Router getFirstUserLsaGroup
ospfV3UserLsaGroup getFirstUserLsa
set cmd [ospfV3UserLsaGroup getNextUserLsa]
showCmd $cmd
$cmd config -enable 0
ospfV3UserLsaGroup setUserLsa
ospfV3Server write
```

```
# Editing “UserLsa” on fly, using “get” by name.
ospfV3Server select $chassis $card $port
ospfV3Server getRouter router1
ospfV3Router getUserLsaGroup userLsaGroup1
set cmd [ospfV3UserLsaGroup getUserLsa userLsa5]
showCmd $cmd
if { [$cmd cget -type] == $::ospfV3LsaAsExternal} {
ospfV3LsaAsExternal config -prefixAddress
{900:0:0:0:0:0:0:0}
ospfV3LsaAsExternal config -metric 45
ospfV3LsaAsExternal config -enable 0
ospfV3UserLsaGroup setUserLsa userLsa5
ospfV3Server write
}
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3UserLsaGroup*

NAME - ospfV3UserLsaGroup

ospfV3UserLsaGroup — configures a User LSA group for use in OSPFv3.

SYNOPSIS

ospfV3UserLsaGroup *subcommand options*

DESCRIPTION

The *ospfV3UserLSAGroup* command describes a list of LSAs to be associated with advertised routes. The list consists of elements constructed through the use of the a number of commands, one command for each type of LSA:

- *ospfV3LsaAsExternal*
- *ospfV3LsaInterAreaPrefix*
- *ospfV3LsaInterAreaRouter*
- *ospfV3LsaIntraAreaPrefix*
- *ospfV3LsaLink*
- *ospfV3LsaNetwork*
- *ospfV3LsaRouter*

An LSA is added to a user LSA group by configuring the LSA with the appropriate command from the list above and then adding it to the group with *ospfV3UserLsaGroup addUserLsa id type*, where *type* indicates which of the LSAs to use. An LSA may be retrieved from a user LSA group through the use of *getUserLsa / getFirstUserLsa / getNextUserLsa.* These commands return the **name** of the command that was used to configure the LSA. This is typically used in the following sequence of commands:

```
 set lsaCmd [ospfV3LsaGroup getFirstUserLsa]
```

```
$lsaCmd config -enable 0
```

Each of the LSA commands also has a *type*option which uniquely identifies the type of the LSA.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on OSPFv3 testing with Ixia equipment. Refer to *ospfV3UserLSAGroup* for an overview of this command.

STANDARD OPTIONS

areaId

The area ID for the LSA group. *(default = 0)*

description

A commentary description for the user LSA group. *(default = "")*

enable true / false

Enables the use of this router in the simulated OSPF network. *(default = false)*

COMMANDS

The **ospfV3UserLsaGroup** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ospfV3UserLsaGroup **addUserLsa** *userLsaLocalId lsaType*

Adds the user LSA described in the *ospfV3UserLsa* command and of type *lsaType* to the list. The LSA's entry in the list is given an identifier of *userLsaLocalId*. The choices of *lsaType* are:

Option	Value	Usage
ospfV3LsaRouter	0	LSA is built using the *ospfV3LsaRouter* command.
ospfV3LsaNetwork	1	LSA is built using the *ospfV3LsaNetwork* command.
ospfV3LsaInterAreaPrefix	2	LSA is built using the *ospfV3LsaInterAreaPrefix* command.
ospfV3LsaInterAreaRouter	3	LSA is built using the *ospfV3LsaInterAreaRouter* command.
ospfV3LsaAsExternal	4	LSA is built using the *ospfV3LsaAsExternal* command.
ospfV3LsaLink	5	LSA is built using the *ospfV3LsaLink* command.
ospfV3LsaIntraAreaPrefix	6	LSA is built using the *ospfV3LsaIntraAreaPrefix* command.

Specific errors are:

- The LSA type in *ospfV3UserLsa* is not valid
- The parameters in *ospfV3UserLsa* are invalid
- A LSA with this *userLsaLocalId* exists already in the list

ospfV3UserLsaGroup cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ospfV3UserLsaGroup** command.

ospfV3UserLsaGroup clearAllUserLsas

Deletes all of the user LSAs.

ospfV3UserLsaGroup config *option value*

Modify the configuration options of the *ospfV3UserLsaGroup*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *ospfV3UserLsaGroup*.

ospfV3UserLsaGroup delUserLsa *userLsaLocalId*

Deletes the user LSA with an identifier of *userLsaLocalId*. Specific errors are:

- No LSA with this *userLsaLocalId* exists in the list.

ospfV3UserLsaGroup getFirstUserLsa

Access the first user LSA in the list. The results of the command is the **name**of the command used to make the LSA. This command may be symbolically used to view/modify the LSAs contents. Each LSA contains a *type* option that uniquely identifies the LSA's type. Specific errors are:

- There are no LSAs in the list.

ospfV3UserLsaGroup getNextUserLsa

Access the next user LSA in the list. The results of the command is the **name** of the command used to make the LSA. This command may be symbolically used to view/modify the LSAs contents. Each LSA contains a *type* option that uniquely identifies the LSA's type. Specific errors are:

- ospfV3UserLsaGroup getFirstUserLsa has not been called.
- There is no more LSAs in the list.

ospfV3UserLsaGroup getUserLsa *userLsaLocalId*

Accesses the LSA's entry in the list with an identifier of *userLsaLocalId*. The user LSA is accessed in the *ospfV3UserLsa* command. Specific errors are:

- No LSA with this *userLsaLocalId* exists in the list.

ospfV3UserLsaGroup setDefault

Sets default values for all configuration options.

ospfV3UserLsaGroup setUserLsa *userLsaLocalId*

Allows the configuration values for a User LSA with an identifier of *userLsaLocalId* to be overwritten on the fly.

EXAMPLES

See examples under *ospfV3Server*.

SEE ALSO

ospfV3Interface, *ospfV3IpV6Prefix*, *ospfV3LsaAsExternal*, *ospfV3LsaInterAreaPrefix*, *ospfV3LsaInterAreaRouter*, *ospfV3LsaIntraAreaPrefix*, *ospfV3LsaLink*, *ospfV3LsaNetwork*, *ospfV3LsaRouter*, *ospfV3LsaRouterInterface*, *ospfV3Router*, *ospfV3RouteRange*, *ospfV3Server*

NAME - pimsmDataMdtRange

pimsmDataMdtRange — sets up the parameters associated with a PIM-SM Data MDT range.

SYNOPSIS

pimsmDataMdtRange *subcommand options*

DESCRIPTION

The *pimsmDataMdtRange* command describes a range of Data Multicast Distribution Trees (MDTs) under a PIM-SM interface. Ranges are added into *pimsmInterface* lists using the *pimsmInterface* *addDataMdtRange* command. Refer to *PIM-SM*67 for an overview of this command.

STANDARD OPTIONS

activationInterval

The time interval for the switchover from the Default MDT to the Data MDT, in seconds. *(default = 60)*

dataMdtAddress

The first IPv4 Data MDT multicast group address in the range of group addresses. *(default = 230.0.0.0)*

dataMdtAddressCount

The number of Data MDT addresses, starting with *dataMdtAddress,* to be included in the Data MDT range. *(default = 1)*

enable true / false

Enables the use of this Data MDT range on the simulated interface. *(default = false)*

enableDiscardLearnedStates true / false

pimsmMdtLearnedJoinStates will be available if this flag is disabled. If this flag is enabled, *pimsmLearnedJoinStates* will be available. *(default = true)*

enablePacking true / false

Enables the packing of multiple addresses into a single packet. *(default = true)*

groupAddress

The first IPv4 multicast group address in the range of group addresses. *(default = 225.0.0.0)* **NOTE**: This must be a valid IPv4 multicast address.

groupAddressCount

The number of group addresses, starting with *groupAddress* to be included in the range. *(default = 1)*

sourceAddress

The first IPv4 source address to be included in the range of source addresses*. (default = 0.0.0.1)* **NOTE**: This must be a valid IPv4 unicast (non-loopback) address*.*

sourceAddressCount

Used with the *sourceAddress* to define the range of source addresses. *(default = 1)*

sourceGroupMapping

Sets the type of mapping that occurs when routes are advertised. One of:

Option	Value	Usage
pimsmMappingFullyMeshed	0	*(default)* All sources to all groups.
pimsmMappingOneToOne	1	One source to one group.

COMMANDS

The **pimsmDataMdtRange** cnommand is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmDataMdtRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *pimsmDataMdtRange* command.

pimsmDataMdtRange configure *option value*

Modify the configuration options of the *pimsmDataMdtRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *pimsmDataMdtRange*.

pimsmDataMdtRange getMdtLearnedJoinState

This subcommand should be called after *requestMdtLearnedJoinState*. It must be called until it returns TCL_OK, usually with a wait between calls. The actual join state may be retrieved by using the *pimsmMdtLearnedJoinState* command.

pimsmDataMdtRange requestMdtLearnedJoinState

Requests that the learned join states for this Data MDT range be retrieved from the protocol server. The *getMdtLearnedJoinState* subcommand must be called after this subcommand to determine when the complete list of routes has been retrieved.

pimsmDataMdtRange setDefault

Sets default values for all configuration options.

EXAMPLES

```
package require IxTclHal

set hostname loopback

ixConnectToChassis $hostname
```

```
set chId [chassis cget -id]
set cardId 1
set portId 1
protocolServer config enablePimsmService true
protocolServer set $chId $cardId $portId
pimsmServer select $chassis $card $port
pimsmServer clearAllRouters
pimsmJoinPrune setDefault
pimsmJoinPrune config -enable true
pimsmJoinPrune config -rangeType 1
pimsmJoinPrune config -sourceGroupMapping 0
pimsmJoinPrune config -groupAddress
“225.0.0.0”
pimsmJoinPrune config -groupMaskWidth 32
pimsmJoinPrune config -groupAddressCount 1
pimsmJoinPrune config -sourceAddress
“0.0.0.1”
pimsmJoinPrune config -sourceMaskWidth 32
pimsmJoinPrune config -sourceAddressCount 1
pimsmJoinPrune config -pruneSourceAddress
“0.0.0.0”
pimsmJoinPrune config -pruneSourceMaskWidth 32
pimsmJoinPrune config -pruneSourceCount 0
pimsmJoinPrune config -enablePacking true
pimsmJoinPrune config -rpAddress
“0.0.0.0”
pimsmJoinPrune config -enableFlap false
pimsmJoinPrune config -flapInterval 60
pimsmJoinPrune config -switchoverInterval 5
pimsmJoinPrune config -registerStopTriggerCount 10
pimsmInterface addJoinPrune joinPrune1
pimsmSource setDefault
pimsmSource config -enable true
pimsmSource config -sourceGroupMapping 0
pimsmSource config -groupAddress
"225.0.0.0"
pimsmSource config -groupAddressCount 1
pimsmSource config -sourceAddress
"0.0.0.1"
pimsmSource config -sourceAddressCount 1
pimsmSource config -enableDiscardJoinStates true
pimsmSource config -rpAddress
"0.0.0.0"
pimsmSource config -txIterationGap 60000
pimsmSource config -udpSourcePort 3000
pimsmSource config -udpDestinationPort 3000
pimsmSource config -enableSendNullRegAtBeginning
false
pimsmInterface addSource source1
pimsmDataMdtRange setDefault
pimsmDataMdtRange config -enable true
pimsmDataMdtRange config -groupAddress
"225.0.0.0"
pimsmDataMdtRange config -groupAddressCount 1
pimsmDataMdtRange config -sourceAddress
"0.0.0.1"
pimsmDataMdtRange config -sourceAddressCount 1
pimsmDataMdtRange config -sourceGroupMapping 0
pimsmDataMdtRange config -dataMdtAddress
"240.0.0.0"
pimsmDataMdtRange config -dataMdtAddressCount 1
pimsmDataMdtRange config -enablePacking false
pimsmDataMdtRange config -activationInterval 60
pimsmDataMdtRange config-enableDiscardLearnedStates true
pimsmInterface addDataMdtRange dataMdtRange1
pimsmInterface setDefault
pimsmInterface config -enable true
pimsmInterface config -helloInterval 30
pimsmInterface config -helloHoldTime 105
pimsmInterface config -protocolInterfaceDescription
“GRE - 100:01 - 1”
pimsmInterface config -enablePruneDelay true
pimsmInterface config -pruneDelay 500
pimsmInterface config -overrideInterval 2500
pimsmInterface config -enableSendBidirectionalOption
false
pimsmInterface config -generationIdMode 2
pimsmInterface config -enableSendGenerationId
true
pimsmInterface config -enablePruneDelayTBit
false
pimsmInterface config -enableAutoPickNeighbor
true
pimsmInterface config -ipType 17
pimsmInterface config -neighborIp
“0.0.0.0”
pimsmRouter addInterface interface1
pimsmRouter setDefault
pimsmRouter config -routerId
“172.26.0.1”
pimsmRouter config -enable true
pimsmRouter config -drPriority 0
pimsmRouter config -joinPruneInterval 60
pimsmRouter config -joinPruneHoldTime 180
pimsmRouter config -dataMdtTimeOut 180
pimsmRouter config -dataMdtInterval 60
pimsmServer addRouter router1
pimsmServer setDefault
pimsmServer config -enableRateControl
false
pimsmServer config -interval 0
pimsmServer config -sourceMessagesPerInterval 0
pimsmServer config -joinPruneMessagesPerInterval 0
pimsmServer config -registerStopMessagesPerInterval 0
pimsmServer config -dataMdtFramePerInterval 0
pimsmServer set
pimsmServer write
# Disable dataMdtRange dataMdtRangeLocalId1
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterface getDataMdtRangedataMdtRangeLocalId1
pimsmDataMdtRange config-enablefalse
pimsmInterfacesetInterfacedataMdtRangeLocalId1
pimsmServer write
# Get the dataMdtAddress of the second dataMdtRange in the list
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacegetFirstDataMdtRange
pimsmInterfacegetNextDataMdtRange
set dataMdtAddress[pimsmDataMdtRange cget dataMdtAddress]
# Deleting a dataMdtRange dataMdtRangeLocalId1
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacedelDataMdtRangedataMdtRangeLocalId1
pimsmServer write
# Set dataMdtAddress of dataMdtRangeLocalId1
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmDataMdtRangeconfig-dataMdtAddress< 240.0.0.20
pimsmInterfacesetDataMdtRange dataMdtRangeLocalId1
pimsmServer write
# Get the First InterfacelearnedInfo
```

```
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacerequestLearnedInfo
pimsmInterfacegetLearnedInterfaceList
pimsmInterfacegetFirstLearnedInterface
showCmd pimsmInterfaceLearnedInfo
# Get the 3rd InterfaceLearnedInfo
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacerequestLearnedInfo
pimsmInterfacegetLearnedInterfaceList
pimsmInterfacegetFirstLearnedInterface
pimsmInterfacegetNextLearnedInterface
pimsmInterfacegetNextLearnedInterface
showCmd pimsmInterfaceLearnedInfo
# Get MulticastGroupRange LearnedInfo
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacegetFirstJoinPrune
pimsmJoinPrunerequestLearnedDataMdt
pimsmJoinPrunegetLearnedDataMdt
showCmd pimsmLearnedDataMdt
# Get LearnedDataMdt for a source
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacegetFirstJoinPrune
pimsmJoinPrunerequestLearnedDataMdt
pimsmJoinPrunegetLearnedDataMdt
showCmd pimsmLearnedDataMdt
pimsmLearnedDataMdtgetAllMdtspimsmFromSource <1.1.1.1
```

```
# Get LearnedDataMdt for a group
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacegetFirstJoinPrune
pimsmJoinPrunerequestLearnedDataMdt
pimsmJoinPrunegetLearnedDataMdt
showCmd pimsmLearnedDataMdt
pimsmLearnedDataMdtgetAllMdtspimsmFromGroup <225.0.0.1
# Get DataMdtRange Learnedinfo
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacegetFirstDataMdtRange
pimsmDataMdtRangerequestMdtLearnedJoinState
pimsmDataMdtRangegetMdtLearnedJoinState
showCmd pimsmMdtLearnedJoinState
# Get MdtLearnedJoinState for a source
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacegetFirstDataMdtRange
pimsmDataMdtRangerequestMdtLearnedJoinState
pimsmDataMdtRangegetMdtLearnedJoinState
showCmd pimsmMdtLearnedJoinState
pimsmMdtLearnedJoinStategetAllMdtJoins pimsmFromSource <1.1.1.1
# Get MdtLearnedJoinState for a group
pimsmServer select $ chId $ cardId $ portId
pimsmServer getFirstRouter
pimsmRouter getFirstInterface
pimsmInterfacegetFirstDataMdtRange
pimsmDataMdtRangerequestMdtLearnedJoinState
pimsmDataMdtRangegetMdtLearnedJoinState
showCmd pimsmMdtLearnedJoinState
```

```
pimsmMdtLearnedJoinStategetAllMdtJoins pimsmFromGroup

<225.0.0.1
```

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmJoinPrune*, *pimsmRouter*, *pimsmServer*, *pimsmLearnedDataMdt*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmCRPRange

pimsmCRPRange — sets up the parameters associated with a PIM-SM Candidate Rendezvous Point range.

SYNOPSIS

pimsmCRPRange *subcommand options*

DESCRIPTION

The *pimsmCRPRange* command describes a range of Candidate Rendezvous Point (CRPs) under a PIM-SM interface. Ranges are added into *pimsmInterface* lists using the *pimsmInterface* *addCRPRange* command. Refer to *PIM-SM*67 for an overview of this command.

STANDARD OPTIONS

enable true / false

Enables the use of this CRP range on the simulated interface. *(default = false)*

cRPAddress

Start address of the set of candidate RPs to be simulated.

routerCount

Total number of candidate RPs to be simulated starting from C-RP Address. A contiguous address range is used for this RP range simulation.

meshingType

This indicates if the mappings for groups and RP addresses are **Fully-Meshed** or **One-To-One**.

Option	Value	Usage
Fully-Meshed	0	*(default)* For each group all RPs shall be advertised for candidacy which is essentially a (group to rp) mapping.
One-To-One	1	For each group only one RP shall be sent regardless of the RP count.

groupAddress

Indicates the starting group address of the group range for which the candidate RP will advertise candidacy.

groupCount

Indicates the number of groups in the range.

groupMaskLen

Mask width (prefix length in bits) for the group range.

periodicAdvertisementInterval

Rate controlling variable indicating how many C-RP-Adv messages can be sent in the specified time interval.

advertisementHoldTime

The time interval (in seconds) between two consecutive Candidate RP advertisements.

backOffInterval

The back off time interval for the C-RP-Adv messages.

priorityValue

Value of priority field sent in candidate RP advertisement messages.

priorityType

This indicates the type of priority to be held by the candidate RPs (CRPs). The options are as follows: One of:

Option	Value	Usage
Same	0	*(default)* CRPs send advertisement messages with time invariant fixed priority as specified in CRP Advertisement Message Priority.
Incremental	1	Priority starts from the configured value and with every Priority Change Interval, the CRP's priority get incremented by 1.
Random	2	The start value is selected based on a pseudorandom number generator with every Priority Change Interval, when sending the next batch of CRP-Adv messages.

priorityChangeInterval

Time interval after which priority of all the RPs get changed, if priority type is incremental or random.

COMMANDS

The **pimsmDataMdtRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmCRPRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *pimsmCRPRange* command.

pimsmCRPRange configure *option value*

Modify the configuration options of the *pimsmCRPRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *pimsmCRPRange*.

pimsmCRPRange setDefault

Sets default values for all configuration options.

pimsmCRPRange addCRPRange

Adds a new CRP range. The CRP range must have been previously configured through the use of the *pimsmCRPRange* command.

EXAMPLES

```
###############################################################
# This Script has been generated by Ixia ScriptGen
# Software Version : IxOS 5.20.401.41 EB
###############################################################
package req IxTclHal
# Command Option Mode - Full (generate full configuration)
if {[isUNIX]} {
if {[ixConnectToTclServer 10.205.27.60]} {
errorMsg "Error connecting to Tcl Server
10.205.27.60 "
return $::TCL_ERROR
}
}
######### Chassis list - {10.205.27.60} #########
ixConnectToChassis {10.205.27.60}
set owner "IxNetwork/ixin-ssarkarlap/ssarkar"
ixLogin $owner
set portList {}
######### Chassis-10.205.27.60 #########
chassis get "10.205.27.60"
set chassis [chassis cget -id]
######### Card Type : 10/100/1000 STXS4-256MB ############
set card 2
card setDefault
card config -txFrequencyDeviation 0
if {[card set $chassis $card]} {
errorMsg "Error calling card set $chassis $card"
set retCode $::TCL_ERROR
}
if {[card write $chassis $card]} {
errorMsg "Error calling card write $chassis $card"
set retCode $::TCL_ERROR
}
######### Chassis-10.205.27.60 Card-2 Port-1 #########
set port 1
port setFactoryDefaults $chassis $card $port
if {[port setPhyMode $::portPhyModeCopper $chassis $card $port]} {
errorMsg "Error calling port setPhyMode
$::portPhyModeCopper $chassis $card $port"
set retCode $::TCL_ERROR
}
port config -speed 1000
port config -duplex full
port config -flowControl false
port config -directedAddress "01 80 C2 00 00 01"
port config -multicastPauseAddress "01 80 C2 00 00 01"
port config -loopback portNormal
port config -transmitMode portTxPacketStreams
port config -receiveMode [expr
$::portRxFirstTimeStamp|$::portRxModeWidePacketGroup]
port config -autonegotiate true
port config -advertise100FullDuplex true
port config -advertise100HalfDuplex true
port config -advertise10FullDuplex true
port config -advertise10HalfDuplex true
port config -advertise1000FullDuplex true
port config -portMode portEthernetMode
port config -rxTxMode gigNormal
port config -ignoreLink false
port config -advertiseAbilities portAdvertiseNone
port config -timeoutEnable true
port config -negotiateMasterSlave 1
port config -masterSlave portSlave
port config -pmaClock
pmaClockAutoNegotiate
port config -enableSimulateCableDisconnect false
port config -enableAutoDetectInstrumentation false
port config -autoDetectInstrumentationMode
portAutoInstrumentationModeEndOfFrame
port config -enableRepeatableLastRandomPattern false
port config -transmitClockDeviation 0
port config -preEmphasis preEmphasis0
port config -MacAddress "00 de bb 00 00 01"
port config -DestMacAddress "00 de bb 00 00 02"
port config -name ""
port config -numAddresses 1
port config -enableManualAutoNegotiate false
port config -enablePhyPolling true
if {[port set $chassis $card $port]} {
errorMsg "Error calling port set $chassis $card $port"
set retCode $::TCL_ERROR
}
stat setDefault
stat config -mode statNormal
stat config -enableValidStats false
stat config -enableProtocolServerStats true
stat config -enableArpStats true
stat config -enablePosExtendedStats true
stat config -enableDhcpStats false
stat config -enableDhcpV6Stats false
stat config -enableEthernetOamStats false
stat config -enableBgpStats false
stat config -enableIcmpStats true
stat config -enableOspfStats false
stat config -enableIsisStats false
stat config -enableRsvpStats false
stat config -enableLdpStats false
stat config -enableIgmpStats false
stat config -enableOspfV3Stats false
stat config -enablePimsmStats true
stat config -enableMldStats false
stat config -enableStpStats false
stat config -enableEigrpStats false
stat config -enableBfdStats false
stat config -enableCfmStats false
stat config -enableLacpStats false
if {[stat set $chassis $card $port]} {
errorMsg "Error calling stat set $chassis $card $port"
set retCode $::TCL_ERROR
}
packetGroup setDefault
packetGroup config -signatureOffset 48
packetGroup config -signature "08 71 18 05"
packetGroup config -insertSignature false
packetGroup config -ignoreSignature false
packetGroup config -groupId 0
packetGroup config -groupIdOffset 52
packetGroup config -enableGroupIdMask false
packetGroup config -enableInsertPgid true
packetGroup config -groupIdMask 0
packetGroup config -latencyControl cutThrough
packetGroup config -preambleSize 8
packetGroup config -sequenceNumberOffset 44
packetGroup config -sequenceErrorThreshold 2
packetGroup config -insertSequenceSignature false
packetGroup config -allocateUdf true
packetGroup config -enableSignatureMask false
packetGroup config -signatureMask "00 00 00 00"
packetGroup config -enableRxFilter false
packetGroup config -headerFilter "00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00"
packetGroup config -headerFilterMask "00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00"
packetGroup config -enable128kBinMode false
packetGroup config -enableTimeBins false
packetGroup config -numPgidPerTimeBin 32
packetGroup config -numTimeBins 1
packetGroup config -timeBinDuration 1000000
packetGroup config -enableLatencyBins false
packetGroup config -latencyBinList ""
packetGroup config -groupIdMode
packetGroupCustom
packetGroup config -sequenceCheckingMode seqThreshold
packetGroup config -multiSwitchedPathMode
seqSwitchedPathPGID
if {[packetGroup setRx $chassis $card $port]} {
errorMsg "Error calling packetGroup setRx $chassis $card
$port"
set retCode $::TCL_ERROR
}
flexibleTimestamp setDefault
flexibleTimestamp config -type
timestampBeforeCrc
flexibleTimestamp config -offset 23
if {[flexibleTimestamp set $chassis $card $port]} {
errorMsg "Error calling flexibleTimestamp set $chassis
$card $port"
set retCode $::TCL_ERROR
}
capture setDefault
capture config -fullAction lock
capture config -sliceSize 8191
capture config -sliceOffset 0
capture config -captureMode
```

```
captureTriggerMode
capture config -continuousFilter 0
capture config -beforeTriggerFilter
captureBeforeTriggerNone
capture config -afterTriggerFilter
captureAfterTriggerFilter
capture config -triggerPosition 1.0
capture config -enableSmallPacketCapture false
if {[capture set $chassis $card $port]} {
errorMsg "Error calling capture set $chassis $card $port"
set retCode $::TCL_ERROR
}
filter setDefault
filter config -captureTriggerDA anyAddr
filter config -captureTriggerSA anyAddr
filter config -captureTriggerPattern anyPattern
filter config -captureTriggerError errAnyFrame
filter config -captureTriggerFrameSizeEnable false
filter config -captureTriggerFrameSizeFrom 12
filter config -captureTriggerFrameSizeTo 12
filter config -captureTriggerCircuit filterAnyCircuit
filter config -captureFilterDA anyAddr
filter config -captureFilterSA anyAddr
filter config -captureFilterPattern anyPattern
filter config -captureFilterError errAnyFrame
filter config -captureFilterFrameSizeEnable false
filter config -captureFilterFrameSizeFrom 12
filter config -captureFilterFrameSizeTo 12
filter config -captureFilterCircuit filterAnyCircuit
filter config -userDefinedStat1DA anyAddr
filter config -userDefinedStat1SA anyAddr
filter config -userDefinedStat1Pattern anyPattern
filter config -userDefinedStat1Error errAnyFrame
filter config -userDefinedStat1FrameSizeEnable false
filter config -userDefinedStat1FrameSizeFrom 12
filter config -userDefinedStat1FrameSizeTo 12
filter config -userDefinedStat1Circuit filterAnyCircuit
filter config -userDefinedStat2DA anyAddr
filter config -userDefinedStat2SA anyAddr
filter config -userDefinedStat2Pattern anyPattern
filter config -userDefinedStat2Error errAnyFrame
filter config -userDefinedStat2FrameSizeEnable 0
filter config -userDefinedStat2FrameSizeFrom 12
filter config -userDefinedStat2FrameSizeTo 12
filter config -userDefinedStat2Circuit filterAnyCircuit
filter config -asyncTrigger1DA anyAddr
filter config -asyncTrigger1SA anyAddr
filter config -asyncTrigger1Pattern anyPattern
filter config -asyncTrigger1Error errAnyFrame
filter config -asyncTrigger1FrameSizeEnable false
filter config -asyncTrigger1FrameSizeFrom 12
filter config -asyncTrigger1FrameSizeTo 12
filter config -asyncTrigger1Circuit filterAnyCircuit
filter config -asyncTrigger2DA anyAddr
filter config -asyncTrigger2SA anyAddr
filter config -asyncTrigger2Pattern anyPattern
filter config -asyncTrigger2Error errAnyFrame
filter config -asyncTrigger2FrameSizeEnable false
filter config -asyncTrigger2FrameSizeFrom 12
filter config -asyncTrigger2FrameSizeTo 12
filter config -asyncTrigger2Circuit filterAnyCircuit
filter config -captureTriggerEnable true
filter config -captureFilterEnable true
filter config -userDefinedStat1Enable false
filter config -userDefinedStat2Enable false
filter config -asyncTrigger1Enable false
filter config -asyncTrigger2Enable false
if {[filter set $chassis $card $port]} {
```

```
errorMsg "Error calling filter set $chassis $card $port"
set retCode $::TCL_ERROR
}
filterPallette setDefault
filterPallette config -DA1 "00 00 00
00 00 00"
filterPallette config -DAMask1 "00 00 00
00 00 00"
filterPallette config -DA2 "00 00 00
00 00 00"
filterPallette config -DAMask2 "00 00 00
00 00 00"
filterPallette config -SA1 "00 00 00
00 00 00"
filterPallette config -SAMask1 "00 00 00
00 00 00"
filterPallette config -SA2 "00 00 00
00 00 00"
filterPallette config -SAMask2 "00 00 00
00 00 00"
filterPallette config -pattern1 "DE ED EF
FE AC CA"
filterPallette config -patternMask1 "00 00 00
00 00 00"
filterPallette config -pattern2 00
filterPallette config -patternMask2 00
filterPallette config -patternOffset1 12
filterPallette config -patternOffset2 12
filterPallette config -matchType1 matchUser
filterPallette config -matchType2 matchUser
filterPallette config -patternOffsetType1
filterPalletteOffsetStartOfFrame
filterPallette config -patternOffsetType2
filterPalletteOffsetStartOfFrame
```

```
filterPallette config -gfpErrorCondition
gfpErrorsOr
filterPallette config -enableGfptHecError true
filterPallette config -enableGfpeHecError true
filterPallette config -enableGfpPayloadCrcError true
filterPallette config -enableGfpBadFcsError true
filterPallette config -circuitList ""
if {[filterPallette set $chassis $card $port]} {
errorMsg "Error calling filterPallette set $chassis $card
$port"
set retCode $::TCL_ERROR
}
ipAddressTable setDefault
ipAddressTable config -defaultGateway "0.0.0.0"
if {[ipAddressTable set $chassis $card $port]} {
errorMsg "Error calling ipAddressTable set $chassis $card
$port"
set retCode $::TCL_ERROR
}
arpServer setDefault
arpServer config -retries 3
arpServer config -mode arpGatewayOnly
arpServer config -rate 2083333
arpServer config -requestRepeatCount 3
if {[arpServer set $chassis $card $port]} {
errorMsg "Error calling arpServer set $chassis $card
$port"
set retCode $::TCL_ERROR
}
if {[interfaceTable select $chassis $card $port]} {
errorMsg "Error calling interfaceTable select $chassis
$card $port"
set retCode $::TCL_ERROR
}
```

```
interfaceTable setDefault
interfaceTable config -dhcpV4RequestRate 0
interfaceTable config -dhcpV6RequestRate 0
interfaceTable config -dhcpV4MaximumOutstandingRequests 100
interfaceTable config -dhcpV6MaximumOutstandingRequests 100
if {[interfaceTable set]} {
errorMsg "Error calling interfaceTable set"
set retCode $::TCL_ERROR
}
interfaceTable clearAllInterfaces
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress {0.0.0.0}
interfaceIpV4 config -maskWidth 24
interfaceIpV4 config -ipAddress {0.0.0.0}
if {[interfaceEntry addItem addressTypeIpV4]} {
errorMsg "Error calling interfaceEntry addItem
addressTypeIpV4"
set retCode $::TCL_ERROR
}
dhcpV4Properties removeAllTlvs
dhcpV4Properties setDefault
dhcpV4Properties config -clientId ""
dhcpV4Properties config -serverId
"0.0.0.0"
dhcpV4Properties config -vendorId ""
dhcpV4Properties config -renewTimer 0
dhcpV4Properties config -relayAgentAddress
"0.0.0.0"
dhcpV4Properties config -relayDestinationAddress
"255.255.255.255"
dhcpV6Properties removeAllTlvs
```

```
dhcpV6Properties setDefault
dhcpV6Properties config -iaType
dhcpV6IaTypePermanent
dhcpV6Properties config -iaId 0
dhcpV6Properties config -renewTimer 0
dhcpV6Properties config -relayLinkAddress
"0:0:0:0:0:0:0:0"
dhcpV6Properties config -relayDestinationAddress
"FF05:0:0:0:0:0:1:3"
interfaceEntry config -enable true
interfaceEntry config -description {Connected
- 0.0.0.0/24 - 100:01 - 1}
interfaceEntry config -macAddress {00 00 01
14 EE A7}
interfaceEntry config -eui64Id {02 00 01
FF FE 14 EE A7}
interfaceEntry config -atmEncapsulation
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableDhcpV6 false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 1
interfaceEntry config -vlanPriority 0
if {[interfaceTable addInterface 0]} {
errorMsg "Error calling interfaceTable addInterface 0"
set retCode $::TCL_ERROR
}
if {[pimsmServer select $chassis $card $port]} {
errorMsg "Error calling pimsmServer select $chassis $card
$port"
set retCode $::TCL_ERROR
}
```

```
pimsmServer clearAllRouters
pimsmCRPRange setDefault
pimsmCRPRange config -enable true
pimsmCRPRange config -cRPAddress "0.0.0.1"
pimsmCRPRange config -routerCount 1
pimsmCRPRange config -meshingType 0
pimsmCRPRange config -groupAddress "225.0.0.0"
pimsmCRPRange config -groupCount 1
pimsmCRPRange config -groupMaskLen 32
pimsmCRPRange config -periodicAdvertisementInterval 60
pimsmCRPRange config -advertisementHoldTime 150
pimsmCRPRange config -backOffInterval 3
pimsmCRPRange config -priorityValue 192
pimsmCRPRange config -priorityType 0
pimsmCRPRange config -priorityChangeInterval 60
pimsmInterface addCRPRange cRPRange1
pimsmInterface setDefault
pimsmInterface config -enable true
pimsmInterface config -helloInterval 30
pimsmInterface config -helloHoldTime 105
pimsmInterface config -disableTriggeredHello 0
pimsmInterface config -triggeredHelloDelay 5
pimsmInterface config -protocolInterfaceDescription
"Connected - 0.0.0.0/24 - 100:01 - 1"
pimsmInterface config -enablePruneDelay true
pimsmInterface config -pruneDelay 500
pimsmInterface config -overrideInterval 2500
pimsmInterface config -enableSendBidirectionalOption false
pimsmInterface config -enableBFDRegistration false
pimsmInterface config -generationIdMode 2
pimsmInterface config -enableSendGenerationId true
pimsmInterface config -enablePruneDelayTBit false
pimsmInterface config -enableAutoPickNeighbor true
pimsmInterface config -ipType 17
pimsmInterface config -neighborIp "0.0.0.0"
pimsmInterface config -enableDiscardDataMdt true
pimsmInterface config -useV4MappedV6Address 0
pimsmInterface config -enableBootstrap true
pimsmInterface config -bootstrapPriority 64
pimsmInterface config -bootstrapHashMaskLen 30
pimsmInterface config -bootstrapInterval 60
pimsmInterface config -bootstrapTimeout 130
pimsmInterface config -forceSemanticFragmentation 0
pimsmInterface config -supportUnicastBootstrap 1
pimsmInterface config -discardLearntRPInfo 0
if {[pimsmRouter addInterface interface1]} {
errorMsg "Error calling pimsmRouter addInterface
interface1"
set retCode $::TCL_ERROR
}
pimsmRouter setDefault
pimsmRouter config -routerId "15.156.0.1"
pimsmRouter config -enable true
pimsmRouter config -drPriority 0
pimsmRouter config -joinPruneInterval 60
pimsmRouter config -joinPruneHoldTime 180
pimsmRouter config -dataMdtTimeOut 180
pimsmRouter config -dataMdtInterval 60
pimsmServer addRouter router1
pimsmServer setDefault
pimsmServer config -enableRateControl false
pimsmServer config -interval 0
pimsmServer config -sourceMessagesPerInterval 0
pimsmServer config -joinPruneMessagesPerInterval 0
pimsmServer config -registerStopMessagesPerInterval 0
pimsmServer config -dataMdtFramePerInterval 0
pimsmServer config -crpFramePerInterval 0
pimsmServer config -bsmFramePerInterval 0
```

```
if {[pimsmServer set]} {
errorMsg "Error calling pimsmServer set"
set retCode $::TCL_ERROR
}
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse false
protocolServer config -enableIgmpQueryResponse false
protocolServer config -enableOspfService false
protocolServer config -enableBgp4Service false
protocolServer config -enableIsisService false
protocolServer config -enableRsvpService false
protocolServer config -enableRipService false
protocolServer config -enableLdpService false
protocolServer config -enableRipngService false
protocolServer config -enableMldService false
protocolServer config -enableOspfV3Service false
protocolServer config -enablePimsmService true
protocolServer config -enableStpService false
protocolServer config -enableEigrpService false
protocolServer config -enableBfdService false
protocolServer config -enableCfmService false
protocolServer config -enableLacpService false
protocolServer config -enableBgp4CreateInterface false
protocolServer config -enableIsisCreateInterface false
protocolServer config -enableOspfCreateInterface false
protocolServer config -enableRipCreateInterface false
protocolServer config -enableRsvpCreateInterface false
protocolServer config -enableIgmpCreateInterface false
if {[protocolServer set $chassis $card $port]} {
errorMsg "Error calling protocolServer set $chassis $card
$port"
set retCode $::TCL_ERROR
}
```

```
oamPort setDefault
oamPort config -enable false
oamPort config -macAddress "00 00 00 00 00
00"
oamPort config -enableLoopback false
oamPort config -enableLinkEvents false
oamPort config -maxOamPduSize 1518
oamPort config -oui "00 00 00"
oamPort config -vendorSpecificInformation "00 00 00 00"
oamPort config -idleTimer 5
oamPort config -enableOptionalTlv false
oamPort config -optionalTlvType 0
oamPort config -optionalTlvValue 00
if {[oamPort set $chassis $card $port]} {
errorMsg "Error calling oamPort set $chassis $card $port"
set retCode $::TCL_ERROR
}
lappend portList [list $chassis $card $port]
ixWritePortsToHardware portList
ixCheckLinkState portList
#################################################################
##
######### Generating streams for all the ports from above
#########
#################################################################
##
######### Chassis-10.205.27.60 Card-2 Port-1 #########
chassis get "10.205.27.60"
set chassis [chassis cget -id]
set card 2
set port 1
if {[port reset $chassis $card $port]} {
errorMsg "Error calling port reset $chassis $card $port"
set retCode $::TCL_ERROR
}

ixWriteConfigToHardware portList -noProtocolServer
```

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmJoinPrune*, *pimsmRouter*, *pimsmServer*, *pimsmLearnedDataMdt*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmInterface

pimsmInterface — configures an interface for a PIM-SM router.

SYNOPSIS

pimsmInterface *subcommand options*

DESCRIPTION

The *pimsmInterface* command holds the information related to a single interface on the simulated router. Interfaces are added into the *pimsmRouter* interface list using the *pimsmRouter* *addInterface* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on PIM-SM testing with Ixia equipment. Refer to *PIM-SM* for an overview.

Several lists are maintained for each interface:

- Joins/Prunes — a list of multicast address ranges that the interface is interested in receiving.
- Sources — a list of multicast addresses for which the interface will send Register messages to a Rendezvous Point.
- Data MDT Ranges — a list of Data MDT ranges of multicast addresses.
- Candidate RPs — a list of candidate rendezvous points behind emulated routers connected by one or more interfaces with the DUT.

STANDARD OPTIONS

disableTriggeredHello true |false

If true, then triggered HELLO messages are disabled. *(default = false)*

enable true |false

If set, enables the use of this interface for the simulated router. *(default = false)*

enableAutoPick Neighbor

If set, then the auto-pick neighbor feature is enabled and the *neighborIp*field is ignored. In this mode, the upstream neighbor address used in the join/prune messages is determined automatically from received Hello messages. The first time a Hello message is received containing a source address that does not belong to this interface, that source address will be used as the upstream neighbor address. *(default = true)*

enableBfdRegistration true |false

Indicates if a BFD session is to be created to the PIMSM peer IP address once the PIMSM session is established. This allows PIMSM to use BFD to maintain IPv4 connectivity the PIMSM peer.

enableDiscardDataMdt true | false

If set, Data MDT learned information will be discarded and Interface learned information will be available. *(default = false)*

enablePruneDelay true | false

If set, the LAN Prune propagation delay is enabled for this interface, as indicated in the *pruneDelay*option. The option is indicated in Hello messages from the interface. *(default = true)*

enablePruneDelayTBit true | false

If set, the T flag bit in the LAN Prune Delay option of the Hello message is set (=1). Setting this bit specifies that the sending PIM-SM router has the ability to disable Join message suppression. *(default = false)*

enableSend BidirectionalOption true | false

If set, sets the header bi-directional PIM-SM flag bit (=1), per IETF DRAFT draft-ietf-pim-bidir-04. **NOTE**: Designated Forwarder election is not currently supported. *(default = false)*

enableSendGeneration Id true | false

If set, the Send generation ID option is enabled, and the Generation ID Mode field will become available to make a mode selection. *(default = true)*

generationIdMode

The mode used for creating the 32-bit value for the Generation Identifier (GenID) option in Hello messages. A new GenID is created each time an interface or router starts or restarts PIM-SM forwarding. A change in this value indicates to the neighbors that the change of state may have occurred, and that the old Hello information received from these interfaces should be discarded. The choices of *generationIdMode* are:

Option	Value	Usage
pimsmGenerationIdModeIncremental	0	The GenID increases by 1 for each successive Hello message.
pimsmGenerationIdModeRandom	1	The GenID is randomly generated for each successive Hello message.
pimsmGenerationIdModeConstant	2	*(default)* The GenID remains the same in all of the Hello messages.

helloHoldTime

The timeout period, in seconds, specified in Hello messages. This is the length of time the receiver of this message must keep the neighbor reachable. The default is 3.5 times the *helloInterval*value. *(default = 105)*

helloInterval

The length of time, in seconds, between the transmission of Hello messages. *(default = 30)*

ipType

The IP addressing type of *neighborIp*, one of the following options:

Option	Value	Usage
addressTypeIpV4	17	*(default)* An IPv4 address is added from the options associated with the *interfaceIpV4* command.
addressTypeIpV6	18	An IPv6 address is added from the options associated with the *interfaceIpV6* command.

neighborIp

The neighbor's IP address, of the type indicated by *ipType. (default = "0.0.0.0")*

overrideInterval

The delay interval, in milliseconds, for randomizing the transmission time for override messages, which are used when scheduling a delayed Join message. This is part of the LAN Prune Delay option included in Hello messages. *(default = 2,500)*

protocolInterface Description

The *description* option associated with an *pimsmServer* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

pruneDelay

The value, in seconds, of the LAN Prune propagation delay for this interface. It indicates to an upstream router how long to wait for a Join override message before it prunes an interface. *(default = 1,000)*

triggeredHelloDelay

The value, in seconds, of the triggered HELLO delay for this interface. It indicates to an upstream router how long to wait for a HELLO message before it. *(default = 1,000)*

usev4mappedV6 Address true | false

Indicates that PIMSM will use an IPv6 type address (which is the v4-mapped-v6 address on the GRE interface) as the source address instead of using the link-local address in the hello packets.

enableBootstrap true | false

If set, enables the PIM-SM interface to participate in Bootstrap Router election procedure. *(default = false)*

bootstrapPriority

Indicates the priority of the Bootstrap Router (BSR) that is set with the same name in all Bootstrap Messages sent by this BSR. *(default = 64) (range = 1-255)*

bootstrapHashMaskLen

Hash Mask Length of the Bootstrap Router (BSR) that is set with the same name in all Bootstrap Messages sent by this BSR. *(default = 30(IPv4) / 126(IPv6)) (range 1-32(1Pv4) / 1-128(IPv6))*

bootstrapInterval

Indicates the time interval (in seconds) between two consecutive bootstrap messages sent by the BSR. *(default = 60) (range = 1-65535)*

bootstrapTimeout

Amount of time (in seconds) of not receiving any Bootstrap Messages, after which the BSR if candidate at that point of time; will decide that the currently elected BSR has gone down and will restart BSR election procedure. *(default = 130) (range = 1-65535)*

forceSemanticFragmentation true | false

If set, this forces the BSR to send only one group specific RP list per bootstrap message, even if there is space in the packet to push in more RP list information pertaining to a different group. *(default = false)*

supportUnicastBootstrap true | false

If enabled, this supports the sending and processing of Unicast bootstrap messages. *(default = true)*

discardLearntRPInfo true | false

If set, disregards group mappings learnt from Bootstrap Message (in case not acting as elected BSR) or from Candidate RP Advertisement (in case of elected BSR). *(default = true)*

learnSelectedRPSet

If enabled, it controls whether all RP-to-group mappings are stored or the selected RP set consisting of one best RP per group is stored.

DEPRECATED OPTIONS

neighborIpV6

Replaced by *neighborIp*and *ipType.*

COMMANDS

The **pimsmInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmInterface addCRPRange crpRangeLocalId

Adds a new CRP range. The CRP range must have been previously configured through the use of the *pimsmCRPRange* command.

pimsmInterface addDataMdtRange *dataMdtLocalId*

Adds the set of Data MDT addresses described in the *pimsmDataMdtRange* command to the list of Data MDT Ranges associated with the interface. The Data MDT range entry in the list is given an identifier of *dataMdtRangeLocalId*.

pimsmInterface addJoinPrune *joinsPrunesLocalId*

Adds the set of joins/prunes described in the *pimsmJoinPrune* command to the list of joins/prunes associated with the interface. The joins/prunes's entry in the list is given an identifier of *joinsPrunesLocalId*. Specific errors are:

- The parameters in *pimsmJoinPrune* are invalid.
- A router with this *JoinPruneLocalId* exists already in the list.

pimsmInterface addSource *sourcesLocalId*

Adds the source described in the *pimsmSource* command to the list of sources associated with the interface. The source's entry in the list is given an identifier of *sourcesLocalId*. Specific errors are:

- The parameters in *pimsmSource* are invalid.
- A router with this *SourceLocalId* exists already in the list.

pimsmInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *pimsmInterface* command.

pimsmInterface clearCRPRanges

Deletes all of the CRP Ranges.

pimsmInterface clearAllDataMdtRanges

Deletes all of the Data MDT Ranges.

pimsmInterface clearAllJoinsprune

Deletes all of the joins/prune.

pimsmInterface clearAllSources

Deletes all of the sources.

pimsmInterface configuration *option value*

Modify the configuration options of the *pimsmInterface*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for pimsmInterface.

pimsmInterface delCRPRange crpRangeLocalId

Deletes the CRP Range with an identifier of *crpRangeLocalId*.

pimsmInterface delDataMdtRange *dataMdtRangeLocalId*

Deletes the Data MDT Range with an identifier of *dataMdtRangeLocalId*.

pimsmInterface delJoinPrune *joinsPrunesLocalId*

Deletes the joins/prunes with an identifier of *joinsPrunesLocalId*. Specific errors are:

- No joins/prunes with this *JoinPruneLocalId* exists in the list.

pimsmInterface delSource *sourcesLocalId*

Deletes the source with an identifier of *sourcesLocalId*. Specific errors are:

- No source with this *SourceLocalId* exists in the list.

pimsmInterface getCRPRange crpRangeLocalId

Accesses the CRP Range entry in the list with an identifier of *crpRangeLocalId*. The joins/prune is accessed in the *pimsmLearnedCRPInfo* command.

pimsmInterface getDataMdtRange *dataMdtRangeLocalId*

Accesses the Data MDT Range entry in the list with an identifier of *DataMdtLocalId*. The joins/prune is accessed in the *pimsmDataMdtRange* command.

pimsmInterface getFirstCRPRange

Access the first CRP Range in the list. The results may be accessed using the *pimsmLearnedCRPInfo* command.

pimsmInterface getFirstDataMdtRange

Access the first Data MDT Range in the list. The results may be accessed using the *pimsmDataMdtRange* command.

pimsmInterface getFirstJoinPrune

Access the first joins/prunes *in the list*. The results may be accessed using the *pimsmJoinPrune* command. Specific errors are:

- *pimsmServer* select has not been called.
- There are no joins/prune in the list.

pimsmInterface getFirstSource

Access the first source *in the list*. The results may be accessed using the *pimsmSource* command. Specific errors are:

- *pimsmServer* select has not been called.
- There are no sources in the list.

pimsmInterface getJoinPrune *joinsPrunesLocalId*

Accesses the joins/prune's entry in the list with an identifier of *joinsPrunesLocalId*. The joins/prune is accessed in the *pimsmJoinPrune* command. Specific errors are:

- A joins/prune with this JoinPruneLocalId does not exist in the list.

pimsmInterface getSource *sourcesLocalId*

Accesses the source's entry in the list with an identifier of *sourcesLocalId*. The source is accessed in the *pimsmSource* command. Specific errors are:

- A source with this SourceLocalId does not exist in the list.

pimsmInterface getNextCRPRange

Access the nexNAME - pimsmLearnedCRPInfot CRP Range in the list. The results may be accessed using the *pimsmLearnedCRPInfo* command.

pimsmInterface getNextDataMdtRange

Access the next Data MDT Range in the list*.* The results may be accessed using the *pimsmDataMdtRange* command.

pimsmInterface getNextJoinPrune

Access the next joins/prune in the list. The results may be accessed using the *pimsmJoinPrune* command. Specific errors are:

- *pimsmInterface getFirstJoinPrune* has not been called.
- There is no more joins/prunes in the list.

pimsmInterface getNextSource

Access the next source in the list. The results may be accessed using the *pimsmSource* command. Specific errors are:

- *pimsmInterface getFirstSource* has not been called.
- There is no more sources in the list.

pimsmInterface setCRPRange crpRangeLocalId

Sets the values for the CRP Range entry in the list with an identifier of crpRangeLocalId based on changes made through the *pimsmLearnedCRPInfo* command.

This command can be used to change a running configuration and must be followed by an *pimsmServer* write command in order to send these changes to the protocol server.

pimsmInterface setDataMdtRange *[dataMdtRangeLocalId]*

Sets the values for the Data MDT Range entry in the list with an identifier of *dataMdtRangeLocalId* based on changes made through the *pimsmDataMdtRange* command. *dataMdtRangeLocalId* may only be omitted if *getFirstDataMdtRange* and *getNextDataMdtRange* were used to select the Data MDT Range, in which case the currently selected Data MDT Range is set. This command can be used to change a running configuration and must be followed by an *pimsmServer* *write* command in order to send these changes to the protocol server.

pimsmInterface setDefault

Sets default values for all configuration options.

pimsmInterface setJoinPrune *[joinsPrunesLocalId]*

Sets the values for the joins/prune's entry in the list with an identifier of *joisPrunesLocalId* based on changes made through the *pimsmJoinPrune* command. *joinsPrunesLocalId* may only be omitted if *getFirstJoinPrune* and *getNextJoinPrune* were used to select the joins/prune, in which case the currently selected joins/prune is set. This command can be used to change a running configuration and must be followed by an *pimsmServer* *write* command in order to send these changes to the protocolserver. Specific errors are:

- A router with this *JoinPruneLocalId* does not exist in the list.

pimsmInterface setSource *[sourcesLocalId]*

Sets the values for the source's entry in the list with an identifier of *SourceLocalId* based on changes made through the pimsmSource command. *SourceLocalId* may only be omitted if *getFirstSource* and *getNextSource* were used to select the source, in which case the currently selected source is set. This command can be used to change a running configuration and must be followed by an *pimsmServer* *write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *SourceLocalId* does not exist in the list.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterfaceLearnedInfo, *pimsmLearnedJoinState*, *pimsmJoinPrune*, *pimsmRouter*, *pimsmServer*, *pimsmLearnedDataMdt*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmInterfaceLearnedInfo

pimsmInterfaceLearned — accesses learned information associated with a PIM-SM interface.

SYNOPSIS

pimsmInterfaceLearnedInfo *subcommand options*

DESCRIPTION

The *pimsmInterfaceLearnedInfo* commandmakes available Data MDT TLV information learned by the simulated interface in the current *pimsmInterface* command. (This information will be visible only for a GRE interface.) Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

mdtGroupAddress

(Read-only) The learned MDT (PE) Group Address contained in this Data MDT TLV.

mdtSourceAddress

(Read-only) The learned MDT (PE) Source Address contained in this Data MDT TLV.

ceGroupAddress

(Read-only) The learned MDT CE Group address contained in this Data MDT TLV.

ceSourceAddress

(Read-only) The learned MDT CE Source address contained in this Data MDT TLV.

age

(Read-only) The amount time remaining before this Data MDT TLV times out, in seconds.

COMMANDS

The **pimsmInterfaceLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmInterfaceLearnedInfo cget

Returns the options available for the *pimsmInterfaceLearnedInfo* command.

pimsmInterfaceLearnedInfo setDefault

Sets the options to default values.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterface, *pimsmLearnedJoinState*, *pimsmJoinPrune*, *pimsmRouter*, *pimsmRouter*, *pimsmServer*, *pimsmLearnedDataMdt*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmJoinPrune

pimsmJoinPrune — sets up the parameters associated with a PIM-SM join/prune.

SYNOPSIS

pimsmJoinPrune *subcommand options*

DESCRIPTION

The *pimsmJoinPrune* command describes a join/prune of addresses that a PIM-SM interface is interested in receiving. Route ranges are added into *pimsmInterface* lists using the *pimsmInterface addJoinPrune* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on PIM-SM testing with Ixia equipment. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

enable true / false

Enables the use of this join/prune for the simulated interface. *(default = false)*

enableDataMdtFlag true / false

If true, *pimsmLearnedDataMdt* will be available. *(default = false)*

enableFlap true / false

If *true,* enables simulated flapping of this join/prune. *(default = false)*

enablePacking true / false

If *true*, multiple groups are included within a single packet. *(default = true)*

flapInterval

If *enableFlap* is *true*, this is the amount of time, in seconds, between simulated flap events. *(default = 60)*

groupAddress

An IPv4 address used with the *groupMaskWidth* to create a range of multicast addresses. Not used with *rangeType = pimsmJoinPruneTypeG. (default = 225.0.0.0)*

groupAddressCount

The number of multicast group addresses to be included in the multicast group range. Not used with *rangeType = pimsmJoinPruneTypeG. (default = 1)*

groupMaskWidth

The number of bits in the mask applied to *groupAddress*. *(default = 32)*

pruneSourceAddress

The prune source address (which must be a unicast address). Not used with *rangeType = pimsmJoinPruneTypeRP. (default = 0.0.0.0)*

pruneSourceAddress Count

The number of prune source addresses to be included. *(default = 0)*

pruneSourceMaskWidth

The number of bits in the mask applied to *pruneAddress. (default = 32)*

rangeType

The joins/prune type. One of:

Option	Value	Usage
pimsmJoinsPrunesTypeRP	0	*(default)* (*,*,RP) wildcard group set. Refers to all groups associated with a specific RP.
pimsmJoinsPrunesTypeG	1	(*,G) group specific type. Refers to all sources associated with a specific group G.
pimsmJoinsPrunesTypeSG	2	(S,G) source specific type. Refers only to specific combinations of source S and group G.
pimsmJoinsPrunesTypeSPT SwitchOver	3	(*,G)-(S,G) switchover type. Indicates that the simulated router will switch over from a non-source specific group state to a source-specific group state.
pimsmJoinsPrunesType RegisterTriggeredSG	4	Sends (S,G) when matching registers have been received. Sends register stop after *registerStopTriggerCount* registers have been received.

registerStopTrigger Count

If *rangeType* is set to *pimsmJoinsPrunesTypeRegisterTriggeredSG*, then this is the count of register messages received that will trigger transmission of a (S,G) message. *(default = 10)*

rpAddress

The IP address of the Rendezvous Point (RP) router, the root of the RP shared multicast distribution tree (RPT). *(default = 0.0.0.0)*

sourceAddress

The source address that generates multicast traffic. It must be a unicast address.

sourceAddressCount

The number of source addresses that generate multicast traffic. *(default = 1)*

sourceGroupMapping

Sets the type of mapping that occurs when routes are advertised. This only applies for (S, G) and switchover types for MGR and is meaningful for RR. One of:

Option	Value	Usage
pimsmMappingFullyMeshed	0	*(default)* All sources to all groups.
pimsmMappingOneToOne	1	One source to one group.

sourceMaskWidth

The number of bits in the mask applied to *sourceAddress*. *(default = 32)*

switchoverInterval

The time interval, in seconds, allowed for the switch from using the RP tree to using a source-specific tree. Used for *rangeType = pimsmJoinPruneTypeGSPTSwitchOver. (default = 5)*

COMMANDS

The **pimsmJoinPrune** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmJoinPrune cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *pimsmJoinPrune* command.

pimsmJoinPrune config *option value*

Modify the configuration options of the pimsmJoinPrune. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *pimsmJoinPrune*.

pimsmJoinPrune getLearnedDataMdt

Accesses the learned Data MDT information.

pimsmJoinPrune requestLearnedDataMdt

Requests the learned Data MDT information.

pimsmJoinPrune setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmRouter*, *pimsmServer*, *pimsmLearnedDataMdt*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmLearnedDataMdt

pimsmLearnedDataMdt — accesses learned Data MDT information associated with a PIM-SM MDT Join/Prune.

SYNOPSIS

pimsmLearnedDataMdt *subcommand options*

DESCRIPTION

The *pimsmLearnedDataMdt* command makes available learned Data MDT information associated with the current *pimsmJoinPrune* command. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

numGroupsPerSource

(Read-only) The number of groups received per source address.

numSources

(Read-only) The number of sources associated with the *pimsmLearnedDataMdt*.

COMMANDS

The **pimsmLearnedDataMdt** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmLearnedDataMdt cget

Returns the current value of the configuration option given by *option*.

pimsmLearnedDataMdt getAllDataMdts

Accesses the learned information for all Data MDTs associated with this Data MDT Join/Prune.

pimsmLearnedDataMdt getDataMdt

Accesses the learned information for a Data MDT associated with this Data MDT Join/Prune.

pimsmLearnedDataMdt setDefault

Sets the options to default values.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmRouter*, *pimsmServer*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmLearnedBSRInfo

pimsmLearnedBSRInfo — accesses learned BSR information associated with a PIM-SM Bootstrap router.

SYNOPSIS

pimsmLearnedBSRInfo *subcommand options*

DESCRIPTION

The *pimsmLearnedBSR* command makes available learned BSR information associated with the current *pimsmLearnedBSRInfo* command. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

bSRAddress

(Read-only) The address of the elected bootstrap Router that is sending periodic bootstrap messages.

bSRPriority

(Read-only) Priority of the elected Bootstrap router as received in Bootstrap messages or configured priority.

bSRState

(Read-only) Indicates the state of the configured bootstrap router. The options are

- Not started
- Pending
- Candidate
- Elected

bSRTimerValue

(Read-only) Indicates the elapsed time (in seconds) since last bootstrap message was received (in case not acting as elected bootstrap router) or since last bootstrap message was sent (in case of elected bootstrap router).

COMMANDS

The **pimsmLearnedBSR** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmLearnedBSR cget

Returns the current value of the configuration option given by *option*.

pimsmLearnedBSR requestLearnedCRPBSRInfo

Requests the learned BSR information for the respective BSR Router.

pimsmLearnedBSR getLearnedBSRInfo

Retrieves the full list of learned BSR info.

pimsmLearnedBSR setDefault

Sets the options to default values.

EXAMPLES

See examples under *pimsmCRPRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmRouter*, *pimsmServer*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmLearnedCRPInfo

pimsmInterfaceLearnedCRPInfo — accesses learned CRP information associated with a PIM-SM Interface.

SYNOPSIS

pimsmInterfaceLearnedCRP *subcommand options*

DESCRIPTION

The *pimsmInterfaceLearnedCRPInfo* command makes available learned CRP information associated with the current *pimsmLearnedCRPInfo* command. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

cRPAddress

(Read-only) The RP address expressing candidacy for this specific group. If the entire set is displayed then, there can be multiple RPs that have expressed candidacy for the same group.

groupAddress

(Read-only) Group Address learned through Candidate RP Advertisements or Bootstrap Messages.

Configured C-RP-Range values on this PIM interface are not shown here.

cRPPriority

(Read-only) Indicates thepriority of this candidate RP.

mappingExpiryTimerValue

(Read-only) The expiry time for this specific record as received in C-RP-Adv Message/Bootstrap Message.

groupMaskWidth

The number of bits in the mask applied to the group address. (The masked bits in the group address form the address prefix.)The default value is 32. The valid range is 1 to 128, depending on address family type.

COMMANDS

The **pimsmInterfaceLearnedCRPInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmInterfaceLearnedCRPInfo

Returns the current value of the configuration option given by *option*.

pimsmInterfaceLearnedCRPInfo setDefault

Sets the options to default values.

pimsmInterfaceLearnedCRPInfo requestLearnedCRPBSRInfo

Requests the learned CRP information for the respective PIMSM interface.

pimsmInterfaceLearnedCRPInfo **getFirstLearnedCRPInfo**

Retrieves the first learned info entry, then iterates through the list of additional entries.

pimsmInterfaceLearnedCRPInfo **getNextLearnedCRPInfo**

Retrieves the first learned info entry, then iterates through the list of additional entries.

EXAMPLES

See examples under *pimsmCRPRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmRouter*, *pimsmServer*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmLearnedJoinState

pimsmLearnedJoinState — accesses learned information associated with a PIM-SM source.

SYNOPSIS

pimsmLearnedJoinState *subcommand options*

DESCRIPTION

The *pimsmLearnedJoinState* command makes available learned join state information from the simulated source in the current *pimsmSource* command. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

numGroupsPerSource

(Read-only) The number of groups received per source address.

numSources

(Read-only) The number of sources in the learned Join state.

COMMANDS

The **pimsmLearnedJoinState** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmLearnedJoinState cget

Returns the list of options for the *pimsmLearnedJoinState* command.

pimsmLearnedJoinState receivedAllJoins

Returns TCL_OK if the Join messages that have been received include all groups for all source addresses.

pimsmLearnedJoinState receivedAllJoinsForGroup *groupIpAddress*

Returns TCL_OK if the Join messages that have been received for a specific group include all source addresses.

pimsmLearnedJoinState receivedAllJoinsForSource *sourceIpAddress*

Returns TCL_OK if the Join messages that have been received for a specific source include all group addresses.

pimsmLearnedJoinState receivedJoin *sourceIpAddress groupIpAddress*

Returns TCL_OK if the Join messages that have been received include the specific source and group address.

pimsmLearnedJoinState setDefault

Sets the options to default values.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedDataMDt*, *pimsmRouter*, *pimsmServer*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmMdtLearnedJoinState

pimsmMdtLearnedJoinState — accesses learned information associated with a PIM-SM Data MDT Range.

SYNOPSIS

pimsmMdtLearnedJoinState *subcommand options*

DESCRIPTION

The *pimsmMdtLearnedJoinState* command makes available learned Data MDT state information associated with the current *pimsmDataMdtRange* command. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

numGroupsPerSource

(Read-only) The number of groups received per source address.

numSources

(Read-only) The number of sources associated with the *pimsmMdtLearnedJoinState*.

COMMANDS

The **pimsmMdtLearnedJoinState** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmLearnedDataMdt cget

Returns the list of options available for the *pimsmMdtLearnedJoinState* command.

pimsmLearnedDataMdt getAllMdtJoins

Accesses the learned states for all Joins associated with the PIM-SM Data MDT Range.

pimsmLearnedDataMdt getMdtJoin

Accesses the learned states for a Join associated with the PIM-SM Data MDT Range.

pimsmLearnedDataMdt setDefault

Sets the options to default values.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmRouter*, *pimsmServer*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmRouter

pimsmRouter — configures a PIM-SM router.

SYNOPSIS

pimsmRouter *subcommand options*

DESCRIPTION

The *pimsmRouter* command represents a simulated router. In addition to some identifying options, it holds a single list for the router:

- Interfaces — router interface, constructed in the *pimsmInterface* command.

Routers defined in this command are added to an *pimsmServer* using the *pimsmServer addRouter* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on PIM-SM testing with Ixia equipment. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

dataMdtInterval

The time interval, in seconds, between transmissions of Data MDT Join TLV messages by the source PE Router. *(default = 60)*

dataTimeOut

The Data MDT hold time, in seconds. If a PE router connected to a receiver does not receive a Data MDT Join TLV message within this time period, it will leave the Data MDT group. *(default = 180)*

drPriority

The Designated Router (DR) priority assigned to this simulated router. This value is used in the election of the DR, and is included in *Hello* messages sent to neighbors. The larger the DR value, the higher the priority. *(default = 1)*

enable true / false

Enables the use of this router in the simulated PIM-SM network. *(default = false)*

joinPruneHoldTime

The period, in seconds, during which a router receiving a Join/Prune message must keep the state alive. The default is 3 times the Join/Prune interval. If this value is 0xffff, then the timeout is infinite and if this value is 0, the timeout is immediate. *(default = 180)*

joinPruneInterval

The length of time, in seconds, between transmissions of Join/Prune messages. *(default = 60)*

routerId

The ID of the router, in IPv4 format. *(default = 0.0.0.1)*

COMMANDS

The **pimsmRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmRouter addInterface *interfaceLocalId*

Adds the router interface described in the *pimsmInterface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *interfaceLocalId*. Specific errors are:

- The parameters in *pimsmInterface* are invalid.
- A router with this *interfaceLocalId* exists already in the list.

pimsmRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **pimsmRouter** command.

pimsmRouter clearAllInterfaces

Deletes all of the router interfaces.

pimsmRouter config *option value*

Modify the configuration options of the pimsmRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for pimsmRouter.

pimsmRouter delInterface *interfaceLocalId*

Deletes the router interface with an identifier of *interfaceLocalId*. Specific errors are:

- No router with this *interfaceLocalId* exists in the list.

pimsmRouter getFirstInterface

Access the first interface in the list. The results may be accessed using the *pimsmInterface* command. Specific errors are:

- pimsmRouter select has not been called.
- There are no interfaces in the list.

pimsmRouter getInterface *interfaceLocalId*

Accesses the interface's entry in the list with an identifier of *interfaceLocalId*. The router interface is accessed in the *pimsmInterface* command. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

pimsmRouter getNextInterface

Access the next interface in the list. The results may be accessed using the *pimsmInterface* command. Specific errors are:

- pimsmRouter getFirstInterface has not been called.
- There is no more interfaces in the list.

pimsmRouter setDefault

Sets default values for all configuration options.

pimsmRouter setInterface *[interfaceLocalId]*

Sets the values for the interface's entry in the list with an identifier of *interfaceLocalId* based on changes made through the *pimsmInterface* command. *interfaceLocalId* may only be omitted if *getFirstInterface* and *getNextInterface* were used to select the interface, in which case the currently selected interface is set. This command can be used to change a running configuration and must be followed by an *pimsmServer* *write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmLearnedDataMdt*, *pimsmServer*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmServer

pimsmServer — accesses the PIM-SM component of the protocol server for a particular port.

SYNOPSIS

pimsmServer *subcommand options*

DESCRIPTION

The *pimsmServer* command is necessary in order to access the PIM-SM protocol server for a particular port. The *select* subcommand **must** be used before all other PIM-SM commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on PIM-SM testing with Ixia equipment. Refer to *PIM-SM* for an overview.

STANDARD OPTIONS

bsmFramePerInterval

If *enableRateControl* is *true*, this is the number of BSM messages that will be transmitted every *interval* milliseconds. *(default = 0)*

crpFramePerInterval

If *enableRateControl* is *true*, this is the number of CRP messages that will be transmitted every *interval* milliseconds. *(default = 0)*

enableRateControl

If *true*, enables the use of the *messagesPerInterval*and *interval* options to control the rate of PIMSM generated messages. *(default = false)*

interval

If *enableRateControl* is *true*, this is the interval, expressed in milliseconds, over which *messagesPerInterval* messages will be sent. *(default = 0)*

joinPruneMessagesPer Interval

If *enableRateControl* is *true*, this is the number of join/prune messages that will be transmitted every *interval* milliseconds. *(default = 0)*

registerStopMessages PerInterval

If *enableRateControl* is *true*, this is the number of register stop messages that will be transmitted every *interval* milliseconds. *(default = 0)*

sourceMessagesPer Interval

If *enableRateControl* is *true*, this is the number of source messages that will be transmitted every *interval* milliseconds. *(default = 0)*

denyGrePimIpPrefix

Ixia will reject all GRE-PIM packets whose outer source IP address falls within this specified network prefix.

STANDARD OPTIONS

messagesPerInterval

If *enableRateControl* is *true*, this is the number of messages that will be transmitted every *interval* milliseconds.

COMMANDS

The **pimsmServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmServer addRouter *routerLocalId*

Adds the PIM-SM router described in the *pimsmRouter* command to the list of routers associated with the port. The router's entry in the list is given an identifier of *routerLocalId*. Specific errors are:

- pimsmServer select has not been called.
- The router parameters in *pimsmRouter* are invalid.
- A router with this *routerLocalId* exists already in the list.

pimsmServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *pimsmServer* command.

pimsmServer clearAllRouters

Deletes all the PIM-SM routers in the list. Specific errors are:

- pimsmServer select has not been called.
- There is no router with this *routerLocalId* in the list.

pimsmServer config *option value*

Modify the configuration options of the protocolServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for protocolServer.

pimsmServer delRouter *routerLocalId*

Deletes the PIM-SM router described that has an identifier of *routerLocalId*. Specific errors are:

- pimsmServer select has not been called.
- There is no router with this *routerLocalId* in the list.

pimsmServer get

Gets the current configuration of the protocol server for the last selected port from its hardware. Call this command before calling *pimsmServer* cget *option value* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.

pimsmServer getFirstRouter

Access the first PIM-SM router in the list*.* The results may be accessed using the *pimsmRouter* command. Specific errors are:

- pimsmServer select has not been called.
- There are no routers in the list.

pimsmServer getNextRouter

Access the next PIM-SM router in the list. The results may be accessed using the *pimsmRouter* command. Specific errors are:

- pimsmServer select has not been called.
- pimsmServer getFirstRouter has not been called.
- There is no more routers in the list.

pimsmServer getRouter *routerLocalId*

Access the PIM-SM router with an identifier of *routerLocalId.* The results may be accessed using the *pimsmRouter* command. Specific errors are:

- pimsmServer select has not been called.
- There is no router with this *routerLocalId* in the list.

pimsmServer select *chasID cardID portID*

Accesses the PIM-SM component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The PIM-SM protocol package has not been installed.
- Invalid port specified.

pimsmServer set

Sets the current configuration of the protocol server on the most recently selected port to its hardware. Call this command before calling *pimsmServer* cget *option value* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- The configured parameters are not valid for this port.

pimsmServer setDefault

Sets default values for all configuration options.

pimsmServer setRouter *[routerLocalId]*

Sets the values for the router's entry in the list with an identifier of *routerLocalId* based on changes made through the *pimsmRouter* command. *routerLocalId* may only be omitted if *getFirstRouter* and *getNextRouter* were used to select the router, in which case the currently selected router is set. This command should be used to change a running configuration and must be followed by an *pimsmServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *routerLocalId* does not exist in the list.
- Argument is omitted and no router is currently selected.

pimsmServer write

Sends any changes made with *pimsmRouter* *setInterface,* or *pimsmServer setRouter* to the protocol server for immediate application. This command **must** be used after those mentioned above in order for their changes to have an effect.

EXAMPLES

```
package req IxTclHal

chassis add astro

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts “Could not connect to $host”

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {

ixPuts $::ixErrorInfo

return 1

}

# Get the chassis ID to use in port lists

set ch [ixGetChassisID $host]

set card 12

set port 1
set portList [list [list $ch $card $port]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
# Example which configures the PIM-SM protocol and retreives the
# configuration using IDs.
# In this method each object is added by ID and the same ID is used
# in get/set in the reverse order of addition.
# These IDs don't appear in the IxExplorer GUI and if the wish
console
# is closed, the configuration cannot be retreived by ID again.
# Make sure to be consistent when using IDs. If IDs are in use, do
not
# mix them with calls to getFirst/GetNext
port setFactoryDefaults $ch $card $port
pimsmServer select $ch $card $port
pimsmServer clearAllRouters
# configure a join/prune
pimsmJoinPrune config -enable true
# Add join/prune to an imterface
if [pimsmInterface addJoinsPrune joinPrune1] {
logMsg "Error in pimsmInterface addJoinsPrune"
}
# Configure a source
pimsmSource config -enable true
# Add the source to the interface
if [pimsmInterface addSource source1] {
```

```
logMsg "Error in pimsmInterface addSource"
}
# Add the interface to the router
pimsmInterface config -enable true
if [pimsmRouter addInterface interface1] {
logMsg "Error in pimsmRouter addInterface"
}
# Add the router to the server
pimsmRouter config -enable true
if [pimsmServer addRouter router1] {
logMsg "Error in pimsmServer addRouter"
}
# You can retreive the configured objects by using the get method
by name
# or getFirst/getNext. Make sure to be consistent in using these
methods.
# Either get by name or getFirst/getNext should be used
exclusively on
# a particular object.
# Select the port to retreive from
pimsmServer select 1 2 1
pimsmServer getRouter router1
pimsmRouter getInterface interface1
pimsmInterface getJoinPrune joinsPrune1
pimsmInterface getSource source1
# The set methods can be used to make modifications on the fly
# The set methods take an optional name as an argument, which
should
# be omitted when using getFirst/getNext. If using get by name,
then you
# should also use set by name
# Get / set by name
pimsmServer select 1 2 1
pimsmServer getRouter router1
```

```
pimsmRouter config -enable false
if [pimsmServer setRouter router1] {
logMsg "Error in pimsmServer setRouter"
}
# getFirst/getNext/set
pimsmServer select 1 2 1
pimsmServer getFirstRouter
pimsmRouter config -enable false
if [pimsmServer setRouter] {
logMsg "Error in pimsmServer setRouter"
}
# Getting the learned join state for a particular source
pimsmSource requestLearnedJoinState
set timer 0
while { $timer < 10} {
if {[pimsmSource getLearnedJoinState]} {
incr timer
} else {
break;
}
}
if {$timer == 10} {
logMsg "Error getting the learned join state"
} else {
if {![pimsmLearnedJoinState receivedAllJoins]} {
logMsg "Received all joins."
showCmd pimsmLearnedJoinState
}
}
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
```

```
if [isUNIX] {

ixDisconnectTclServer $host

}
```

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmLearnedDataMdt*, *pimsmRouter*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmSource*

NAME - pimsmSource

pimsmSource — sets up the parameters associated with a PIM-SM source.

SYNOPSIS

pimsmSource *subcommand options*

DESCRIPTION

The *pimsmSource* command describes an source of addresses that a PIM-SM interface will register with a Rendezvous Point (RP). Route ranges are added into *pimsmInterface* lists using the *pimsmInterface* *addSource* command. Refer to *PIM-SM* for an overview of this command.

STANDARD OPTIONS

activationInterval

(for Data MDTs only) The time period, in seconds, after which the sources will start sending packets to support teh switchover from the Default MDT to the Data MDT. *(default = 60)*

enable true / false

Enables the use of this route range for the simulated interface. *(default = false)*

enableDiscardJoin States true / false

If *true,* the learned join states sent by the RP (DUT) in response to this specific Register message will be discarded. *(default = true)*

enableSendNullRegAt Beginning true | false

If *true*, an initial null registration will be sent at emulation startup. *(default = false)*

groupAddress

The first IPv4 multicast group address in the range of group addresses included in the Register message. *(default = 255.0.0.0)*

groupAddressCount

The number of group addresses, starting with *groupAddress* to be included in the Register message. *(default = 1)*

rpAddress

The IP address of the Rendezvous Point (RP) router, the root of the RP shared multicast distribution tree (RPT). *(default = 0.0.0.0)*

sourceAddress

The first source address to be included in the Register messages*. (default = 0.0.0.1)*

sourceAddressCount

The number of register source addresses to be included. *(default = 1)*

sourceGroupMapping

Sets the type of mapping that occurs when routes are advertised. This only applies for (S, G) and switchover types for MGR and is meaningful for RR. One of:

Option	Value	Usage
pimsmMappingFullyMeshed	0	*(default)* All sources to all groups.
pimsmMappingOneToOne	1	One source to one group.

txIterationGap

The gap, in milliseconds, between periodically transmitted Register messages. *(default = 5,000)*

udpDestinationPort

The number of UDP destination ports in the receiving multicast group. *(default = 3,000)*

udpSourcesPort

The number of UDP source ports sending encapsulated UDP packets to multicast groups via Register messages to the RP. *(default = 3,000)*

COMMANDS

The **pimsmSource** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

pimsmSource cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *pimsmSource* command.

pimsmSource config *option value*

Modify the configuration options of the pimsmSource. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *pimsmSource*.

pimsmSource getLearnedJoinState

This subcommand should be called after *requestLearnedJoinState*. It must be called until it returns TCL_OK, usually with a wait between calls. The actual join state may be retrieved by using the *pimsmLearnedJoinState* command. Specific errors are:

- The list has not been completely retrieved yet.

pimsmSource requestLearnedJoinState

Requests that the learned join states for this register be retrieved from the protocol server. The *getLearnedJoinState* subcommand must be called after this subcommand to determine when the complete list of routes has been retrieved.

pimsmSource setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *pimsmServer* and *pimsmDataMdtRange*.

SEE ALSO

pimsmInterface, *pimsmInterfaceLearnedInfo*, *pimsmLearnedJoinState*, *pimsmLearnedDataMdt*, *pimsmRouter*, *pimsmJoinPrune*, *pimsmDataMdtRange*, *pimsmMdtLearnedJoinState*, *pimsmServer*

NAME - portGroup

portGroup — sets up a group of ports.

SYNOPSIS

portGroup *subcommand options*

DESCRIPTION

This command allows the user to set up an autonomous group of ports on which to perform an action or command, such as take ownership, start transmit, capture, or clearing statistics, to name a few. A port group must be created and the desired ports (or port) added to it in order to execute the selected action or command. When the port group is no longer needed, it should be destroyed.

NOTE: This command is a duplicate of the same command in the Ixia Tcl Development Guide, except that a different set of commands is listed for the setCommand subcommand.

STANDARD OPTIONS

lastTimeStamp

Read-only. 64-bit value. The relative time of transmit for all the ports in the port group.

COMMANDS

The **portGroup** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

portGroup add *groupID chasID cardID portID*

Adds this port to a group with ID *groupID.* Specific errors are:

- No connection to a chassis.
- The groupID port group does not exist.

portGroup canUse *groupID*

Verifies whether all the ports in this group can be used by the current logged in user. Specific errors are:

- No connection to a chassis.
- The groupID port group does not exist.

portGroup cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **portGroup** command.

portGroup clearScheduledTransmitTime *groupID*

Clears the scheduled transmit time associated with a group of ports. See *setScheduledTransmitTime*. Specific errors are:

- No connection to a chassis.
- The *groupID* port group does not exist.

portGroup config *option value*

Modify the configuration options of all the ports. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for portGroup. (There are currently no configurable options for **portGroup** and therefore no use for this command).

portGroup create *groupID*

Creates a port group and assigns it the ID *groupID*. Specific errors are:

- The groupID port group already exists.

portGroup del *groupID chasID cardID portID*

Deletes this port from the group with ID *groupID*. Specific errors are:

- No connection to a chassis.
- The groupID port group does not exist.

portGroup destroy *groupID*

Destroys the port group with ID *groupID*. Specific errors are:

- The groupID port group does not exist.

portGroup setCommand *groupID cmd*

Performs the action or command *cmd* specified on all ports in the group with ID *groupID*. Note that some of the command values previously listed in this table have been moved to the *IxRouter Tcl Development Guide. cmd* may be one of the following:

Option	Value	Usage
transmitIgmpJoin startIgmp	54	transmit IGMP join messages
transmitIgmpLeave	55	transmit IGMP join messages
startBgp4	63	start BGP operations
stopBgp4	64	stop BGP operations
startOspf	65	start OSPF operations
stopOspf	66	stop OSPF operations
startIsis	76	start ISIS operations
stopIsis	77	stop ISIS operations
startRsvp	82	start RSVP operations
stopRsvp	83	stop RSVP operations
startRip	89	start RIP operations

Option	Value	Usage
stopRip	90	stop RIP operations
startLdp	95	start LDP operations
stopLdp	96	stop LDP operations
startRipng	97	start RIPng operations
stopRipng	98	stop RIPng operations
startPimsm	111	start Pimsm operations
stopPimsm	112	stop Pimsm operations
startMld	113	start Mld operations
stopMld	114	stop Mld operations
stopIgmp	117	stop Igmp operations
startOspfV3	115	start OspfV3 operations
stopOspfV3	116	stop OspfV3 operations

Specific errors are:

- No connection to a chassis.
- One or more ports in the port group are being used by another user.
- One or more ports in the port group are invalid.
- Network error between the client and chassis.

portGroup setDefault

Sets default values for all configuration options.

portGroup setScheduledTransmitTime *groupID time*

This feature only applies to ports which support the *portFeatureScheduledTxDuration* feature (see *portisValidFeature).* This subcommand sets the transmit time duration associated with the group of ports. *time* is expressed in seconds. When a scheduled transmit time is set, and a *portGroup setCommand <group startTransmit* is issued, the ports in the port group will transmit until their streams are exhausted or the specified *time* has elapsed, whichever comes first. This value may be cleared with the *clearScheduledTransmitTime* subcommand to this command. Specific errors are:

- No connection to a chassis.
- The *groupID* port group does not exist.
- Invalid *time* value.

portGroup write *groupID [writeProtocolServer]*

Commits port properties information such as speed, duplex mode, and autonegotiation in hardware. If *writeProtocolServer* is true, then the protocol server will be stopped and all applicable objects written to it. Otherwise, the protocol server will not be affected. Specific errors are:

- No connection to a chassis.
- The port group specified by groupID has not been created.

- One or more ports in the port group are being used by another user.
- Network error between the client and chassis.

portGroup writeConfig *groupID [writeProtocolServer]*

Configures streams, filter and capture parameters of all ports in the group except the port properties such as speed, duplex mode, and autonegotiation. If *writeProtocolServer* is true, then the protocol server will be stopped and all applicable objects written to it. Otherwise, the protocol server will not be affected. Specific errors are:

- No connection to a chassis.
- The port group specified by groupID has not been created.
- One or more ports in the port group are being used by another user.
- Network error between the client and chassis.

DEPRECATED COMMANDS

portGroup get *groupID objectID*

Gets the type of object designated by *objectID* for a list of ports. The only defined value for *objectID* is *usbConfig (0)*, which must be applied to USB configured ports. Specific errors are:

- Invalid *objectID.*
- The *groupID* port group does not exist.

EXAMPLES

SEE ALSO

NAME - protocolServer

protocolServer — configures the protocol server services for a port.

SYNOPSIS

protocolServer *subcommand options*

DESCRIPTION

This command allows the user to select a protocol service and configure that service. The protocol server is used to enable and disable the ability to respond to protocol requests received from DUTs.

STANDARD OPTIONS

enableArpResponse true / false

Enables ARP response. *(default = false)*

enableBfdService true / false

Enables BFD service. *(default = false)*

enableBgp4Create Interface true / false

Enables the automatic creation of BGP interfaces. This is useful for programs writing for version 3.55 and earlier releases. It causes the interface table to be automatically created from IP table configurations. Programs written to use the *interfaceTable* will run with this option set, but slowly. *(default = true)*

enableBgp4Service true / false

Enables BGP4 service. *(default = false)*

enableCfmService true / false

Enables CFM service. *(default = false)*

enableEigrpService true / false

Enables EIGRP service. *(default = false)*

enableIgmpCreate Interface true / false

Enables the automatic creation of IGMP interfaces. This is useful for programs writting for version 3.55 and earlier releases. It causes the interface table to be automatically created from IP table configurations. Programs written to use the *interfaceEntry* will run with this option set, but slowly. *(default = true)*

enableIgmpQuery Response true / false

Enables IGMP query response. *(default = false)*

enableIsisCreate Interface true / false

Enables the automatic creation of ISIS interfaces. This is useful for programs writting for version 3.55 and earlier releases. It causes the interface table to be automatically created from IP table configurations. Programs written to use the *interfaceTable* will run with this option set, but slowly. *(default = true)*

enableIsisService true / false

Enables ISIS service. *(default = false)*

enableLacpService true / false

Enables LACP service. *(default = false)*

enableLdpService true / false

Enables LDP service. *(default = false)*

enableMldService true / false

Enables MLD service. *(default = false)*

enableOspfCreate Interface true / false

Enable the automatic creation of OSPF interfaces. This is useful for programs writting for version 3.55 and earlier releases. It causes the interface table to be automatically created from IP table configurations. Programs written to use the *interfaceTable* will run with this option set, but slowly. *(default = true)*

enableOspfService true / false

Enables OSPF service. *(default = false)*

enableOspfV3Service true / false

Enables OSPFv3 service. *(default = false)*

enablePimsmService true / false

Enables PIM-SM service. *(default = false)*

enablePingResponse true / false

Enables PING response. *(default = false)*

enableRipCreate Interface true / false

Enables the automatic creation of RIP interfaces. This is useful for programs writting for version 3.55 and earlier releases. It causes the interface table to be automatically created from IP table configurations. Programs written to use the *interfaceTable* will run with this option set, but slowly. *(default = true)*

enableRipService true / false

Enables RIP service. *(default = false)*

enableRipngService true / false

Enables RIPng service. *(default = false)*

enableRsvpCreate Interface true / false

Enables the automatic creation of RSVP interfaces. This is useful for programs writting for version 3.55 and earlier releases. It causes the interface table to be automatically created from IP table configurations. Programs written to use the *interfaceTable* will run with this option set, but slowly. *(default = true)*

enableRsvpService true / false

Enables RSVP service. *(default = false)*

enableStpService true / false

Enables STP service. *(default = false)*

enableMplsTpService true / false

Enables MPLS-TP service. *(default = false)*

DEPRECATED STANDARD OPTIONS

arpServerEnable

Enable the ARP response engine to send out ARP responses when ARP requests are received on a port.

count

The total number of addresses in the address table.

IpAddress

Initial IP address for the protocol server address table.

MacAddress

Initial MAC address for the protocol server address table.

mapType

The type of mac/ip address mapping selected. Options include:

Option	Value	Usage
oneIpToOneMAC	0	Exactly one MAC address will be associated with each IP address *(default)*
manyIpToOneMAC	1	Only one MAC address will be associated with multiple IP addresses

pingServerEnable *true/false*

Enable the PING response engine to send out PING responses when PING requests are received on a port. *(default = false)*

rate

(default = -1)

repeatCount

(default = -1)

COMMANDS

The **protocolServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

protocolServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **protocolServer** command.

protocolServer config *option value*

Modify the configuration options of the protocolServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for protocolServer.

protocolServer get *chasID cardID portID*

Gets the current configuration of the protocol server on port with ID *portID* on card *cardID*, chassis *chasID*. from its hardware. Call this command before calling **protocolServer** cget *option value* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.
- Invalid port number.

protocolServer set *chasID cardID portID*

Sets the current configuration of the protocol server on port with ID *portID* on card *cardID*, chassis *chasID*. from its hardware. Call this command before calling **protocolServer** cget *option value* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- The configured parameters are not valid for this port.

protocolServer setDefault

Sets default values for all configuration options.

protocolServer **write** *chasID cardID portID*

Writes or commits the changes in IxHAL to hardware the protocol server configuration for each port with ID *portID* on card *cardID*, chassis *chasID.* Before using this command, use the **protocolServer** *set* command to configure the port related in IxHAL. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- Network error between the client and chassis.

EXAMPLES

See examples under *arpServer*, *bgp4Server*, *cfmServer*, *igmpServer*, *isisServer*, *ldpServer*, *mldServer*, *ospfServer*, *ospfV3Server*, *ripServer*, *and* *rsvpServer*

SEE ALSO

pimsmInterfaceLearnedCRPInfo

NAME - ripInterfaceRouter

ripInterfaceRouter — configures an RIP router.

SYNOPSIS

ripInterfaceRouter *subcommand options*

DESCRIPTION

The *ripInterfaceRouter* command represents a simulated router. In addition to some identifying options, it holds a list for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *ripRouteRange* command.

Routers defined in this command are added to an *ripServer* using the *ripServer addRouter* command. Refer to *ripInterfaceRouter* for an overview of this command.

STANDARD OPTIONS

authorizationPassword

If *enableAuthorization* is true, then this is the password to be included in the authentication part of the RIP messages. *(default = "")*

enableAuthorization true / false

Indicates whether authorization is included in update messages. *(default = false)*

enableRouter true / false

Enables or disables the simulated router. *(default = false)*

protocolInterface Description

The *description* option associated with an *interfaceEntry* when it was created. The IP address and mask are read from the interface entry. *(default = "")*

receiveType

Filters the version of messages this router will receive. One of:

Option	Value	Usage
ripReceiveVersion1	1	RIP Version 1 messages only.
ripReceiveVersion2	2	*(default)* RIP Version 2 messages only.
ripReceiveVersion1and2	3	Both RIP version messages.

responseMode One of:

Option	Value	Usage
ripDefault	0	
ripSplitHorizon	1	

Option	Value	Usage
ripPoisonReverse	2	
ripSplitHorizonSpaceSaver	3	(default)
ripSilent	4	

The current implementation uses *ripSplitHorizonSpaceSaver* as its update mode regardless of the setting.

sendType

The method for sending RIP packets. One of:

Option	Value	Usage
ripMulticast	0	*(default)* sends Version 2 packets via multicast
ripBroadcastV1	1	sends V1 packets via broadcast.
ripBroadcastV2	2	sends V2 packets via broadcast.

updateInterval

The time, in seconds, between transmitted update messages. *(default = 30)*

updateIntervalOffset

A random percentage of this time value, expressed in seconds,which will be added or subtracted from the update interval. *(default = 5)*

COMMANDS

The **ripInterfaceRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ripInterfaceRouter addRouteRange *routeRangeId*

Adds the route range described in the *ripRouteRange* command to the list of route ranges associated with the router. The range's entry in the list is given an identifier of *routeRangeId*. Specific errors are:

- The parameters in *ripRouteRange* are invalid.
- A router with this *routeRangeId* exists already in the list.

ripInterfaceRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ripInterfaceRouter** command.

ripInterfaceRouter clearAllRouteRange

Deletes all of the route ranges.

ripInterfaceRouter config *option value*

Modify the configuration options of the ripInterfaceRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ripInterfaceRouter.

ripInterfaceRouter delRouteRange *routeRangeId*

Deletes the route range with an identifier of *routeRangeId*. Specific errors are:

- No router with this *routeRangeId* exists in the list.

ripInterfaceRouter getFirstRouteRange

Access the first route range in the list. The results may be accessed using the *ripRouteRange* command. Specific errors are:

- There are no route ranges in the list.

ripInterfaceRouter getNextRouteRange

Access the next route range in the list. The results may be accessed using the *ripRouteRange* command. Specific errors are:

- ripInterfaceRouter getFirstRouteRange has not been called.
- There is no more route ranges in the list.

ripInterfaceRouter getRouteRange *routeRangeId*

Accesses the range's entry in the list with an identifier of *routeRangeId*. The router range is accessed in the *ripRouterRange* command. Specific errors are:

- A router with this *routeRangeId* does not exist in the list.

ripInterfaceRouter setDefault

Sets default values for all configuration options.

ripInterfaceRouter setRouteRange *routeRangeId*

Sets the values for the route range's entry in the list with an identifier of *routeRangeId* based on changes made through the *ripRouteRange* command. This command should be used to change a running configuration and must be followed by a *ripServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *routeRangeId* does not exist in the list.

EXAMPLES

See examples under *ripServer*.

SEE ALSO

ripServer, *ripRouteRange*

NAME - ripngInterface

ripngInterface — configures an RIPng router interface.

SYNOPSIS

ripngInterface *subcommand options*

DESCRIPTION

The *ripngInterface* command represents an interface on a simulated RIPng router. A RIPng interface uses an interface defined with the *interfaceEntry* command. Refer to RIPng for an overview of this command.

STANDARD OPTIONS

enable true | false

Enables or disables the use of this interface. *(default = false)*

interfaceMetric

The value of the metric associated with this interface. If the value of the *enableInterfaceMetric* option of the ripngRouter command is *true*, then this value is added to the metric in the routing table before transmitting updates through this interface. *(default = 0)*

protocolInterface Description

The *description* name used when a *interfaceEntry* was created. *(default = "")*

responseMode

Indicates how the interface will respond to requests for routing updates. These options are only meaningful if the *receiveType* option of the ripngRouter command is *ripngStore*. The options are:

Option	Value	Usage
ripngSplitHorizon	0	*(default)* Do not include routes received from a router back to that router.
ripngNoSplitHorizon	1	Repeat all received routes back to all routers.
ripngPoisonReverse	2	Repeat routes received from a router back to that router, but with a metric of 16 indicating an unavailable route.

COMMANDS

The **ripngInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ripngInterface cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ripngInterface** command.

ripngInterface config *option value*

Modify the configuration options of the ripngInterface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ripngInterface.

ripngInterface setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ripngServer*.

SEE ALSO

ripngRouter, *ripngRouteRange*, *ripngServer*

NAME - ripngRouter

ripngRouter — configures an RIPng router.

SYNOPSIS

ripngRouter *subcommand options*

DESCRIPTION

The *ripngRouter* command represents a simulated router. In addition to some identifying options, it holds two lists for the router:

- Route ranges — routes to be advertised by the simulated router, constructed in the *ripngRouteRange* command.
- Interfaces — protocol interfaces that received and send Ripng messages, constructed in the *ripngInterface* command.

Routers defined in this command are added to an *ripngServer* using the *ripngServer* *addRouter* command.Refer to *RIPng* for an overview of this command.

STANDARD OPTIONS

enable true | false

If *true*, the router is enabled. *(default = false)*

enableInterfaceMetric true | false

If *true*, then the value found in the *ripngInterface*'s *interfaceMetric* will be added to the metric associated with each advertisement. This allows the metrics transmitted by multiple interfaces to differ. *(default = false)*

receiveType

Determines what the simulated router will do with received RIPng updates. One of:

Option	Value	Usage
ripngIgnore	0	(default) Received updates are ignored.
ripngStore	1	Received updates are held and re-advertised.

routerId

The ID associated with the simulated router. *(default = 1)*

updateInterval

The time, in seconds, between transmitted update messages. *(default = 30)*

updateIntervalOffset

A random percentage of this time value, expressed in seconds,which will be added or subtracted from the update interval. *(default = 5)*

COMMANDS

The **ripngRouter** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ripngRouter addInterface *interfaceId*

Adds the interface described in the *ripngInterface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *interfaceId*.Specific errors are:

- The parameters in *ripngInterface* are invalid.
- A router with this *interfaceId* exists already in the list.

ripngRouter addRouteRange *routeRangeId*

Adds the route range described in the *ripngRouteRange* command to the list of route ranges associated with the router. The range's entry in the list is given an identifier of *routeRangeId*. Specific errors are:

- The parameters in *ripRouteRange* are invalid.
- A router with this *routeRangeId* exists already in the list.

ripngRouter cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ripngRouter** command.

ripngRouter clearAllInterfaces

Deletes all of the interfaces.

ripngRouter clearAllRouteRanges

Deletes all of the route ranges.

ripngRouter config *option value*

Modify the configuration options of the ripngRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ripngRouter.

ripngRouter delInterface *interfaceId*

Deletes the interface with an identifier of *interfaceId*. Specific errors are:

- No router with this *interfaceId* exists in the list.

ripngRouter delRouteRange *routeRangeId*

Deletes the route range with an identifier of *routeRangeId*. Specific errors are:

- No router with this *routeRangeId* exists in the list.

ripngRouter getFirstInterface

Access the first interface in the list. The results may be accessed using the *ripngInterface* command. Specific errors are:

- There are no interfaces in the list.

ripngRouter getFirstRouteRange

Access the first route range in the list. The results may be accessed using the *ripngRouteRange* command. Specific errors are:

- There are no route ranges in the list.

ripngRouter getInterface *interfaceId*

Accesses the interface's entry in the list with an identifier of *interfaceId*. The interface is accessed in the *ripngInterface* command. Specific errors are:

- A router with this *interfaceId* does not exist in the list.

ripngRouter getNextInterface

Access the next interface in the list. The results may be accessed using the *ripngInterface* command. *getFirstInterface* must be called before this call. Specific errors are:

- ripngRouter getFirstInterface has not been called.
- There is no more interfaces in the list.

ripngRouter getNextRouteRange

Access the next route range in the list. The results may be accessed using the *ripngRouteRange* command. *getFirstRouteRange* must be called before this call. Specific errors are:

- ripngRouter getFirstRouteRange has not been called.
- There is no more route ranges in the list.

ripngRouter getRouteRange *routeRangeId*

Accesses the range's entry in the list with an identifier of *routeRangeId*. The router range is accessed in the *ripngRouteRange* command. Specific errors are:

- A router with this *routeRangeId* does not exist in the list.

ripngRouter setDefault

Sets default values for all configuration options.

ripngRouter setInterface *interfaceId*

Sets the enable for the interface's entry in the list with an identifier of *interfaceId* based on changes made through the *ripngInterface* command. This command should be used to change a running configuration and must be followed by a *ripngServer* *write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *interfaceId* does not exist in the list.
- The port is owned by another user.
- Invalid ripngInterface configuration.

ripngRouter setRouteRange *routeRangeId*

Sets the enable for the route range's entry in the list with an identifier of *routeRangeId* based on changes made through the *ripngRouteRange* command. This command should be used to change a running configuration and must be followed by a *ripngServer* *write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *routeRangeId* does not exist in the list.
- The port is owned by another user.
- Invalid ripngRouteRange configuration.

EXAMPLES

See examples under *ripngServer*.

SEE ALSO

ripngInterface, *ripngRouteRange*, *ripngServer*

NAME - ripngRouteRange

ripngRouteRange — sets up the parameters associated with an RIPng route range.

SYNOPSIS

ripngRouteRange *subcommand options*

DESCRIPTION

This command describes a set of routes. Route ranges are added into *ripngRouter* lists using the *ripngRouter* *addRouteRange* command. A router also holds two lists: route ranges defined in *ripngRouteRange* and interfaces defined in *ripngInterface*.

A number of routes are generated starting with *networkIpAddress.* A number of routes (*numRoutes*) are generated by incrementing the network part of the address by *step* as indicated by the *networkMaskWidth*.

Refer to *RIPng* for an overview of this command.

STANDARD OPTIONS

enable true | false

Enables the use of this route range for the simulated router. *(default = false)*

enableIncludeLoopback true | false

If *false,* then the loopback address is skipped when routes are generated. *(default = true)*

enableIncludeMulticast true | false

If *false,* then the multicast addresses are skipped when routes are generated. *(default = true)*

maskWidth

The network mask to be applied to the *networkIpAddress* to yield the non-host part of the address. A value of 0 means there is no subnet address. *(default = 64)*

metric

The total metric cost for these routes. The valid range is from 1 to 16 (inclusive). A value of 16 means that the destination is not reachable, and that route will be removed from service. *(default = 1)*

networkIpAddress

The network address to be used in creating this route range. *(default = 0:0:0:0:0:0:0:0)*

nextHop

The immediate next hop IP address on the way to the destination address. *(default = 0:0:0:0:0:0:0:0)*

numRoutes

The number of route ranges to generate for this route range. Each new route is generated by adding *step* to the network portion of the address. The loopback and/or multicast range may be skipped or included based on the setting of the *enableIncludeLoopback* and *enableIncludeMulticast* flags. *(default = 1)*

routeTag

A arbitrary value associated with the routes in this range. It is used to provide a means for distinguishing internal versus external RIP routes. *(default = 0)*

step

If *numRoutes* is greater than one, then this is the value added to the network portion of the IP address between values. *(default = 1)*

COMMANDS

The **ripngRouteRange** command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

ripngRouteRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ripngRouteRange** command.

ripngRouteRange config *option value*

Modify the configuration options of the ripngRouteRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ripngRouteRange.

ripngRouteRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ripngServer*.

SEE ALSO

ripngInterface, *ripngRouter*, *ripngServer*

NAME - ripngServer

ripngServer — accesses the RIPng component of the protocol server for a particular port.

SYNOPSIS

ripngServer *subcommand options*

DESCRIPTION

The *ripngServer* command is necessary in order to access the RIPng protocol server for a particular port. The *select* subcommand **must** be used before all other RIP commands. Refer to *RIPng* for an overview of this command.

The number of routes can be very large; the *numRoutes* and *timePeriod* options provide a means or limiting the rate at which route announcements are transmitted. If both are set to non-zero values, then *numRoutes* routes are transmitted every *timePeriod* milliseconds.

STANDARD OPTIONS

numRoutes

The number of routes to transmit every *timePeriod* milliseconds. A value of 0 disables this feature and transmits all routes immediately for all updates. *(default = 0)*

timePeriod

The time period to use for throttling updates, expressed in milliseconds. A value of 0 disables this feature and transmits all routes immediately for all updates. *(default = 0)*

COMMANDS

The **ripngServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ripngServer addRouter *routerId*

Adds the RIPng router described in the *ripngRouter* command to the list of routers associated with the port. Only one interface may currently be added. The router's entry in the list is given an identifier of *routerId*. Specific errors are:

- ripngServer select has not been called.
- The router parameters in *ripngRouter* are invalid.
- A router with this *routerId* exists already in the list.
- The port is owned by another user.

ripngServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the *ripngServer* command.

ripngServer clearAllRouters

Deletes all the RIPng routers in the list. Specific errors are:

- ripngServer select has not been called.

ripngServer config *option value*

Modify the configuration options of the *ripngServer*. If no *option* is specified, returns a list describing all of the available options for *ripngServer*.

ripngServer delRouter *routerId*

Deletes the RIPng router described that has an identifier of *routerId*. Specific errors are:

- ripngServer select has not been called.
- There is no router with this *routerId* in the list.
- The port is owned by another user.

ripngServer generateStreams *chasID cardID portID action*

Generate streams creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each enabled route range associated with each rip router; each stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

ripngServer getFirstRouter

Access the first RIPng router in the list. The results may be accessed using the *ripngRouter* command. Specific errors are:

- ripngServer select has not been called.
- There are no routers in the list.

ripngServer getNextRouter

Access the next RIPng router in the list*.* The results may be accessed using the *ripngRouter* command. Specific errors are:

- ripngServer select has not been called.
- ripngServer getFirstRouter has not been called.
- There is no more routers in the list.

ripngServer getRouter *routerId*

Access the RIPng router with an identifier of *routerId.* The results may be accessed using the *ripngRouter* command. Specific errors are:

- ripngServer select has not been called.
- There is no router with this *routerId* in the list.

ripngServer select *chasID cardID portID*

Accesses the RIPng component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The RIPng protocol package has not been installed.
- Invalid feature on selected port.

ripngServer setRouter *routerId*

Sets the enable for the router's entry in the list with an identifier of *routerId* based on changes made through the *ripngRouter* command. This command should be used to change a running configuration and must be followed by an *ripngServer write* command in order to send these changes to the protocol server. Specific errors are:

- ripngServer select has not been called.
- A router with this *routerId* does not exist in the list.

ripngServer write

Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card, and port*.* Before using this command, use the *ripngServer select* command to select the port.

EXAMPLES

```
package req IxTclHal

# Define parameters used by OSPF router

set host localhost

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {
```

```
if [ixConnectToTclServer $host] {
ixPuts "Could not connect to $host"
return 1
}
}
# Now connect to the chassis
if [ixConnectToChassis $host] {
ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
# Port is: card 4, port 1
set card 4
set port 1
set portList [list [list $chassis $card $port]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $portList] {
ixPuts $::ixErrorInfo
return 1
}
interfaceTable select $chassis $card $port
interfaceTable clearAllInterfaces
interfaceIpV6 setDefault
interfaceIpV6 config -maskWidth 64
interfaceIpV6 config -ipAddress {0:0:0:0:0:0:0:1}
interfaceEntry addItem addressTypeIpV6
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description {1 - 04:01 -
0:0:0:0:0:0:0:1/64}
interfaceEntry config -macAddress {00 00 09 79 B4 78}
interfaceTable addInterface
ripngServer select $chassis $card $port
ripngServer clearAllRouters
ripngInterface setDefault
ripngInterface config -enable true
ripngInterface config -protocolInterfaceDescription \
{1 - 04:01 - 0:0:0:0:0:0:0:1/64}
ripngRouter addInterface interface1
ripngRouteRange setDefault
ripngRouteRange config -enable true
ripngRouteRange config -networkIpAddress {0:0:0:0:1:0:0:0}
ripngRouteRange config -maskWidth 64
ripngRouteRange config -numRoutes 10
ripngRouteRange config -nextHop {0:10:0:0:0:0:0:0}
ripngRouter addRouteRange routeRange1
ripngRouter setDefault
ripngRouter config -routerId 1
ripngRouter config -receiveType ripngStore
ripngRouter config -enable true
ripngServer addRouter router1
ripngServer setDefault
ripngServer config -numRoutes 100
ripngServer config -timePeriod 10
ripngServer set
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enableRipngService true
protocolServer set $chassis $card $port
ixWritePortsToHardware portList
# And start RIPng on the port
ixStartRipng portList
```

```
# Disable routeRange1 while ripnp server is runnung.
# This is the same as removing the route range from router
ripngServer select $chassis $card $port
if [ripngServer getRouter router1] {
logMsg “Error getting router1”
}
if [ripngRouter getRouteRange routeRange1] {
logMsg “Error getting routeRange1”
}
# Disable the route range
ripngRouteRange config -enable false
if [ripngRouter setRouteRange routeRange1] {
logMsg “Error setting routeRange1”
}
if [ripngServer write] {
logMsg “Error writing ripServer”
}
# If you wanted to add a route range while ripng server is
running,
# -Configure it disabled before starting rip server and then
enable it
# Let go of the ports that we reserved
ixClearOwnership $portList
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

ripngInterface, ripngRouter, ripngRouteRange

NAME - ripRouteRange

ripRouteRange — sets up the parameters associated with an RIP route range.

SYNOPSIS

ripRouteRange *subcommand options*

DESCRIPTION

This command describes a set of routes. Route ranges are added into *ripInterfaceRouter* lists using the *ripInterfaceRouter addRouteRange* command. Refer to ripRouteRange for an overview of this command.

STANDARD OPTIONS

enableRouteRange

Enables the use of this route range for the simulated router. *(default = false)*

metric

The total metric cost for these routes. The valid range is from 1 to 16 (inclusive). A value of 16 means that the destination is not reachable, and that route will be removed from service. *(default = 1)*

networkIpAddress

The network address to be used in creating this route range. *(default = 0.0.0.0)*

networkMaskWidth

The network mask to be applied to the *networkIpAddress* to yield the non-host part of the address. A value of 0 means there is no subnet address. *(default = 24)*

nextHop

The immediate next hop IP address on the way to the destination address. *(default = 0.0.0.0)*

numberOfNetworks

The number of networks to be generated for this route range, based on the network address plus the network mask. *(default = 1)*

routeTag

A arbitrary value associated with the routes in this range. It is used to provide a means for distinguishing internal versus external RIP routes. *(default = 0)*

COMMANDS

The **ripRouteRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ripRouteRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ripRouteRange** command.

ripRouteRange config *option value*

Modify the configuration options of the ripRouteRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ripRouteRange.

ripRouteRange setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *ripServer*.

SEE ALSO

ripServer, *ripInterfaceRouter*

NAME - ripServer

ripServer — accesses the RIP component of the protocol server for a particular port.

SYNOPSIS

ripServer *subcommand options*

DESCRIPTION

The *ripServer* command is necessary in order to access the RIP protocol server for a particular port. The *select* subcommand **must** be used before all other RIP commands. Refer to ripServer59 for an overview.

STANDARD OPTIONS

none

COMMANDS

The **ripServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

ripServer addRouter *routerId*

Adds the RIP router described in the *ripInterfaceRouter* command to the list of routers associated with the port. The router's entry in the list is given an identifier of *routerId*. Specific errors are:

- ripServer select has not been called.
- The router parameters in *ripInterfaceRouter* are invalid.
- A router with this *routerId* exists already in the list.

ripServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **ripServer** command.

ripServer **clearAllRouter**

Deletes all the RIP routers in the list. Specific errors are:

- ripServer select has not been called.

ripServer config *option value*

Modify the configuration options of the ripServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for ripServer.

ripServer delRouter *routerId*

Deletes the RIP router described that has an identifier of *routerId*. Specific errors are:

- ripServer select has not been called.
- There is no router with this *routerId* in the list.

ripServer generateStreams *chasID cardID portID action*

Generate streams creates additional streams for the indicated port and replaces the port's current streams or adds it to the current streams depending on the *action* option:

Option	Value	Usage
protocolServerStreamReplace	0	Replace the port's current streams.
protocolServerStreamAppend	1	Add the streams to the port's current streams.

A separate stream is generated for each enabled route range associated with each rip router; each stream covers the count of IP addresses associated with the route range. The characteristics of the generated streams are:

- Each stream but the last is set to advance to the next stream; the last stream is set to return to the first stream.
- Ethernet II encapsulation is used.
- IPv4 framing is used.
- The transmission rate is the port's maximum rate.
- Minimum frame sizes are used.
- A data pattern of incrementing bytes (00 01 02...) is used.
- The source MAC address is set from the value associated with the indicated sending port.
- The destination MAC address is set via an ARP lookup on the destination IP address, which is set using UDF4.
- A single burst of packets is sent per stream, with the count equal to the count of addresses in the route range.
- UDF4 or IP address controls are used to iterate through the count of addresses in the route range; it should not be reprogrammed.

ripServer getFirstRouter

Access the first RIP router in the list*.* The results may be accessed using the *ripInterfaceRouter* command. Specific errors are:

- ripServer select has not been called.
- There are no routers in the list.

ripServer getNextRouter

Access the next RIP router in the list*.* The results may be accessed using the *ripInterfaceRouter* command. Specific errors are:

- ripServer select has not been called.
- ripServer getFirstRouter has not been called.
- There is no more routers in the list.

ripServer getRouter *routerId*

Access the RIP router with an identifier of *routerId.* The results may be accessed using the *ripInterfaceRouter* command. Specific errors are:

- ripServer select has not been called.
- There is no router with this *routerId* in the list.

ripServer select *chasID cardID portID*

Accesses the RIP component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The RIP protocol package has not been installed.
- Invalid port specified.

ripServer setRouter *routerLocalId*

Sets the values for the router's entry in the list with an identifier of *routerLocalId* based on changes made through the *ripInterfaceRouter* command. This command should be used to change a running configuration and must be followed by an *ripServer write* command in order to send these changes to the protocol server. Specific errors are:

- · A router with this *routerLocalId* does not exist in the list.

ripServer write

Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card and port. Before using this command, use the *ripServer select* command to select the port.

EXAMPLES

```
package req IxTclHal

set host localhost

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts “Could not connect to $host”

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {

ixPuts $::ixErrorInfo

return 1

}
```

```
# Get the chassis ID to use in port lists
set chas [ixGetChassisID $host]
# Port is chassis 1, card 1, port 1 with a MAC addr of
# 00 0a de 00 01 01
set card 1
set port 1
set pl [list [list 1 $card $port]]
set myMac [format "00 0a de 00 %02x %02x" $card $port]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}
# Take ownership of the ports we'll use
if [ixTakeOwnership $pl] {
ixPuts $::ixErrorInfo
return 1
}
# The router being simulated on the port
set router 198.18.1.2
set neighbor 198.18.1.1
# Basic parameters for the RIP router
set mask 255.255.255.0
set sendType ripBroadcastV2
set receiveType ripReceiveVersion2
set updateInterval 30
set updateIntervalOffset 5
#An array of multiple route ranges, organized as a table
# Num
# RangeIp Prefix Routes Metric Next Hop Tag
set routeRanges {{10.0.0.0 13 1000 4 198.18.1.2 9} \
{20.0.0.0 17 1000 4 198.18.1.2 10}}
# Initialize port
port setFactoryDefaults $chas $card $port
port setDefault
port set $chas $card $port
# Select the port and clear all defined routers
ripServer select $chas $card $port
ripServer clearAllRouters
set routerId 1
set rangeId 1
# For all the defined routers
while {[llength $routeRanges] 0} {
# Extract the information from the table and set the route
ranges
scan [lindex $routeRanges 0] “%s %s %s %s %s %s” \
myIp prefix numRoutes metric nextHop routeTag
ripRouteRange setDefault
ripRouteRange config -enableRouteRange true
ripRouteRange config -routeTag $routeTag
ripRouteRange config -networkIpAddress $myIp
ripRouteRange config -networkMaskWidth $prefix
ripRouteRange config -numberOfNetworks $numRoutes
ripRouteRange config -nextHop $nextHop
ripRouteRange config -metric $metric
# Create a name for each individual route range
ripInterfaceRouter addRouteRange \
[format “routeRange%02d” $rangeId]
incr rangeId
set routeRanges [lrange $routeRanges 1 end]
}
# Set up the interface table for an IPv4 and IPv6 interface
# on the port
interfaceTable select $ch $ca $po
interfaceTable clearAllInterfaces
interfaceIpV6 setDefault
interfaceIpV6 config -ipAddress
{0:0:0:0:0:0:0:1}
```

```
interfaceIpV6 config -maskWidth 64
interfaceEntry addItem addressTypeIpV6
interfaceIpV4 setDefault
interfaceIpV4 config -ipAddress $router
interfaceIpV4 config -gatewayIpAddress
$neighbor
interfaceIpV4 config -maskWidth 24
interfaceEntry addItem addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable
true
interfaceEntry config -description
{Port 01:01 Interface}
interfaceEntry config -macAddress
$myMac
interfaceTable write
# Now set the basic parameters for the router
ripInterfaceRouter setDefault
ripInterfaceRouter config -enableRouter true
ripInterfaceRouter config -protocolInterfaceDescription {Port
04:01 Interface}
ripInterfaceRouter config -sendType $sendType
ripInterfaceRouter config -receiveType $receiveType
ripInterfaceRouter config -updateInterval $updateInterval
ripInterfaceRouter config -updateIntervalOffset
$updateIntervalOffset
# And add the router to the ripServer with a unique ID
ripServer addRouter [format “router%02d” $routerId]
arpServer setDefault
arpServer config -mode arpGatewayOnly
arpServer set $chas $card $port
protocolServer get $chas $card $port
protocolServer config -enableArpResponse true
protocolServer config -enableRipService true
protocolServer config -enablePingResponse false
protocolServer set $chas $card $port
# Send to the hardware
ixWritePortsToHardware pl
ixCheckLinkState pl
# And start RIP on the port
ixStartRip pl
# Disable routeRange1 while rip server is runnung.
# This is the same as removing the route range from router
ripServer select $chas $card $port
if [ripServer getRouter router01] {
logMsg "Error getting router01"
}
if [ripInterfaceRouter getRouteRange routeRange01] {
logMsg "Error getting routeRange01"
}
# Disable the route range
ripRouteRange config -enableRouteRange false
if [ripInterfaceRouter setRouteRange routeRange01] {
logMsg "Error setting routeRange01"
}
if [ripServer write] {
logMsg "Error writing ripServer"
}
# If you wanted to add a route range while rip server is running,
# -Configure it disabled before starting rip server and then
# enable it
# Let go of the ports that we reserved
ixClearOwnership $pl
# Disconnect from the chassis we're using
ixDisconnectFromChassis $host
# If we're running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
```

```
    }
```

SEE ALSO

ripInterfaceRouter, *ripRouteRange*

NAME - rsvpCustomTlv

rsvpCustomTlv — sets up a custom RSVP TLV item.

SYNOPSIS

rsvpCustomTlv *subcommand options*

DESCRIPTION

The *rsvpCustomTlv* command holds a generalized type-length-value object used in many RSVP protocol messages. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on RSVP testing with Ixia equipment. Refer to *rsvpCustomTlv* for an overview of this command.

STANDARD OPTIONS

cType

The C-Type distinguisher of the TLV. *(default = 0)*

data

The data associated with the TLV. The length of the TLV will be calculated from the length of this data. *(default = {})*

tlvClass

The class distinguisher of the TLV. *(default = 0)*

COMMANDS

The **rsvpCustomTlv** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpCustomTlv cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpCustomTlv** command.

rsvpCustomTlv config *option value*

Modify the configuration options of the rsvpCustomTlv. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpCustomTlv.

rsvpCustomTlv setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpNeighborPair, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpDestinationRange

rsvpDestinationRange — configures an RSVP destination range.

NOTE: Destination Range as used in this document is synonymous with Tail Range of IxNetwork application.

SYNOPSIS

rsvpDestinationRange *subcommand options*

DESCRIPTION

The *rsvpDestinationRange* command represents a simulated destination range. In addition to some identifying options, it holds a number of lists for the destination range:

- Sender ranges — a set of MPLS routers which are the tunnel start-points, constructed in the *rsvpSenderRange* command.
- ERO item — a set of addresses associated with the Explicit Route Option of RSVP messages, constructed in the *rsvpEroItem* command. This option indicates the path through a set of MPLS routers that the tunnel is to take. This is used when the destination range is associated with an Ingress router.
- RRO item — a set of addresses associated with the Returned Route Option, constructed in the *rsvpRroItem* command. This option indicates the set of MPLS routers that were used in a tunnel's creation. This is used when the destination range is associated with an Egress router.
- RESV TLV — a set of custom TLVs to be included in RESV messages. These may only be used for egress routers.
- RESV TEAR TLV — a set of custom TLVs to be included in RESV TEAR messages. These may only be used for egress routers.
- RESV ERR TLV — a set of custom TLVs to be included in RESV ERR messages. These may only be used for ingress routers.
- PATH TLV — a set of custom TLVs to be included in PATH messages. These may only be used for egress routers.

Destination ranges defined in this command are added to an *rsvpNeighborPair* using the *rsvpNeighborPair addDestinationRange* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on RSVP testing with Ixia equipment. Refer to rsvpDestinationRange for an overview of this command.

STANDARD OPTIONS

bandwidth

The requested bandwidth for the tunnel, expressed in kbits per second. *(default = 0)*

behavior

Indicates whether the destination range corresponds to an Ingress or Egress router. One of:

Option	Value	Usage
rsvpIngress	0	*(default)* Ingress Mode
rsvpEgress	1	Egress Mode

egressBehavior

Dictates the RSVP reservation style when the value of *behavior* is *rsvpEgress.* One of:

Option	Value	Usage
rsvpEgressAlwaysUseConfiguredStyle	0	*(default)* Always use the shared-explicit mode.
rsvpEgressUseSEIfInAttribute	1	Use SE reservation style when "SE Style Desired" flag is on in the Session Attribute of the PATH message.

emulationType

Selects the type of emulation for the RSVP Destination Ranges. One of:

Option	Value	Usage
rsvpTrafficEndpoint	0	
rsvpTunnelEndpoint	1	default
rsvpP2MPTunnelEndPoint	2	

enableDestination Range true / false

Enables the use of this destination range in the simulation. *(default = false)*

enableEro true / false

Enables use of the ERO option in Ingress mode. *(default = false)*

enableFixedLabelFor Resv true / false

Enables the use of a fixed label in RESV messages while in Egress mode. *(default = false)*

enableReflectRro true / false

Enables the reflection of a received RRO object for Egress mode destination ranges. When selected, any RRO items added with *addRroItem* are ignored. *(default = true)*

enableResvConf true / false

Enables the generation of RESV Confirmation messages for received RESV messages which contain a RESV Confirmation Class object. *(default = false)*

enableSendRro true / false

When the destination range is used in Ingress mode, this indicates that a SEND RRO option is to be included in RSVP messages sent downstream. *(default = false)*

eroMode

Indicates whether the DUT's address is to be prepended to the ERO list and whether it is a LOOSE or STRICT entry. One of:

Option	Value	Usage
rsvpNone	0	Do not prepend the DUT's address
rsvpPrependLoose	1	*(default)* Prepend the DUT's address as a LOOSE address.
rsvpPrependStrict	2	Prepend the DUT's address as a STRICT address.

fromIpAddress

The IP address of the first destination router. *(default = 0.0.0.0)*

isConnectedIp Appended true / false

If true, append the connected IP as a RRO/SRRO subobject at the end of the RRO / SRRO list in the packet. Note that all flags will be set to 0 if this automatic inclusion option is used.

isHeadIpPrepended true / false

If true, prepend the tunnel head IP as a RRO/SRRO subobject at the beginning of the RRO / SRRO list in the packet. Note that all flags will be set to 0 if this automatic inclusion option is used.

isLeafIpPrepended true / false

If true, prepend the tunnel leaf IP as a RRO/SRRO subobject at the beginning of the RRO / SRRO list in the packet. Note that no label will be automatically inserted and all flags will be set to 0 if this automatic inclusion option is used.

isSendingAsRro true / false

If true, send this as a RRO.

isSendingAsSrro true / false

If true, send this as a SRRO. Note that both Send as RRO and Send as SRRO can be selected at the same time if so required by the user.

labelValue

If *enableFixedLabelForResv* is *true,* then this is the fixed label to use. One of:

Option	Value	Usage
rsvpLabelValueExplicitNull	0	*(default)*
rsvpLabelValueRouterAlert	1	
rsvpLabelValueIPv6ExplicitNull	2	
rsvpLabelValueImplicitNull	3	

p2mpId

The P2MP identifier represented in IP address format.

prefixLength

If the DUT's address is to be prepended to the ERO list, this indicates what prefix length is to be used for the entry. *(default = 32)*

rangeCount

The number of destination routers. Each router's address is one greater than the previous one's. *(default = 1)*

refreshInterval

When the destination range is *rsvpEgress*, this indicates the time, in seconds, between the simulated router's message to the DUT. *(default = 30,000)*

reservationStyle

The reservation style desired. One of:

Option	Value	Usage
rsvpFF	1	Fixed filtered mode
rsvpSE	2	*(default)* Shared explicit mode

timeoutMultiplier

The number of Hellos before a router is declared dead*. (default = 3)*

tunnelIdEnd

The end of the range of tunnel IDs. *(default = 1)*

tunnelIdStart

Sets the start of the range of Tunnel IDs to be used in simulations. *(default = 1)*

COMMANDS

The **rsvpDestinationRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

GENERAL COMMANDS

rsvpDestinationRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpDestinationRange** command.

rsvpDestinationRange config *option value*

Modify the configuration options of the rsvpDestinationRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpDestinationRange.

rsvpDestinationRange setDefault

Sets default values for all configuration options.

ERO LIST COMMANDS

rsvpDestinationRange addEroItem

Adds the ERO item described in the *rsvpEroItem* command to the list of ERO items associated with the destination range. Specific errors are:

- The parameters in *rsvpEroItem* are invalid.

rsvpDestinationRange clearAllEro

Deletes all of the ERO items.

rsvpDestinationRange getFirstEroItem

Access the first ERO item in the list. The results may be accessed using the *rsvpEroItem* command. Specific errors are:

- rsvpServer select has not been called.
- There are no ERO items in the list.

rsvpDestinationRange getNextEroItem

Access the next ERO item in the list*.* The results may be accessed using the *rsvpEroItem* command. Specific errors are:

- rsvpDestinationRange getFirstEroItem has not been called.
- There is no more ERO items in the list.

PATH TLV LIST COMMANDS

rsvpDestinationRange addPathTlv

Adds the PATH TLV described in the *rsvpCustomTlv* command to the list of TLVs associated with the destination range.

rsvpDestinationRange clearPathTlvList

Deletes all of the PATH TLVs.

rsvpDestinationRange delPathTlv

Deletes the current PATH TLV accessed through the use of *getFirstPathTlv/getNextPathTlv*. Specific errors are:

- No items in the list.
- *getFirstPathTlv* has not been called.

rsvpDestinationRange getFirstPathTlv

Access the first PATH TLV in the list. The results may be accessed using the *rsvpCustomTlv* command.Specific errors are:

- There are no items in the list.

rsvpDestinationRange getNextPathTlv

Access the next PATH TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- *getFirstPathTlv* has not been called.
- There is no more items in the list.

RESV TLV LIST COMMANDS

rsvpDestinationRange addResvTlv

Adds the RESV TLV described in the *rsvpCustomTlv* command to the list of TLVs associated with the destination range.

rsvpDestinationRange clearResvTlvList

Deletes all of the RESV TLVs.

rsvpDestinationRange delResvTlv

Deletes the current RESV TLV accessed through the use of *getFirstResvTlv/getNextResvTlv*. Specific errors are:

- No items in the list.
- *getFirstResvTlv* has not been called.

rsvpDestinationRange getFirstResvTlv

Access the first RESV TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- There are no items in the list.

rsvpDestinationRange getNextResvTlv

Access the next RESV TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- *getFirstResvTlv* has not been called.
- There is no more items in the list.

RESV ERR TLV LIST COMMANDS

rsvpDestinationRange addResvErrTlv

Adds the RESV ERR TLV described in the *rsvpCustomTlv* command to the list of TLVs associated with the destination range.

rsvpDestinationRange clearResvErrTlvList

Deletes all of the RESV ERR TLVs.

rsvpDestinationRange delResvErrTlv

Deletes the current RESV ERR TLV accessed through the use of *getFirstesvErrTlv/getNextResvErrTlv*. Specific errors are:

- No items in the list.
- *getFirstResvErrTlv* has not been called.

rsvpDestinationRange getFirstResvErrTlv

Access the first RESV ERR TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- There are no items in the list.

rsvpDestinationRange getNextResvErrTlv

Access the next RESV ERR TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- *getFirstResvErrTlv* has not been called
- There is no more items in the list.

RESV TEAR TLV LIST COMMANDS

rsvpDestinationRange addResvTearTlv

Adds the RESV TEAR TLV described in the *rsvpCustomTlv* command to the list of TLVs associated with the destination range.

rsvpDestinationRange clearResvTearTlvList

Deletes all of the RESV TEAR TLVs.

rsvpDestinationRange delResvTearTlv

Deletes the current RESV TEAR TLV accessed through the use of *getFirstResvTearTlv/getNextResvTearTlv*. Specific errors are:

- No items in the list.
- *getFirstResvTearTlv* has not been called.

rsvpDestinationRange getFirstResvTearTlv

Access the first RESV TEAR TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- There are no items in the list.

rsvpDestinationRange getNextResvTearTlv

Access the next RESV TEAR TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- *getFirstResvTearTlv* has not been called.
- There is no more items in the list.

RRO LIST COMMANDS

rsvpDestinationRange addRroItem

Adds the RRO item described in the *rsvpRroItem* command to the list of RRO items associated with the destination range. Specific errors are:

- The parameters in *rsvpRroItem* are invalid.

rsvpDestinationRange clearAllRro

Deletes all of the user LSA groups.

rsvpDestinationRange getFirstRroItem

Access the first RRO item in the list. The results may be accessed using the *rsvpRroItem* command. Specific errors are:

- There are no RRO items in the list.

rsvpDestinationRange getNextRroItem

Access the next RRO item in the list. The results may be accessed using the *rsvpRroItem* command. Specific errors are:

- rsvpDestinationRange getFirstRroItem has not been called.
- There is no more RRO items in the list.

SENDER RANGE LIST COMMANDS

rsvpDestinationRange addSenderRange*senderRangeId*

Adds the route range described in the *rsvpSenderRange* command to the list of sender ranges associated with the destination range. The sender range's entry in the list is given an identifier of *senderRangeId*. Specific errors are:

- The parameters in *rsvpSenderRange* are invalid.
- A destination range with this *senderRangeId* exists already in the list.

rsvpDestinationRange clearAllSender

Deletes all of the sender ranges.

rsvpDestinationRange delSenderRange *senderRangeId*

Deletes the sender range with an identifier of *senderRangeId*. Specific errors are:

- No destination range with this *senderRangeId* exists in the list.

rsvpDestinationRange getFirstSenderRange

Access the first sender range in the list. The results may be accessed using the *rsvpSenderRange* command. Specific errors are:

- There are no sender ranges in the list.

rsvpDestinationRange getNextSenderRange

Access the next sender range in the list. The results may be accessed using the *rsvpSenderRange* command. Specific errors are:

- rsvpDestinationRange getFirstSenderRange has not been called.
- There is no more sender ranges in the list.

rsvpDestinationRange getSenderRange *senderRangeId*

Accesses the sender range's entry in the list with an identifier of *senderRangeId*. The sender range is accessed in the *rsvpSenderRange* command. Specific errors are:

- A destination range with this *senderRangeId* does not exist in the list.

rsvpDestinationRange setSenderRange *senderRangeId*

Sets the values for the sender range's entry in the list with an identifier of *senderRangeId* based on changes made through the *rsvpSenderRange* command. This command can be used to change a running configuration and must be followed by an *rsvpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *senderRangeId* does not exist in the list.

P2MP COMMANDS

rsvpDestinationRange addTunnelLeafRange *tunnelLeafRangeId*

Adds the tunnel leaf range described in the *rsvpTunnelLeafRange* command to the list of sender ranges associated with the destination range. The sender range's entry in the list is given an identifier of *tunnelLeafRangeId*. Specific errors are:

- The parameters in *rsvpTunnelLeafRange* are invalid.
- A destination range with this *tunnelLeafRangeId* exists already in the list.

rsvpDestinationRange addTailTrafficEndPoint *tailTrafficEndpointId*

Adds the tail traffic endpoint described in the *rsvpTunnelHeadTrafficEndpoint* command to the list of sender ranges associated with the destination range. The sender range's entry in the list is given an identifier of *senderRangeId*. Specific errors are:

- The parameters in *rsvpTunnelHeadTrafficEndpoint* are invalid.
- A destination range with this *tailTrafficEndpointId* exists already in the list.

rsvpDestinationRange clearAllTunnelLeafRange

Deletes all of the tunnel leaf ranges.

rsvpDestinationRange clearAllTailTrafficEndPoint

Deletes all of the tail traffic endpoints.

rsvpDestinationRange delTunnelLeafRange *tunnelLeafRangeId*

Deletes the tunnel leaf range with an identifier of *tunnelLeafRangeId*. Specific errors are:

- No destination range with this *tunnelLeafRangeId* exists in the list.

rsvpDestinationRange delTailTrafficEndPoint *tailTrafficEndpointId*

Deletes the sender range with an identifier of *tailTrafficEndpointId*. Specific errors are:

- No destination range with this *tailTrafficEndpointId* exists in the list.

rsvpDestinationRange getFirstTunnelLeafRange

Access the first tunnel leaf range in the list. The results may be accessed using the *rsvpTunnelLeafRange* command. Specific errors are:

- There are no tunnel leaf ranges in the list.

rsvpDestinationRange getFirstTailTrafficEndPoint

Access the first tail traffic endpoint in the list. The results may be accessed using the *rsvpTunnelHeadTrafficEndpoint* command.

rsvpDestinationRange getNextTunnelLeafRange

Access the next tunnel leaf range in the list. The results may be accessed using the *rsvpTunnelLeafRange* command.

rsvpDestinationRange getNextTailTrafficEndPoint

Access the next tail traffic endpoint in the list. The results may be accessed using the *rsvpTunnelHeadTrafficEndpoint* command.

rsvpDestinationRange getTunnelLeafRange

Access the first sender range in the list. The results may be accessed using the *rsvpTunnelLeafRange* command. Specific errors are:

- There are no tunnel leaf ranges in the list.

rsvpDestinationRange getTailTrafficEndPoint

Access the next sender range in the list. The results may be accessed using the *rsvpTunnelHeadTrafficEndpoint* command. Specific errors are:

- rsvpDestinationRange getFirstTailTrafficEnpointRange has not been called.
- There is no more sender ranges in the list.

rsvpDestinationRange setTunnelLeafRange tunnelLeafRangeId

Sets the values for the tunnel leaf range's entry in the list with an identifier of *tunnelLeafRangeId* based on changes made through the *rsvpTunnelLeafRange* command. This command can be used to change a running configuration and must be followed by an *rsvpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *tunnelLeafRangeId* does not exist in the list.

rsvpDestinationRange setTailTrafficEndPoint tailTrafficEndpointId

Sets the values for the tail traffic endpoint entry in the list with an identifier of *tailTrafficEndpointId* based on changes made through the *rsvpTunnelTailTrafficEndpoint* command. This command can be used to change a running configuration and must be followed by an *rsvpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *tailTrafficEndpointId* does not exist in the list.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpServer, *rsvpNeighborPair*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpEroItem

rsvpEroItem — sets up the parameters associated with an RSVP ERO item.

SYNOPSIS

rsvpEroItem *subcommand options*

DESCRIPTION

The *rsvpEroItem* holds the information related to an ERO item used for in Ingress mode. ERO items are added into the *rsvpDestinationRange* list using the *rsvpDestinationRange addEroItem* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on RSVP testing with Ixia equipment. Refer to rsvpEroItem for an overview of this command.

STANDARD OPTIONS

asNumber

If the *type* field is *rsvpAs*, then this is the ERO value as an Autonomous System Number. *(default = 0)*

enableLooseFlag

Indicates whether the ERO item is to be considered a LOOSE item or a STRICT item. One of:

Option	Value	Usage
rsvpPrependLoose	1	*(default)* prepend the DUT's address as a LOOSE address.
rsvpPrependStrict	2	prepend the DUT's address as a STRICT address.

ipAddress

If the *type* field is *rsvpEroIpv4*, then this is the ERO value as an IP address prefix. *(default = 0.0.0.0)*

prefixLength

If the *type* field is *rsvpEroIpv4*, then this defines the prefix length of the DUT IP address. *(default = 0)*

type

The type of contents in the ERO entry. One of:

Option	Value	Usage
rsvpEroIpv4	0	*(default)* an IPv4 address
rsvpAs	1	an Autonomous System

COMMANDS

The **rsvpEroItem** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpEroItem cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpEroItem** command.

rsvpEroItem config *option value*

Modify the configuration options of the rsvpEroItem. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpEroItem.

rsvpEroItem setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *rsvpServer*

SEE ALSO

rsvpServer, *rsvpNeighborPair*, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpRroItem*

NAME - rsvpNeighborPair

rsvpNeighborPair — configures an RSVP neighbor pair.

SYNOPSIS

rsvpNeighborPair *subcommand options*

DESCRIPTION

The *rsvpNeighborPair* command represents a pair of routers the DUT and a directly connected simulated router. In addition to some identifying options, it holds three lists for the neighbor pair:

- Destination ranges — routers which are the destination of constructed MPLS tunnels, constructed in the *rsvpDestinationRange* command.
- Hello TLVs — generalized TLV messages that are included with all HELLO messages and built with the *rsvpCustomTlv* command.

Neighbor pairs defined in this command are added to an *rsvpServer* using the *rsvpServer addNeighborPair* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on RSVP testing with Ixia equipment. Refer to *rsvpNeighborPair* for an overview of this command.

STANDARD OPTIONS

actualRestartTime

The actual restart time is the interval after which a hello packet is sent with a new Src Instance Id. The default value is 15000 ms.

assignedLabel

Label value assigned to the LSP/Tunnel (by the Ixia-emulated router) in response to a Label Request from the DUT.

dutAddresss

The IP address of the Device Under Test. This is the RSVP router that the simulated router is directly connected to. *(default = 0.0.0.0)*

enableBFDRegistration

If true, enables BFD registration with RSVP-TE.

enableBundleMessageSending

If true, enables the sending of RSVP Bundle Message.

enableGraceful RestartHelperMode true | false

If true, enables the graceful restart helper mode.

enableGraceful RestartingMode true | false

If true, enables the graceful restart - restarting mode.

enableHello true | false

Enables the transmission of HELLO messages between the simulated router and the DUT. *(default = 0)*

enableNeighborPair true | false

Enables the use of this neighbor in the simulation. *(default = 0)*

enableRefresh Reduction true | false

Enables the use of RSVP refresh reduction, indicating that summary refresh messages can be sent to compatible nodes to maintain state. Messages are sent with the interval expressed in *summaryRefreshInterval*. *(default = false)*

gracefulRestart StartTime

The time interval after this restart timer is fired, and the neighboring nodes are restarted. During this interval the hello message are not being sent.

The default value is 30000 ms.

gracefulRestartUpTime

After the Restarting time is over, Ixia waits for this configured interval before trying to repeat the Restart cycle. This is effective only when the number of restarts is not equal to the user-configured number of Graceful Restart cycles . After that Ixia will not take any action to being down a Neighborship on its own.

The default value is 30000 ms.

helloInterval

The interval, in seconds, between HELLO messages. *(default = 5)*

helloTimeoutMultiplier

The number of Hellos sent without confirmation before the DUT is considered dead. *(default = 3)*

ipAddress

The IP address of the simulated router. *(default = 0.0.0.0)*

labelSpaceEnd

The last label to be used for RSVP tunnels. *(default =1000)*

labelSpaceStart

The first label to be used for RSVP tunnels. *(default = 16)*

leafIp

It contains the value of the leafIp which identifies one particular P2MP RSVP-TE sub-lsp for which the label was returned. This does not have any significance for P2P lsps.

lsp_tunnel

(Read-only.) This is a string identifier that contains the information to map the returned label to a particular P2P lsp or P2MP lsp.

numAssignedLabels

The total number of assigned labels.

numberOfGraceful Restarts

The number of times the Ixia emulated RSVP neighbor will move to Restarting / Recovering and Up states before stopping the cycle.

The default value is 0.

numRxLabels

(Read-only.) The number of RSVP labels received from a successful *rsvpNeighborPair getLabels* command.

recoveryTimeInterval

Ixia waits for a configured interval for the DUT to help it recover the egress LSPs. If no recovery label is received from the DUT within this time, those Egress LSPs are treated as having time-outed and the labels are removed.

The default value is 30000 ms.

reservationState

The reservation state, once there is a graceful restart. The values are:

- None=0 (Default)
- Stale=1 (Recovery State but Recovery Label not yet received)
- Recovered=2 (Recovery Label received)
- Restarting=3 (RSVP emulated Router is restarting)

restartTimeInterval

This value along with the Recovery Time is advertised in the Hello-packets as part of a Restart-capability object.

The default value is 30000 ms.

rxLabel

(Read-only.) This is the MPLS label associated with the tunnel ID in *lsp_tunnel.*

summaryRefresh Interval

If *enableRefreshReduction* is *true*, the time interval, expressed in milliseconds, between summary refreshes. *(default = 15,000)*

type

This signifies the type of the lsp for which the current label was returned. The values are:

- RSVP-TE
- RSVP-TE P2MP

COMMANDS

The **rsvpNeighborPair** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

GENERAL COMMANDS

rsvpNeighborPair cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpNeighborPair** command.

rsvpNeighborPair config *option value*

Modify the configuration options of the rsvpNeighborPair. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpNeighborPair.

rsvpNeighborPair setDefault

Sets default values for all configuration options.

DESTINATION RANGE COMMANDS

rsvpNeighborPair addDestinationRange *destId*

Adds the Destination Range described in the *rsvpDestinationRange* command to the list of Destination Ranges associated with the router. The Destination Range's entry in the list is given an identifier of *destId*. Specific errors are:

- The parameters in *rsvpDestinationRange* are invalid.
- A router with this *destId* exists already in the list.

rsvpNeighborPair clearAllDestinationRange

Deletes all of the router Destination Ranges.

rsvpNeighborPair delDestinationRange *destId*

Deletes the router Destination Range with an identifier of *destId*. Specific errors are:

- No router with this *destId* exists in the list.

rsvpNeighborPair getFirstDestinationRange

Access the first Destination Range in the list. The results may be accessed using the *rsvpDestinationRange* command. Specific errors are:

- rsvpServer select has not been called.
- There are no Destination Ranges in the list.

rsvpNeighborPair getDestinationRange *destId*

Accesses the Destination Range's entry in the list with an identifier of *destId*. The Destination Range is accessed in the *rsvpDestinationRange* command. Specific errors are:

- A router with this *destId* does not exist in the list.

rsvpNeighborPair getNextDestinationRange

Access the next Destination Range in the list. The results may be accessed using the *rsvpDestinationRange* command. Specific errors are:

- rsvpNeighborPair getFirstDestinationRange has not been called.
- There is no more Destination Ranges in the list.

rsvpNeighborPair setDestinationRange *destId*

Sets the values for the destination range's entry in the list with an identifier of *destId* based on changes made through the *rsvpDestinationRange* command. This command can be used to change a running configuration and must be followed by an *rsvpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *interfaceLocalId* does not exist in the list.

HELLO TLV LIST COMMANDS

rsvpNeighborPair addHelloTlv

Adds the HELLO TLV described in the *rsvpCustomTlv* command to the list of TLVs associated with the neighbor pair.

rsvpNeighborPair clearHelloTlvList

Deletes all of the HELLO TLVs.

rsvpNeighborPair delHelloTlv

Deletes the current HELLO TLV accessed through the use of *getFirstHelloTlv/getNextHelloTlv*. Specific errors are:

- No items in the list.
- *getFirstHelloTlv* has not been called.

rsvpNeighborPair getFirstHelloTlv

Access the first HELLO TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- There are no items in the list.

rsvpNeighborPair getNextHelloTlv

Access the next HELLO TLV in the list*.* The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- *getFirstHelloTlv* has not been called.
- There is no more items in the list.

Received LABEL RETRIEVAL COMMANDS

rsvpNeighborPair getFirstLabel

This command must be preceded by use of the *rsvpNeighborPair getLabels* command and followed by multiple uses of *rsvpNeighborPair getNextLabel.* This command fetches the first of the labels in the list into memory. The data associated with the individual label may be read through the use of the *lsp_tunnel* and *rxLabel* options.

rsvpNeighborPair getLabels

This command must be preceded by use of the *rsvpNeighborPair requestRxLabels* and followed by a call to *rsvpNeighborPair getFirstLabel.* This command determines whether the reading of labels from the protocol server has completed. This command should be called until it returns a `0', or until some suitable period of time has elapsed. The number of labels is available in the *numRxLabels* option.

rsvpNeighborPair getNextLabel

This command must be preceded by use of the *rsvpNeighborPair getFirstLabel* command and repeated multiple times to obtain all of the learned LSAs*.* This command fetches the next of the labels in the list into memory. The data associated with the individual label may be read through the use of the *lsp_tunnel* and *rxLabel* options.

Assigned LABEL RETRIEVAL COMMANDS

rsvpNeighborPair requestAssignedLabels

This command requests that the assigned RSVP labels associated with this neighbor pair be retrieved from the protocol server. This command must be followed by call to *rsvpNeighborPair getLabels.*

rsvpNeighborPair getLabels

This is the second step in retrieving assigned labels from the DUT. This subcommand allows the Tcl program to wait until the labels have been retrieved.

rsvpNeighborPair getFirstLabel

This command must be preceded by the use of the *rsvpNeighborPair getLabels* command and followed by multiple uses of *rsvpNeighborPair getNextLabel.* This command fetches the first of the labels in the list into memory. The data associated with the individual label may be read through the use of the *lsp_tunnel* and *assignLabel* options.

rsvpNeighborPair getNextLabel

This command must be preceded by use of the *rsvpNeighborPair getFirstLabel* command and repeated multiple times to obtain all of the learned LSAs*.* This command fetches the next of the labels in the list into memory. The data associated with the individual label may be read through the use of the *lsp_tunnel* and *assignLabel* options.

rsvpNeighborPair requestRxLabels

Requests that the received RSVP labels associated with this neighbor pair be retrieved from the protocol server. This command must be followed by call to rsvpNeighborPair getLabels.

EXAMPLES

See examples under *rsvpServer*

SEE ALSO

rsvpServer, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpPlrNodeIdPair

rsvpPlrNodeIdPair — sets up a Point of Local Repair node for fast reroute.

SYNOPSIS

rsvpPlrNodeIdPair *subcommand options*

DESCRIPTION

The *rsvpPlrNodeIdPair* command holds a single pair of items related to the DETOUR object used for RSVP fast reroute.

STANDARD OPTIONS

avoidNodeId

The IPv4 address identifying the immediate downstream node that the PLR is trying to avoid. The Router ID of the downstream node is the preferred value. *(default = 0.0.0.0)*

plrId

The IPv4 address identifying the beginning point of detour which is a PLR. Any local address on the PLR can be used. *(default = 0.0.0.0)*

COMMANDS

The **rsvpPlrNodeIdPair** command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

rsvpPlrNodeIdPair cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpPlrNodeIdPair** command.

rsvpPlrNodeIdPair config *option value*

Modify the configuration options of the rsvpPlrNodeIdPair. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpPlrNodeIdPair.

rsvpPlrNodeIdPair setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpServer, *rsvpNeighborPair*, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpRroItem*

NAME - rsvpRroItem

rsvpRroItem — sets up the parameters associated with an RSVP RRO item.

SYNOPSIS

rsvpRroItem *subcommand options*

DESCRIPTION

The *rsvpRroItem* holds the information related to an RRO item used for in Ingress mode. RRO items are added into the *rsvpDestinationRange* list using the *rsvpDestinationRange addRroItem* command. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on RSVP testing with Ixia equipment. Refer to rsvpRroItem for an overview of this command.

STANDARD OPTIONS

cType

If the *type* field is *rsvpRroIpv4*, then this is the C_Type of the included Label Object. *(default = 1)*

enableBandwidth Protection true | false

For use with egress destination ranges only, this indicates that bandwidth protection is available on the backup path. *(default = false)*

enableGlobalLabel true / false

If the *type* field is *rsvpRroIpv4*, then this indicates that the label will be understood if received on any interface. *(default = false)*

enableNode Protection true | false

For use with egress destination ranges only, this indicates that node protection is available on the backup path. *(default = false)*

enableProtection Available true / false

If the *type* field is *rsvpRroIpv4*, then this indicates that local protection is made available for the downstream link. *(default = false)*

enableProtectionInUse true / false

If the *type* field is *rsvpRroIpv4*, then this indicates that the local protection is being used currently to maintain this tunnel. *(default = false)*

ipAddress

If the *type* field is *rsvpRroIpv4*, then this is the RRO value as an IPv4 address. *(default = 0.0.0.0)*

label

If the *type* field is *rsvpLabel*, then this is the RRO value as an assigned label. *(default = 0)*

type

The type of contents in the ERO entry. One of:

Option	Value	Usage
rsvpRroIpv4	0	*(default)* an IPv4 address
rsvpLabel	1	an MPLS label

COMMANDS

The **rsvpRroItem** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpRroItem cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpRroItem** command.

rsvpRroItem config *option value*

Modify the configuration options of the rsvpRroItem. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpRroItem.

rsvpRroItem setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpServer, *rsvpNeighborPair*, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*

NAME - rsvpSenderRange

rsvpSenderRange — sets up the parameters associated with an RSVP sender range.

SYNOPSIS

rsvpSenderRange *subcommand options*

DESCRIPTION

The *rsvpSenderRange* command holds the information related to the originating routers for the MPLS tunnels being simulated in Ingress cases. Sender ranges are added into the *rsvpDestinationRange* list using the *rsvpDestinationRange addSenderRange* command. Three lists are maintained in this command:

- PLR — the fast reroute Point of Local Repair, constructed with the *rsvpPlrNodeIdPair* command.
- PATH TLV — a set of custom TLVs to be included in PATH messages.
- TEAR TLV — a set of custom TLVs to be included in TEAR messages.

Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on RSVP testing with Ixia equipment. Refer to *rsvpSenderRange* for an overview of this command.

STANDARD OPTIONS

bandwidth

Double value. The bandwidth requested for the connection, expressed in kbits/sec. *(default = 0.0)*

enableAutoSession Name true / false

Enables the Session Name to be generated automatically. *(default = true)*

enableBandwidth ProtectionDesired true | false

This indicates that bandwidth protection is desired from the PLRs along the backup protected LSP path. *(default = false)*

enableFacilityBackup Desired true | false

For fast reroute, indicates that the use of the Facility Backup feature is enabled. This may be combined with *enableOneToOneBackupDesired*. *enableFastReroute* must be *true* for this option to have any effect. *(default = false)*

enableFastReroute true | false

Enables the use of the fast reroute feature. *(default = false)*

enablePath Reoptimization

Enables the use of the Path Re-optimization option.

backupLspIdPoolStart

Enables to set the LSP Id for the re-optimized LSP.

enableLabelRecording Desired true / false

This indicates that label information is to be included when doing a record route operation. *(default = false)*

enableLocalProtection Desired true / false

This permits transit routers to use a local traffic rerouting repair mechanism, in the event of a fault on an adjacent downstream link or node. This may result in a violation of the explicit route object. *(default = true)*

enableNode ProtectionDesired true | false

This indicates that node protection is desired from the PLRs along the backup protected LSP path. *(default = false)*

enableOneToOne BackupDesired true | false

For fast reroute, indicates that the use of the Facility Backup feature is enabled. This may be combined with *enableFacilityBackupDesired*. *enableFastReroute* must be *true* for this option to have any effect. *(default = false)*

enableRaSession Attribute true / false

Enables the use of resource affinities as set by *excludeAny*, *includeAny*, and *includeAll*. *(default = false)*

enableSendDetour true | false

Enables the generation of the fast reroute DETOUR object, using the values in the PLR list. *enableFastReroute* must be *true* for this option to have any effect. *(default = false)*

enableSenderRange Desired true / false

Enables the sender range entry. *(default = false)*

enableSeStyleDesired true / false

This indicates that the tunnel ingress node may reroute this tunnel without tearing it down. A tunnel egress node should use the SE Style when responding with an RESV message. *(default = true)*

excludeAny

Represents a set of attribute filters associated with a tunnel, any of which renders a link unacceptable. *(default = 00 00 00 00)*

fastRerouteBandwidth

Double value. An element of the FAST_REROUTE object that indicates the bandwidth estimate for the protection path in bytes per second, expressed in 32-bit IEEE floating point format. *(default = 0.0)*

fastRerouteExcludeAny

An element of the FAST_REROUTE object that represents a set of attribute filters associated with a backup path, any of which render a link unacceptable. *(default = {00 00 00 00})*

fastRerouteHolding Priority

An element of the FAST_REROUTE object that indicates the priority of the backup path with respect to holding resources, in the range of 0 (highest) to 7 (lowest). *(default = 7)*

fastRerouteHopLimit

An element of the FAST_REROUTE object that is the maximum number of extra hops the backup path is allowed to take from the current PLR node to a merge point. *(default = 3)*

fastRerouteIncludeAll

An element of the FAST_REROUTE object that represents a set of attribute filters associated with a backup path, all of which must be present to render a link acceptable. *(default = {00 00 00 00})*

fastRerouteIncludeAny

An element of the FAST_REROUTE object that represents a set of attribute filters associated with a backup path, any of which render a link acceptable. *(default = {00 00 00 00})*

fastRerouteSetup Priority

An element of the FAST_REROUTE object that indicates the priority of the backup path with respect to taking resources, in the range of 0 (highest) to 7 (lowest). *(default = 7)*

fromIpAddress

The IP address of the first sender router. *(default = 0.0.0.0)*

holdingPriority

This is the session priority with respect to *holding* resources, such as keeping a session during preemption. The valid range is from 0 to 7. The highest priority is indicated by 0. *(default = 7)*

includeAll

Represents a set of attribute filters associated with a tunnel, all of which must be present for a link to be acceptable (with respect to this test). When all bits are set to 0 (null set), it automatically passes. *(default = 00 00 00 00)*

includeAny

Represents a set of attribute filters associated with a tunnel, any of which makes a link acceptable (with respect to this test). When all bits are set to 0 (null set), it automatically passes. *(default = 00 00 00 00)*

lspIdEnd

The end of the range of LSP IDs. *(default = 0)*

lspIdStart

The start of the range of LSP IDs to be generated. *(default = 0)*

maxPacketSize

The maximum packet size associated with the RSVP Sender's Traffic Specification. Expressed in bytes.

minPolicedUnit

The minimum policed unit size associated with the RSVP Sender's Traffic Specification. Expressed in bytes.

peakDataRate

The peak data rate associated with the RSVP Sender's Traffic Specification. Expressed in bytes per second.

rangeCount

The number of routers in the sender range. Each sender router has an IP address one higher than its predecessor. *(default = 1)*

refreshInterval

The value of the refresh interval, in milliseconds. *(default = 30,000)*

sessionName

If *enableAutoSessionName* is not set, this is the name assigned to this Session. *(default = "")*

setupPriority

This is the session priority with respect to *taking* resources, such as preempting another session. The valid range is from 0 to 7. The highest priority is indicated by 0. *(default = 7)*

timeoutMultiplier

The number of Hellos before a neighbor is declared dead. *(default = 3)*

tokenBucketRate

The token bucket rate associated with the RSVP Sender's Traffic Specification. Expressed in bytes per second.

tokenBucketSize

The token bucket size associated with the RSVP Sender's Traffic Specification. Expressed in bytes.

LOCAL EXECS

doMakeBeforeBreak

Triggers a block event per head range.

- selfId = A list of objects on which this exec can be used. This exec requires an object reference as an argument.
    - /vport/protocols/rsvp/neighbor Pair/destinationRange/Ingress.

sendReEvaluationRequest

Stops STP on a port or group of ports.

- selfId = A list of objects on which this exec can be used. This exec requires an object reference as an argument.
    - /vport/protocols/rsvp/neighbor Pair/destinationRange/Ingress.

COMMANDS

The **rsvpSenderRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

GENERAL COMMANDS

rsvpSenderRange cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpSenderRange** command.

rsvpSenderRange config *option value*

Modify the configuration options of the rsvpSenderRange. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpSenderRange.

rsvpSenderRange setDefault

Sets default values for all configuration options.

PATH TLV LIST COMMANDS

rsvpSenderRange addPathTlv

Adds the PATH TLV described in the *rsvpCustomTlv* command to the list of TLVs associated with the sender range.

rsvpSenderRange clearPathTlvList

Deletes all of the PATH TLVs.

rsvpSenderRange delPathTlv

Deletes the current PATH TLV accessed through the use of *getFirstPathTlv/getNextPathTlv*. Specific errors are:

- No items in the list.
- *getFirstPathTlv* has not been called.

rsvpSenderRange getFirstPathTlv

Access the first PATH TLVin the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- There are no items in the list.

rsvpSenderRange getNextPathTlv

Access the next PATH TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- *getFirstPathTlv* has not been called.
- There is no more items in the list.

PLR LIST COMMANDS

rsvpSenderRange addPlr

Adds the PLR described in the *rsvpPlrNodeIdPair* command to the list of TLVs associated with the sender range.

rsvpSenderRange clearPlrList

Deletes all of the PLRs.

rsvpSenderRange getFirstPlr

Access the first PLR in the list. The results may be accessed using the *rsvpPlrNodeIdPair* command. Specific errors are:

- There are no items in the list.

rsvpSenderRange getNextPlr

Access the next PLR in the list. The results may be accessed using the *rsvpPlrNodeIdPair* command. Specific errors are:

- *getFirstPlr* has not been called.
- There is no more items in the list.

TEAR TLV LIST COMMANDS

rsvpSenderRange addTearTlv

Adds the TEAR TLV described in the *rsvpCustomTlv* command to the list of TLVs associated with the destination range.

rsvpSenderRange clearTearTlvList

Deletes all of the TEAR TLVs.

rsvpSenderRange delTearTlv

Deletes the current TEAR TLV accessed through the use of *getFirstTearTlv/getNextTearTlv*. Specific errors are:

- No items in the list.
- *getFirstTearTlv* has not been called.

rsvpSenderRange getFirstTearTlv

Access the first TEAR TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- There are no items in the list.

rsvpSenderRange getNextTearTlv

Access the next TEAR TLV in the list. The results may be accessed using the *rsvpCustomTlv* command. Specific errors are:

- *getFirstTearTlv* has not been called.
- There is no more items in the list.

GRACEFUL RESTART COMMANDS

rsvpSenderRange addTunnelHeadToLeaf *tunnelHeadToLeafId*

Adds the tunnel head to leaf as described in the *rsvpTunnelHeadToLeaf* command to the list of tunnels associated with the destination range.

rsvpSenderRange addHeadTrafficEndPoint *headTrafficEndPointId*

Adds the head traffic endpoint as described in the *rsvpTunnelHeadTrafficEndPoint* command to the list of endpoint associated with the destination range.

rsvpSenderRange clearAllTunnelHeadToLeaf

Deletes all of the tunnel head to leaf.

rsvpSenderRange clearAllHeadTrafficEndPoint

Deletes all of the head traffic endpoint.

rsvpSenderRange delTunnelHeadToLeaf *tunnelHeadToLeafId*

Deletes the current tunnel accessed through the use of *getFirstTunnelHeadToLeaf/getNextTunnelHeadToLeaf*. Specific errors are:

- No items in the list.
- *getFirstTunnelHeadToLeaf* has not been called.

rsvpSenderRange delHeadTrafficEndPoint *headTrafficEndPointId*

Deletes the current traffic endpoint accessed through the use of *getFirstHeadTrafficEndPoint/getNextHeadTrafficEndPoint*. Specific errors are:

- No items in the list.
- *getFirstHeadTrafficEndPoint* has not been called.

rsvpSenderRange getTunnelHeadToLeaf *tunnelHeadToLeafId*

Retrieves the tunnel specified by *tunnelHeadToLeafId*.

rsvpSenderRange getHeadTrafficEndPoint *headTrafficEndPointId*

Retrieves the endpoint specified by *headTrafficEndPointId*.

rsvpSenderRange getFirstTunnelHeadToLeaf *tunnelHeadToLeafId*

Access the first current tunnel in the list. The results may be accessed using the *rsvpTunnelHeadToLeaf* command. Specific errors are:

- There are no items in the list.

rsvpSenderRange getFirstHeadTrafficEndPoint *headTrafficEndPointId*

Access the first head traffic endpoint in the list*.* The results may be accessed using the *rsvpHeadTrafficEndPoint* command. Specific errors are:

- There are no items in the list.

rsvpSenderRange getNextTunnelHeadToLeaf

Access the next current tunnel in the list. The results may be accessed using the *rsvpTunnelHeadToLeaf* command. Specific errors are:

- *getFirstTunnelHeadToLeaf* has not been called.
- There is no more items in the list.

rsvpSenderRange getNextHeadTrafficEndPoint

Access the next head traffic endpoint in the list. The results may be accessed using the *rsvpHeadTrafficEndPoint* command. Specific errors are:

- *getFirstTHeadTrafficEndPoint* has not been called.
- There is no more items in the list.

rsvpSenderRange setTunnelHeadToLeaf *tunnelHeadToLeafId*

Sets the values for the tunnels entry in the list with an identifier of *tunnelHeadToLeafId*. This command should be used to change a running configuration and must be followed by an rsvpServer write command in order to send these changes to the protocol server. Specific errors are:

- There is no more items in the list.

rsvpSenderRange setHeadTrafficEndPoint *headTrafficEndPointId*

Sets the values for the endpoint entry in the list with an identifier of *headTrafficEndPointId*. This command should be used to change a running configuration and must be followed by an rsvpServer write command in order to send these changes to the protocol server. Specific errors are:

- There is no more items in the list.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpServer, *rsvpNeighborPair*, *rsvpDestinationRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpTunnelHeadToLeaf

rsvpTunnelHeadToLeaf — This command is used for the enhanced functionality of ERO and SERO configuration for the head range.

SYNOPSIS

rsvpTunnelHeadToLeaf *subcommand options*

DESCRIPTION

The *rsvpTunnelHeadToLeaf* command enhances the functionality of ERO and SERO configuration for the head range.

STANDARD OPTIONS

dutHopType

Based on the input, the corresponding L bit in the packet is set. [RFC 3209]

dutPrefixLength

Prefix length of DUT.

enabled true / false

If true, the tunnel is enabled.

isAppendTunnelLeaf

If enabled, this appends the tunnel leaf at the end of the ERO / SERO list in the packet.

isPrependDut true / false

Enables prepend DUT to the ERO / SERO list.

isSendingAsEro true / false

If enabled, the entire configuration would go as ERO.

isSendingAsSero true / false

If enabled, the entire configuration would go as SERO.

NOTE: If ERO and SERO are both enabled, then the configuration would go both as ERO and SERO for that <head, leaf tuple.

subObjectList

The sub-object list for this ERO/SERO can be configured by typing it as a string.

- Input String: = NULL| [<Subobject ;< Subobject list]
- Sub-object list: = NULL| [<Subobject ;< Subobject list]
- Subobject: = <AS :< 1-65535 :< S|L>| <IP :< IP Addr/<1-32 :< S|L>

- IP Addr: = <0-255.<0-255.<0-255.<0-255
- NULL: = **Example** IP:2.2.2.2/24:S;AS:100:L;IP:33.33.33.33/32:S "

tunnelLeafCount

The count of tunnel leaf.

tunnelLeafIpStart

This contains the start IP address of leaf for which the ERO / SERO will be configured.

tunnelLeafHopType

This is enabled if Append Leaf is enabled. Based on the input, corresponding L bit in the packet is set. [RFC 3209]

tunnelLeafPrefixLength

Prefix length of tunnel leaf.

COMMANDS

The **rsvpTunnelHeadToLeaf** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpTunnelHeadToLeaf setDefault

Sets default values for all options.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpNeighborPair, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpTunnelHeadTrafficEndPoint

rsvpTunnelHeadTrafficEndPoint — This command configures the IP addresses to be used in the Source IP field in traffic to be sent over the LSPs originating from this Head Range.

SYNOPSIS

rsvpTunnelHeadTrafficEndPoint *subcommand options*

DESCRIPTION

The *rsvpTunnelHeadTrafficEndPoint* command is introduced to have future extendibility of supporting multiple ranges per Head Range.

STANDARD OPTIONS

endPointType

Sets the endpoint type for this head traffic endpoint. One of:

Option	Value	Usage
EndPointTypeIPv4		
EndPointTypeIPv6		

insertExplicit TrafficItem

This inserts an IPv6 Explicit NULL as the innermost label in addition to learned label when trying to generate IPv6 traffic over the IPv4 lsp. The purpose is to route the traffic to the IPv6 Protocol Stack at the egress for routing towards the IPv6 destination.

ipCount

This is used to simulate traffic from multiple source endpoints to be sent over the LSPs originated from the Head Range.

NOTE: Allows value greater than or equal to Tunnel Head IP Count. Default is 1.

ipStart

The start source IP address, one of IPv4 or IPv6, to be used for traffic to be sent over LSPs from the Head End Point.

COMMANDS

The **rsvpTunnelHeadTrafficEndPoint** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpTunnelHeadTrafficEndPoint setDefault

Sets default values for all options.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpNeighborPair, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpTunnelLeafRange

rsvpTunnelLeafRange — It describes a range of tunnel leaf endpoints when the emulation type in Tunnel Tail Ranges is set to RSVP-TE P2MP.

SYNOPSIS

rsvpTunnelLeafRange *subcommand options*

DESCRIPTION

The *rsvpTunnelLeafRange* command is enabled only if the emulation type in Tunnel Tail Ranges is set to RSVP-TE P2MP. There is separate configuration for Ingress and Egress leaf nodes.

STANDARD OPTIONS

enabled true / false

If true the tunnel leaf range is enabled.

ipCount

The number of IPv4 addresses in the range of Tunnel Tail addresses.

ipStart

The first IPv4 address in the range of Tunnel Tail addresses to be associated with the parent Tail Range. The p2mp RSVP-TE LSPs will terminate the sub-lsps for each p2mp lsp in the Tail Range to the set of endpoints identified by the these IPv4 addresses.

subLspDown true / false

This is a run-time configuration option which has immediate effect on RSVP state machine, unlike most other configuration options in IxNetwork, which require config object disable/enable for change to take effect.

COMMANDS

The **rsvpTunnelLeafRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpTunnelLeafRange setDefault

Sets default values for all options.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpNeighborPair, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpTunnelTailTrafficEndPoint

rsvpTunnelTailTrafficEndPoint — This command configures the IP addresses to be used in the Destination IP field in traffic to be sent over the LSPs terminating on this Tail Range.

SYNOPSIS

rsvpTunnelTailTrafficEndPoint *subcommand options*

DESCRIPTION

The *rsvpTunnelTailTrafficEndPoint* command is introduced to have future extendibility of supporting multiple ranges per Tail Range.

STANDARD OPTIONS

endPointType

Sets the endpoint type. One of:

Option	Value	Usage
EndPointTypeIPv4		
EndPointTypeIPv6		

ipCount

This indicates that the number of destination IPs to which the traffic sent over the P2MP RSVP-TE tunnel is destined. The minimum and default value is 1.

ipStart

The Start Destination IP Address for traffic that will be sent over the P2MP RSVP-TE tunnel. Normally, this will be an IPv4 or IPv6 Multicast address.

COMMANDS

The **rsvpTunnelTailTrafficEndPoint** command is invoked with the following sub-commands. If no subcommand is specified, returns a list of all subcommands available.

rsvpTunnelTailTrafficEndPoint **setDefault**

Sets default values for all options.

EXAMPLES

See examples under *rsvpServer*.

SEE ALSO

rsvpNeighborPair, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - rsvpServer

rsvpServer — accesses the RSVP component of the protocol server for a particular port.

NOTE: Sender Range as used in this document is synonymous with Head Range of IxNetwork application.

SYNOPSIS

rsvpServer *subcommand options*

DESCRIPTION

The *rsvpServer* command is necessary in order to access the RSVP protocol server for a particular port. The *select* subcommand **must** be used before all other RSVP commands. Refer to the *Ixia Reference Manual, Theory of Operations: Protocols* chapter for a discussion on RSVP testing with Ixia equipment. Refer to *rsvpServer* for an overview of this command.

STANDARD OPTIONS

enableBgpOverLsp true / false

Setting this option to true allows non-RSVP control packets (such as BGP control packets destined to the far-end PE) to be encapsulated with the MPLS label learned by RSVP. If it is set to false, no control packets are encapsulated with the MPLS label.

The sequence of commands for this option to be set is:

rsvpServer configure enableBgpOverLsp <*true/false* rsvpServer set rsvpServer write

COMMANDS

The **rsvpServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

rsvpServer addNeighborPair *NeighborPairId*

Adds the RSVP neighbor pair described in the *rsvpNeighborPair* command to the list of neighbor pairs associated with the port. The neighbor pair's entry in the list is given an identifier of *NeighborPairId*. Specific errors are:

- rsvpServer select has not been called.
- The neighbor pair parameters in *rsvpNeighborPair* are invalid.
- A neighbor pair with this *NeighborPairId* exists already in the list.

rsvpServer cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **rsvpServer** command.

rsvpServer clearAllNeighborPair

Deletes all the RSVP neighbor pairs in the list. Specific errors are:

- rsvpServer select has not been called.

rsvpServer config *option value*

Modify the configuration options of the rsvpServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for rsvpServer.

rsvpServer delNeighborPair *NeighborPairId*

Deletes the RSVP neighbor pair described that has an identifier of *NeighborPairId*. Specific errors are:

- rsvpServer select has not been called.
- There is no neighbor pair with this *NeighborPairId* in the list.

rsvpServer getFirstNeighborPair

Access the first RSVP neighbor pair in the list. The results may be accessed using the *rsvpNeighborPair* command. Specific errors are:

- rsvpServer select has not been called.
- There are no neighbor pairs in the list.

rsvpServer generateStreams

rsvpServer getNextNeighborPair

Access the next RSVP neighbor pair in the list. The results may be accessed using the *rsvpNeighborPair* command. Specific errors are:

- rsvpServer select has not been called.
- rsvpServer getFirstNeighborPair has not been called.
- There are no more neighbor pairs in the list.

rsvpServer getNeighborPair *NeighborPairId*

Access the RSVP neighbor pair with an identifier of *NeighborPairId.* The results may be accessed using the *rsvpNeighborPair* command. Specific errors are:

- rsvpServer select has not been called.
- There is no neighbor pair with this *NeighborPairId* in the list.

rsvpServer restartNeighbor neighborPairId

Restarts the RSVP neighbor pair with an identifier of *NeighborPairId.* The results may be accessed using the *rsvpNeighborPair* command. Specific errors are:

- rsvpServer select has not been called.

rsvpServer select *chasID cardID portID*

Accesses the RSVP component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The RSVP protocol package has not been installed.

- Invalid port specified.

rsvpServer NeighborPair *NeighborPairId*

Sets the values for the neighbor pair's entry in the list with an identifier of *NeighborPairId* based on changes made through the *rsvpNeighborPair* command. This command should be used to change a running configuration and must be followed by an *rsvpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A router with this *NeighborPairId* does not exist in the list.

rsvpServer setNeighborPair *NeighborPairId*

Sets the values for the neighbor pair's entry in the list with an identifier of *NeighborPairId* based on changes made through the *rsvpNeighborPair* command. This command should be used to change a running configuration and must be followed by an *rsvpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A neighbor with this *rsvpNeighborPair* does not exist in the list.

rsvpServer setDefault

Sets default values for all configuration options.

rsvpServer write

Writes or commits the changes in IxHAL to hardware for the currently selected chassis, card and port*.* Before using this command, use the *rsvpServer select* command to select the port.

EXAMPLES

```
package req IxTclHal

# Define parameters used by rsvp router

set host localhost

set username user

# Check if we’re running on UNIX - connect to the TCL Server

# which must be running on the chassis

if [isUNIX] {

if [ixConnectToTclServer $host] {

ixPuts “Could not connect to $host”

return 1

}

}

# Now connect to the chassis

if [ixConnectToChassis $host] {

ixPuts $::ixErrorInfo
return 1
}
# Get the chassis ID to use in port lists
set ch [ixGetChassisID $host]
# Port is card 4, port 1
set ch [chassis cget -id]
set ca 4
set po 1
set pl [list [list $ch $ca $po]]
# Login before taking ownership
if [ixLogin $username] {
ixPuts $::ixErrorInfo
return 1
}# Take ownership of the ports we'll use
if [ixTakeOwnership $pl] {
ixPuts $::ixErrorInfo
return 1
}
set myMac {00 0a de 01 01 01}
set router 101.101.9.2
set neighbor 101.101.9.1
# Set up IP address table for others who ARP to us
ipAddressTableItem setDefault
ipAddressTableItem config -numAddresses 1
ipAddressTableItem config -mappingOption oneIpToOneMAC
ipAddressTableItem config -overrideDefaultGateway false
ipAddressTableItem config -fromIpAddress $router
ipAddressTableItem config -fromMacAddress $myMac
ipAddressTable addItem
ipAddressTable set $ch $ca $po
# Select port to operate
if [rsvpServer select $ch $ca $po] {
logMsg "Error in selecting the port $ch $ca $po"
}
```

```
# Clear all neighborPairs
rsvpServer clearAllNeighborPair
# Configure the senderRange for first destinationRange
rsvpSenderRange setDefault
rsvpSenderRange config -enableSenderRange true
rsvpSenderRange config -fromIpAddress {1.1.1.1}
rsvpSenderRange config -rangeCount 1
rsvpSenderRange config -lspIdStart 1
rsvpSenderRange config -lspIdEnd 160
rsvpSenderRange config -peakDataRate 1000000.0
rsvpSenderRange config -enableFastReroute true
rsvpSenderRange config -enableFacilityBackupDesired true
rsvpSenderRange config -enableOneToOneBackupDesired true
#Including a Point of Local Repair (PLR) node
rsvpPlrNodeIdPair setDefault
rsvpPlrNodeIdPair config -avoidNodeId {2.2.2.2}
rsvpPlrNodeIdPair config -plrId {3.3.3.3}
if [rsvpSenderRange addPlr] {
logMsg "Error adding PLR to senderRange"
}
# Add senderRange to destinationRange
if [rsvpDestinationRange addSenderRange senderRange1] {
logMsg "Error adding senderRange senderrange1"
}
# Configure destinationRange
rsvpDestinationRange setDefault
rsvpDestinationRange config -enableDestinationRange true
rsvpDestinationRange config -behavior rsvpIngress
rsvpDestinationRange config -fromIpAddress {2.2.2.2}
rsvpDestinationRange config -rangeCount 1
rsvpDestinationRange config -tunnelIdStart 1
rsvpDestinationRange config -tunnelIdEnd 1
rsvpDestinationRange config -egressBehavior
rsvpEgressUseSEIfInAttribute
```

```
rsvpDestinationRange config -enableReflectRro true
rsvpDestinationRange config -enableFixedLabelForResv false
rsvpDestinationRange config -labelValue
rsvpLabelValueExplicitNull
# Add destinatioanRange to the neighborPair
if [rsvpNeighborPair addDestinationRange destinationRange1] {
logMsg "Error adding destinationRange destinationRange1]
}
# Configure the neighborPair
rsvpNeighborPair setDefault
rsvpNeighborPair config -enableNeighborPair true
rsvpNeighborPair config -ipAddress {1.1.1.1}
rsvpNeighborPair config -dutAddress {2.2.2.2}
# Add the neighborPair to the rsvp server
if [rsvpServer addNeighborPair neighborPair1] {
logMsg "Error in adding neighborPair to the server"
}
# Enable RSVP operation and ARP
protocolServer config -enableArpResponse true
protocolServer config -enableRsvpService true
protocolServer set $ch $ca $po
# Send the data to the hardware
ixWriteConfigToHardware pl
# And start rsvp on the port
ixStartRsvp pl
# Disable destinationRange while rsvp server is runnung.
# This is the same as removing the destination from router
rsvpServer select $ch $ca $po
if [rsvpServer getNeighborPair neighborPair1] {
logMsg "Error getting router1"
}
if [rsvpNeighborPair getDestinationRange destinationRange1] {
logMsg "Error getting routeRange1"
}
```

```
# Disable the route range (You can also change other configuration
if you want)
rsvpDestinationRange config -enableDestinationRange false
if [rsvpNeighborPair setDestinationRange destinationRange1] {
logMsg “Error setting routeRange1”
}
if [rsvpServer write] {
logMsg “Error writing rsvpServer”
}
# Stop the server at the end
ixStopRsvp pl
# Let go of the ports that we reserved
ixClearOwnership $pl
# Disconnect from the chassis we’re using
ixDisconnectFromChassis $host
# If we’re running on UNIX, disconnect from the TCL Server
if [isUNIX] {
ixDisconnectTclServer $host
}
```

SEE ALSO

rsvpNeighborPair, *rsvpDestinationRange*, *rsvpSenderRange*, *rsvpEroItem*, *rsvpRroItem*

NAME - stat

stat — gets the statistics on a port of a card on a chassis.

SYNOPSIS

stat *subcommand options*

DESCRIPTION

The **stat** command is used to get statistics. Statistics may be gathered in several ways. All statistics may be obtained through the use of the *stat get allStats <chassis <card <port* followed by calls to get the data using *stat cget*
-statName. All rate statistics may be obtained through the use of the *stat getRate allStats <chassis <card <port* followed by calls to get the data using *stat cget -statName*.

An individual statistic may be collected through the use of the *stat get statName <chassis <card <port* followed by *stat cget -statName*. Note that the *statName* is formed from the standard option name by prepending *stat* to the name and capitalizing the first letter of the option. (For example, for the option **framesSent**, the *statName* is **statFramesSent.**) The statistic is also available through *stat cget -counterVal* and the corresponding rate is available through *stat cget -counterRate.* No call to *stat getRate* is needed to get the rate.

Values are available through the STANDARD OPTIONS following the *stat cget* call. When using *stat cget -statName*, only those statistics valid for that type of port are returned; all others will return an error (see the *enableValidStats* option). Refer to the *Ixia Reference Manual: Available Statistics* for a list of which statistics are available for particular card modules and under particular circumstances.

STANDARD OPTIONS

Standard Options controlling statistics modes and operation.

enableArpStats true/false

Enables the collection of the protocol server's ARP statistics. **enableProtocolServerStats** must also be set to *true. (default = true)* The following statistics are controlled by this option:

rxArpReply	rxArpReply
txArpReply	txArpRequest

enableBfdStats true/false

Enables the collection of the protocol server's BFD statistics. **enableProtocolServerStats** must also be set to true. The following statistics are controlled by this option:

bfdRoutersConfigured	bfdRoutersRunning
bfdSessionsConfigured	bfdSessionsAutoConfigured
bfdAutoConfiguredSessionsUp	bfdSessionsUp

enableBgpStats true/false

Enables the collection of the protocol server's BGP4 statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

bgpTotalSessions	bgpTotalSessionsEstablished

enableCfmStats true/false

Enables the collection of the protocol server's CFM statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

cfmBridgesConfigured	cfmMepsConfigured
cfmBridgesRunning	cfmMepsRunning
cfmMasConfigured	cfmRemoteMepsLearned
cfmMasRunning	

enableEigrpStats true/false

Enables the collection of the protocol server's EIGRP statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

eigrpNeighborsLearned	eigrpRoutersConfigured
eigrpRoutersRunning	

enableIgmpStats true/false

Enables the collection of the protocol server's IGMP statistics, for the newer IGMPv3 emulation. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

rxIgmpFrames	txIgmpFrames

enableIsisStats true/false

Enables the collection of the protocol server's ISIS statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

isisSessionsConfiguredL1	isisSessionsConfiguredL2
isisSessionsUpL1	isisSessionsUpL2

enableLdpStats true/false

Enables the collection of the protocol server's LDP statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

ldpBasicSessionsUp	ldpSessionsConfigured
ldpSessionsUp	

enableMldStats true/false

Enables the collection of the protocol server's MLD statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

rxMldFrames	txMldFrames

enableOspfStats true/false

Enables the collection of the protocol server's OSPF statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

ospfFullNeighbors	ospfTotalSessions

enableOspfV3Stats true/false

Enables the collection of the protocol server's OSPFv3 statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

ospfV3SessionsConfigured	ospfV3essionsUp

enablePimsmStats true/false

Enables the collection of the protocol server's PIM-SM statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

pimsmNeighborsLearned	pimsmRoutersConfigured
pimsmRoutersRunning	

enableProtocol ServerStats true/false

Master enable for the protocol server. This must be set to *true* for ARP, BGP, EIGRP, ICMP, IGMP, ISIS, LDP, MLD, OSPF, OSPFv3, PIM-SM, RSVP, and STP statistics. *(default = true)* The following statistics are controlled by this option:

protocolServerRx	protocolServerTx
protocolServerVlanDroppedFrames	

enableRsvpStats true/false

Enables the collection of the protocol server's RSVP statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

rsvpEgressLSPsUp	rsvpIngressLSPsConfigured
rsvpIngressLSPsUp	

enableStpStats true/false

Enables the collection of the protocol server's STP statistics. **enableProtocolServerStats** must also be set to *true. (default = false)* The following statistics are controlled by this option:

bpduFramesReceived	bpduFramesSent
mstpBpduFramesReceived	mstpBpduFramesSent

Standard Options used to retrieve statistics

bfdRoutersConfigured

(Read-only.)64-bit value. The total number of BFD routers that are configured.The *enableProtocolServer* and *enableBfdStats* options must be true for this value to be valid.

bfdRoutersRunning

(Read-only.)64-bit value. The total number of BFD routers that are running. The *enableProtocolServer* and *enableBfdStats* options must be true for this value to be valid

bfdSessionsConfigured

(Read-only.) 64-bit value. The total number of BFD sessions that are configured. The *enableProtocolServer* and *enableBfdStats* options must be true for this value to be valid

bfdSessionsAutoConfigured

(Read-only.) 64-bit value. The total number of BFD sessions that are autocreated. The *enableProtocolServer* and *enableBfdStats* options must be true for this value to be valid.

bfdAutoConfiguredSessionsUp

(Read-only.)64-bit value . The *enableProtocolServer* and *enableBfdStats* options must be true for this value to be valid.

bfdSessionsUp

(Read-only.)64-bit value. The total number of BFD sessions that are up. The *enableProtocolServer* and *enableBfdStats* options must be true for this value to be valid.

bpduFramesReceived

(Read-only.) 64-bit value. The total number of RSTP/STP Bridged Protocol Data Units (BPDUs) received. The *enableProtocolServer* and *enableStpStats* options must be *true* for this value to be valid.

bpduFramesSent

(Read-only.)64-bit value. The total number of RSTP/STP Bridged Protocol Data Units (BPDUs) sent. The *enableProtocolServer* and *enableStpStats* options must be *true* for this value to be valid.

cfmBridgesConfigured

(Read-only.) 64-bit value. The total number of CFM bridges that are configured. The *enableProtocolServer* and *enableCfmStats* options must be true for this value to be valid.

cfmBridgesRunning

(Read-only.) 64-bit value. The total number of CFM bridges that are in the run state. The *enableProtocolServer* and *enableCfmStats* options must be true for this value to be valid.

cfmMasConfigured

(Read-only.) 64-bit value. The total number of CFM MAs that are configured. The *enableProtocolServer* and *enableCfmStats* options must be true for this value to be valid.

cfmMasRunning

(Read-only.) 64-bit value. The total number of CFM MAs that are in the run state. The *enableProtocolServer* and *enableCfmStats* options must be true for this value to be valid.

cfmMepsConfigured

(Read-only.)64-bit value. The total number of CFM MEPs that are configured. The *enableProtocolServer* and *enableCfmStats* options must be true for this value to be valid.

cfmMepsRunning

(Read-only.) 64-bit value. The total number of CFM MEPs that are in the run state. The *enableProtocolServer* and *enableCfmStats* options must be true for this value to be valid.

eigrpRemoteMepsLearned

(Read-only.) 64-bit value. The number of CFM Remote MEPs successfully learned. The *enableProtocolServer* and *enableCfmStats* options must be *true* for this value to be valid.

eigrpNeighborsLearned

(Read-only.) 64-bit value. The number of EIGRP neighbor routers successfully learned. The *enableProtocolServer* and *enableEigrpStats* options must be *true* for this value to be valid.

eigrpRoutersConfigured

(Read-only.) 64-bit value. The number of emulated EIGRP routers successfully configured. The *enableProtocolServer* and *enableEigrpStats* options must be *true* for this value to be valid.

eigrpRoutersRunning

(Read-only.) 64-bit value. The number of emulated EIGRP routers in the run state. The *enableProtocolServer* and *enableEigrpStats* options must be *true* for this value to be valid.

isisNeighborsL1

(Read-only.) 64-bit value. The total number of level 1 neighbors. The *enableProtocolServer* and *enableIsisStats* options must be *true* for this value to be valid.

isisNeighborsL2

(Read-only.) 64-bit value. The total number of level 1 neighbors. The *enableProtocolServer* and *enableIsisStats* options must be *true* for this value to be valid.

isisSessionConfigured L1

(Read-only.) 64-bit value. The total number of level 1 configured sessions. The *enableProtocolServer* and *enableIsisStats* options must be *true* for this value to be valid.

isisSessionConfigured L2

(Read-only.) 64-bit value. The total number of level 2 configured sessions. The *enableProtocolServer* and *enableIsisStats* options must be *true* for this value to be valid.

isisSessionsUpL1

(Read-only.) 64-bit value. The total number of level 1 configured sessions that are fully up. The *enableProtocolServer* and *enableIsisStats* options must be *true* for this value to be valid.

isisSessionsUpL2

(Read-only.) 64-bit value. The total number of level 2 configured sessions that are fully up. The *enableProtocolServer* and *enableIsisStats* options must be *true* for this value to be valid.

ldpBasicSessionsUp

(Read-only.) 64-bit value. The number of basic LDP sessions that are up. The *enableProtocolServer* and *enableLdpStats* options must be *true* for this value to be valid.

ldpSessionsConfigured

(Read-only.) 64-bit value. The number of LDP targeted sessions successfully configured. The *enableProtocolServer* and *enableLdpStats* options must be *true* for this value to be valid.

ldpSessionsUp

(Read-only.) 64-bit value. The number of LDP targeted sessions successfully considered running. The *enableProtocolServer* and *enableLdpStats* options must be *true* for this value to be valid.

mstpBpduFramesReceived

(Read-only.)64-bit value. The total number of MSTP Bridged Protocol Data Units (BPDUs) received. The *enableProtocolServer* and *enableStpStats* options must be *true* for this value to be valid.

mstpBpduFramesSent

(Read-only.)64-bit value. The total number of MSTP Bridged Protocol Data Units (BPDUs) sent. The *enableProtocolServer* and *enableStpStats* options must be *true* for this value to be valid

ospfFullNeighbors

(Read-only.) 64-bit value. For OSPF - the number of OSPF neighbors that are fully up. The *enableProtocolServer* and *enableOspfStats* options must be *true* for this value to be valid.

ospfTotalSessions

(Read-only.) 64-bit value. For OSPF - the number of OSPF sessions that were configured. The *enableProtocolServer* and *enableOspfStats* options must be *true* for this value to be valid.

ospfv3Sessions Configured

(Read-only.) 64-bit value. For OSPFv3 - the number of OSPFv3 sessions that were configured. The *enableProtocolServer* and *enableOspfV3Stats* options must be *true* for this value to be valid.

ospfV3SessionsUp

(Read-only.) 64-bit value. For OSPFv3 - the number of OSPFv3 neighbors that are fully up. The *enableProtocolServer* and *enableOspfV3Stats* options must be *true* for this value to be valid.

pimsmNeighbors Learned

(Read-only.) 64-bit value. The number of learned PIM-SM neighbor routers. The *enableProtocolServer* and *enablePimsmStats* options must be *true* for this value to be valid.

pimsmRouters Configured

(Read-only.) 64-bit value. The number of configured PIM-SM routers. The *enableProtocolServer* and *enablePimsmStats* options must be *true* for this value to be valid.

pimsmRoutersRunning

(Read-only.) 64-bit value. The number of PIM-SM routers in the run state. The *enableProtocolServer* and *enablePimsmStats* options must be *true* for this value to be valid.

protocolServerRx

(Read-only.)64-bit value. Number of frames received by the protocol server when it is enabled using the **protocolServer** command. The *enableProtocolServer* option must be *true* for this value to be valid.

protocolServerTx

(Read-only.)64-bit value. Number of frames transmitted by the protocol server when it is enabled using the **protocolServer** command. The *enableProtocolServer* option must be *true* for this value to be valid.

protocolServerVlan Dropped Frames

(Read-only.)64-bit value. Number of VLAN tagged dropped frames from the protocol server. The *enableProtocolServer* option must be *true* for this value to be valid.

rsvpEgressLSPsUp

(Read-only.) 64-bit value. The number of egress LSPs configured and running. The *enableProtocolServer* and *enableRsvpStats* options must be *true* for this value to be valid.

rsvpIngressLSPs Configured

(Read-only.) 64-bit value. The number of ingress LSPs configured. The *enableProtocolServer* and *enableRsvpStats* options must be *true* for this value to be valid.

rsvpIngressLSPsUp

(Read-only.) 64-bit value. The number of ingress LSPs configured and running. The *enableProtocolServer* and *enableRsvpStats* options must be *true* for this value to be valid.

rxArpReply

(Read-only.)64-bit value. Number of ARP reply frames received by the protocol server when it is enabled for ARP using the **protocolServer** command. The *enableProtocolServer* and *enableArpStats* options must be *true* for this value to be valid.

rxArpRequest

(Read-only.)64-bit value. Number of ARP request frames received from a DUT by the protocol server when it is enabled for ARP using the **protocolServer** command. The *enableProtocolServer* and *enableArpStats* options must be *true* for this value to be valid.

rxIgmpFrames

(Read-only.) *64-bit value.* Number of IGMP frames received from a DUT by the protocol server for the newer IGMPv3 compatible emulation when it is enabled for IGMP using the **protocolServer** command. The *enableProtocolServer* and *enableIgmpStats* options must be *true* for this value to be valid.

rxMldFrames

(Read-only.) *64-bit value.* Number of MLD frames received from a DUT by the protocol server when it is enabled for MLD using the **protocolServer** command. The *enableProtocolServer* and *enableMldStats* options must be *true* for this value to be valid.

rxPingReply

(Read-only.)64-bit value. Number of PING reply frames received by the protocol server when it is enabled for PING using the **protocolServer** command. The *enableProtocolServer* and *enableIcmpStats* options must be *true* for this value to be valid.

rxPingRequest

(Read-only.)64-bit value. Number of PING request frames received from a DUT by the protocol server when it is enabled for PING using the **protocolServer** command. The *enableProtocolServer* and *enableIcmpStats* options must be *true* for this value to be valid.

txArpReply

(Read-only.)64-bit value. Number of ARP reply frames transmitted for ARP requests received by the protocol server when it is enabled for ARP using the **protocolServer** command. The *enableProtocolServer* and *enableArpStats* options must be *true* for this value to be valid.

txArpRequest

(Read-only.)64-bit value. Number of ARP reply frames transmitted by the protocol server when it is enabled for ARP using the **protocolServer** command. The *enableProtocolServer* and *enableArpStats* options must be *true* for this value to be valid.

txIgmpFrames

(Read-only.) *64-bit value.* Number of IGMP frames transmitted to the DUT by the protocol server for the newer IGMPv3 compatible emulation when it is enabled for IGMP using the **protocolServer** command. The *enableProtocolServer* and *enableIgmpStats* options must be *true* for this value to be valid.

txMldFrames

(Read-only.) *64-bit value.* Number of MLD frames transmitted to a DUT by the protocol server when it is enabled for MLD using the **protocolServer** command. The *enableProtocolServer* and *enableMldStats* options must be *true* for this value to be valid.

txPingReply

(Read-only.)64-bit value. Number of PING reply frames transmitted for PING requests received by the protocol server when it is enabled for PING using **protocolServer** command. The *enableProtocolServer* and *enableIcmpStats* options must be *true* for this value to be valid.

txPingRequest

(Read-only.)64-bit value. Number of PING request frames transmitted by the protocol server when it is enabled for PING using the **protocolServer** command. The *enableProtocolServer* and *enableIcmpStats* options must be *true* for this value to be valid.

COMMANDS

The **stat** command's subcommands are documented in the *Ixia Tcl Development Guide.*

EXAMPLES

See the *stat*examples in the *Ixia Tcl Development Guide.*

SEE ALSO

*stat*examples in the *Ixia Tcl Development Guide.*

NAME - stpBridge

stpBridge — configures a simulated STP Bridge.

SYNOPSIS

stpBridge *subcommand options*

DESCRIPTION

The *stpBridge* command holds the information related to a single simulated Bridge. Interfaces are added into the *stpBridge* interface list using the *stpBridge addInterface* command. Refer to *STP*84 for an overview.

The following lists are maintained for each bridge:

- Interfaces — a list of interfaces associated with this bridge.
- (For MSTP only) MSTIs — a list of Multiple Spanning Tree Instances (MSTIs) associated with this bridge.
- (For PVST+ and RPVST+ only) VLANs — a list of VLANs associated with this bridge. The first VLAN is put under the Bridge by default, because VLAN 1 (Common Spanning Tree/CST) must be run for all interfaces on the bridge for PVST+/RPVST+.

STANDARD OPTIONS

bridgeMacAddress

The 6-byte Bridge MAC Address for this bridge. Part of the Bridge Identifier. *(default = 0F:00:00:00:00:00)*

bridgePriority

The Bridge Priority for this bridge. Part of the Bridge Identifier. The valid range is 0 to 61440, in increments of 4096. *(default = 32768)*

bridgeSystemId

The System ID for this bridge. Part of the Bridge Identifier. The valid range is 0 to 4095. *(default = 0)*

bridgeType

Type of bridge. If it is 1, it is provider bridges and for 0 it is bridges. (Applicable only for STP, RSTP, and MSTP).

cistRemainingHops

(For use with MSTP only) The remaining Common and Internal Spanning Tree (CIST) hops. The number of additional bridge-to-bridge hops that will be allowed for the MSTP BPDUs. The root sets the maximum hop count, and each subsequent bridge decrements this value by 1. The valid range is 1 to 255. *(default = 20)*

cstRootPriority

(For use with PVST+ and RPSVT+ only) The Common Spanning Tree (CST) priority of the root. The valid range is 0 to 61440, in increments of 4096. *(default = 32768)*

cstRootMacAddress

(For use with PVST+ and RPSVT+ only) The Common Spanning Tree (CST) 6-byte Root MAC Address. *(default = 00:00:00:00:00:00)*

cstRootCost

(For use with PVST+ and RPSVT+ only) The Common Spanning Tree (CST) root path cost. The valid range is 0 to 4294967295. *(default = 0)*

cstVlanPortPriority

(For use with PVST+ and RPSVT+ only) The Common Spanning Tree (CST) VLAN port priority. The valid range is 0 to 63. *(default = 32)*

enable true |false

If set, enables the use of this STP bridge. *(True/False; default = false)*

enableAutoPick BridgeMac

If set, the state machine selects one of the MAC addresses among all of the attached interfaces for a particular emulated bridge as its Bridge MAC Address. *(True/False; default = true)*

extRootCost

(For use with MSTP only) The CIST External root path cost. The valid range is 0 to 4294967295. *(default = 0)*

extRootMacAddress

(For use with MSTP only) The CIST External Root MAC Address. Part of the CIST External Root Identifier. A 6-byte Bridge MAC address. *(default = 00:00:00:00:00:00)*

extRootPriority

(For use with MSTP only) The priority value of the root bridge for the CIST/MSTP region (internal). Part of the CIST Regional Root Identifier. The valid range is 0 to 61440, in increments of 4096. *(default = 32768)*

forwardDelay

The delay used for a port's change to the Forwarding state, in milliseconds. The valid range is 500 msec to 255 sec. *(default = 15,000 msec)*

helloInterval

The length of time between transmission of Hello messages, in milliseconds. The valid range is 500 msec to 255 sec. *(default = 2,000 msec/ 2 sec)*

maxAge

The maximum Configuration message aging time, in milliseconds, set by the root bridge. When a configuration message is received, the message age is incremented. The valid range is 500 msecs to 255 sec. *(default = 20,000 msec)*

messageAge

The message age time parameter in the BPDU, in milliseconds. (It should be less than the Max. Age.) See also *maxAge*. The valid range is 500 msec to 255 sec. *(default = 0)*

mode

The version of the STP protocol that the bridge is using. One of:

Option	Value	Usage
bridgeStp	0	The bridge is using the Spanning Tree Protocol (STP).
bridgeRstp	1	The bridge is using the Rapid Spanning Tree Protocol (RSTP).
bridgeMstp	2	The bridge is using the Multiple Spanning Tree Protocol (MSTP).
bridgePvst	3	The bridge is using the Per-VLAN Spanning Tree Plus Protocol (PVST+).
bridgeRpvst	4	The bridge is using the Per-VLAN Spanning Tree Plus Protocol (MSTP).

mstcConfigName

(For use with MSTP only) The name of the Multiple Spanning Tree Configuration (MSTC) being used. *(Format:MSTC ID-n)*

mstcConfigRevisionNumber

(For use with MSTP only) The revision number of the Multiple Spanning Tree Configuration (MSTC) being used. A 2-byte unsigned integer. *(Default =0)*

name

(Read-only) The name of the bridge which will be used as a unique key to retrieve the object.

portPriority

The port priority. The valid range is 0 to 240, in multiples of 16. (*default = 0)*

regRootCost

(For use with MSTP only) The CIST regional (internal) root path cost. The valid range is 0 to 4294967295. *(default = 0)*

regRootMacAddress

(For use with MSTP only) The CIST Regional Root MAC Address. Part of the CIST Regional Root Identifier. A 6-byte Bridge MAC address. *(Default = 00:00:00:00:00:00)*

regRootPriority

(For use with MSTP only) The Regional Root Priority. The priority value of the root bridge for the CIST/MSTP region (external). Part of the CIST External Root Identifier. The valid range is 0 to 61440, in increments of 4096. *(default = 32768)*

rootCost

(For use with STP and RSTP only) The administrative cost for the shortest path from this bridge to the Root Bridge. The valid range is 0 to 4294967295. *(default = 0)*

rootMacAddress

(For use with STP and RSTP only) The 6-byte MAC Address for the Root Bridge. *(default = 00:00:00:00:00:00)*

rootPriority

(For use with STP and RSTP only) The Bridge Priority for the root bridge. The valid range is 0 to 61440, in increments of 4096. *(default = 32768)*

rootSystemId

The System ID for the root bridge. The valid range is 0 to 4095. *(default = 0)*

COMMANDS

The **stpBridge** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpBridge addInterface *interfaceName*

Adds the router interface described in the *stpInterface* command to the list of interfaces associated with the router. The interface's entry in the list is given an identifier of *interfaceName*. Specific errors are:

- The parameters in *stpInterface* are invalid.
- A router with this *interfaceName* exists already in the list.

NOTE: If an interface is added to an existing bridge and is selected by a *get* command before calling *add*, *stpServer write* can be called immediately without calling *setBridge* command. It will behave as *addOnFly*.

stpBridge addMsti *mstiName*

Adds an MSTP Multiple Spanning Tree Instance (MSTI) to this MSTP Bridge. Bridge mode must first be set to *bridgeMstp*.

stpBridge addVlan *vlanLocalId*

Adds a VLAN to this PVST+ or RPVST+ Bridge. Bridge mode must first be set to *bridgePvst* or *bridgeRpvst*.

stpBridge cistTopologyChange

Generates topology change MSTP BPDUs associated with the CIST, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree.

stpBridge clearAllInterfaces

Deletes all of the interfaces in the interface list for the bridge.

stpBridge clearAllMstis

Deletes all of the MSTP MSTIs in the MSTI list for the MSTP bridge.

stpBridge clearAllVlans

Deletes all of the VLANs in the VLAN list for the PVST+ or RPVST+ bridge.

stpBridge config *option value*

Modify the configuration options of the stpBridge. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for stpBridge.

stpBridge delInterface *[interfaceName]*

Deletes the bridge interface with an identifier of *interfaceName* from the interfaces list of a selected bridge. If no interfaceName is specified, it deletes the current one. Specific errors are:

- No router with this *interfaceName* exists in the list

NOTE:*stpServer write* can be called immediately without calling *setBridge* command. It will behave as *delOnFly*.

stpBridge delMsti *[mstiName]*

Deletes the MSTP MSTI with an identifier of *mstiName* from the interfaces list of a selected bridge. If no interfaceName is specified, it deletes the current one.

stpBridge delVlan *[vlanid]*

Deletes the VLAN with an identifier of *vlanid* from the VLAN list for the PVST+ or RPVST+ bridge. If no vlanid is specified, it deletes the current one.

stpBridge generateTopologyChange

Generates topology change BPDUs, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree.

stpBridge getCistLearnedInfo

Gets learned info list for the MSTP Common and Internal Spanning Tree (CIST). When it returns TCL_OK, it means learned info is returned.

stpBridge getFirstInterface

Gets the first interface in the list. The results may be accessed using the *stpInterface* command. Specific errors are:

- stpBridge select has not been called.
- There are no interfaces in the list.

stpBridge getFirstMsti

Gets the first MSTP MSTI in the list.

stpBridge getFirstMstiLearnedInfo

Gets the learned info list for the first Multiple Spanning Tree Instance (MSTI) in the list and refreshes the options. When it returns TCL_OK, it means learned info is returned.

stpBridge getFirstVlan

Gets the first VLAN from the VLAN list and refreshes the options.

stpBridge getFirstVlanLearnedInfo

Gets the learned info list for the VLAN 1 in the list and refreshes the options. When it returns TCL_OK, it means learned info is returned.

stpBridge getFirstVlanWithCST

Gets the first VLAN, associated with the CST, from the VLAN list and refreshes the options.

stpBridge getInterface *interfaceName*

Gets the interface's entry in the list with an identifier of *interfaceName* and refreshes the options. The bridge interface is accessed in the *stpInterface* command. Specific errors are:

- A router with this *interfaceName* does not exist in the list

stpBridge getLearnedInfo

Gets learned info list. When it returns TCL_OK, it means learned info is returned.

stpBridge getMsti *mstiName*

Gets the MSTI entry in the list with an identifier of *mstiName* and refreshes the options.

stpBridge getNextInterface

Gets the next interface in the list of interfaces for the bridge and refreshes the options. The results may be accessed using the *stpInterface* command. Specific errors are:

- *stpBridge getFirstInterface* has not been called.
- There are no more interfaces in the list.

stpBridge getNextMsti

Gets the next MSTP MSTI in the list of MSTIs for the bridge and refreshes the options.

stpBridge getNextMstiLearnedInfo

Accesses learned info list for the next MSTP MSTI in the list. When it returns TCL_OK, it means learned info is returned.

stpBridge getNextVlan

Accesses the next VLAN in the list of VLANs for the bridge and refreshes the options.

stpBridge getNextVlanLearnedInfo

Gets the learned info list for the next VLAN in the list. When it returns TCL_OK, it means learned info is returned.

stpBridge getVlan *vlanid*

Gets the VLAN entry in the list with an identifier of *Vlanid,* and refreshes the options.

stpBridge requestLearnedInfo

Requests Bridge Learned Info. Specific errors are:

- *stpServer select* has not been called

stpBridge setDefault

Sets default values for all configuration options.

stpBridge setInterface *[interfaceName]*

Edit on the fly "interfaceName" on the selected Bridge. If no interfaceName is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *stpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A bridge with this *interfaceName* does not exist in the list.

stpBridge setMsti *[mstiName]*

Edit on the fly "mstiName" on the selected MSTP Bridge. If no mstiName is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *stpServer write* command in order to send these changes to the protocol server.

stpBridge setVlan *[vlanid]*

Edit on the fly "Vlanid" on the selected PVST+ or RPVST+ Bridge. If no Vlanid is specified, the current one will be modified. This command can be used to change a running configuration and must be followed by an *stpServer write* command in order to send these changes to the protocol server.

stpBridge showInterfaceNames

Returns the names of interfaces in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command. Specific errors are:

- *stpServer select* has not been called.
- There are no interfaces in the list.

stpBridge showMstiNames

Returns the names of MSTP MSTIs in the list on the selected bridge. Calling the *select* command getting the bridge is recommended before calling this command.

stpBridge showVlanNames

Returns the names of VLANs in the list on the selected port. Calling the *select* command is recommended before calling this command.

stpBridge updateInterfaceParameters *[interfaceName]*

Updates the current interface. Get commands should be called before calling the update command. Specific errors are:

- A bridge with this *interfaceName* does not exist in the list

stpBridge updateMstiParameters *[mstiName]*

Updates the current MSTP MSTI. Get commands should be called before calling the update command.

stpBridge updateVlanParameters *[vlanid]*

Updates the current VLAN. Get commands should be called before calling the update command.

EXAMPLES

See examples under *stpServer*.

SEE ALSO

stpBridgeLearnedInfo, *stpInterface*, *stpInterfaceLearnedInfo*, *stpLan*, *stpServer*

NAME - stpBridgeCistLearnedInfo

stpBridgeCistLearnedInfo — views the retrieved learned information for a CIST associated with an advertising MSTP Bridge.

SYNOPSIS

stpBridgeCistLearnedInfo *subcommand options*

DESCRIPTION

The *stpBridgeCistLearnedInfo* makes available learned state information (from the advertising bridge) for the CIST associated with the simulated MSTP Bridge in the current stpBridge command. Refer to STP for an overview of this command.

STANDARD OPTIONS

rootMacAddress

(Read-only.) The 6-byte Root Bridge MAC Address being advertised.

rootPriority

(Read-only.) The Priority being advertised for the Root Bridge.

rootCost

(Read-only.) The cost for the shortest path from the advertising bridge to the Root bridge.

regRootMacAddress

(Read-only.) The 6-byte Regional Root MAC Address being advertised by the bridge.

regRootPriority

(Read-only.) The Regional Root Priority being advertised by the bridge.

regRootCost

(Read-only.) The cost for the shortest path from the advertising bridge to the Regional Root bridge.

COMMANDS

The **stpBridgeCistLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpBridgeCistLearnedInfo cget option

Returns the current value of the configuration option given by *option*.

stpBridgeCistLearnedInfo getFirstInterfaceLearnedInfo

Gets the learned information for the first interface.

stpBridgeCistLearnedInfo getNextInterfaceLearnedInfo

Gets the learned information for the next interface.

stpBridgeCistLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *stpMsti*.

SEE ALSO

stpBridge, *stpInterface*, *stpInterfaceLearnedInfo*, *stpCistInterfaceLearnedInfo*, *stpMsti*, *stpMstiVlan*, *stpBridgeMstiLearnedInfo*, *stpServer*

NAME - stpBridgeLearnedInfo

stpBridgeLearnedInfo — view the retrieved learned information associated with an STP Bridge.

SYNOPSIS

stpBridgeLearnedInfo *subcommand options*

DESCRIPTION

The *stpBridgeLearnedInfo* makes available learned state information from the simulated Bridge in the current *stpBridge* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

bridgeMacAddress

(Read-only.) The 6-byte MAC Address being advertised for the Designated Bridge.

designatedMacAddress

(Read-only.) The 6-byte MAC Address being advertised for the Designated Bridge on the LAN segment.

designatedPortId

(Read-only.) The Port ID value being advertised for the Designated Bridge on the LAN segment.

designatedPriority

(Read-only.) The Priority value being advertised for the Designated Bridge on the LAN segment.

rootCost

(Read-only.) The advertised, administrative cost associated with the Root Bridge.

rootMacAddress

(Read-only.) The advertised 6-byte MAC Address of the Root Bridge.

rootPriority

(Read-only.) The advertised Priority of the Root Bridge.

COMMANDS

The **stpBridgeLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpBridgeLearnedInfo cget *option*

Returns the current value of the configuration option given by *option*.

stpBridgeLearnedInfo getFirstInterfaceLearnedInfo

Gets the First Interface learned info for the selected Bridge.

stpBridgeLearnedInfo getNextInterfaceLearnedInfo

Gets the Next Interface learned info for the selected Bridge.

stpBridgeLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *stpServer*.

SEE ALSO

stpBridge, *stpInterface*, *stpInterfaceLearnedInfo*, *stpLan*, *stpServer*

NAME - stpBridgeMstiLearnedInfo

stpBridgeMstiLearnedInfo — views the retrieved learned information for an MSTI associated with an MSTP Bridge.

SYNOPSIS

stpBridgeMstiLearnedInfo *subcommand options*

DESCRIPTION

The *stpBridgeMstiLearnedInfo* makes available learned state information (from the advertising bridge) for the MSTI associated with the simulated Bridge in the current *stpBridge* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

mstiId

(Read-only.) The advertised MSTI identifier.

rootMacAddress

(Read-only.) The advertised 6-byte Root Bridge MAC Address.

rootPriority

(Read-only.) The Priority being advertised for the Root Bridge.

rootCost

(Read-only.) The cost for the shortest path from the advertising bridge to the Root bridge.

COMMANDS

The **stpBridgeMstiLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpBridgeMstiLearnedInfo cget *option*

Returns the current value of the configuration option given by *option*.

stpBridgeMstiLearnedInfo getFirstInterfaceLearnedInfo

Gets the learned information for the first interface.

stpBridgeMstiLearnedInfo getNextInterfaceLearnedInfo

Gets the learned information for the next interface.

stpBridgeMstiLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *stpMsti*.

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpInterfaceLearnedInfo*, *stpCistInterfaceLearnedInfo*, *stpMsti*, *stpMstiVlan*, *stpServer*

NAME - stpBridgeVlanLearnedInfo

stpBridgeVlanLearnedInfo — views the retrieved learned information for a VLAN associated with an advertising PVST+ or RPVST+ Bridge.

SYNOPSIS

stpBridgeVlanLearnedInfo *subcommand options*

DESCRIPTION

The *stpBridgeVlanLearnedInfo* makes available learned state information (from the advertising bridge) for the VLAN associated with the simulated PVST+ or RPVST+ Bridge in the current *stpBridge* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

rootMacAddress

(Read-only.) The 6-byte Root Bridge MAC Address being advertised.

rootPriority

(Read-only.) The Priority being advertised for the Root Bridge.

rootCost

(Read-only.) The cost for the shortest path from the advertising bridge to the Root bridge.

vlanId

(Read-only.) The VLAN identifier being advertised.

COMMANDS

The **stpBridgeVlanLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpBridgeVlanLearnedInfo getFirstVlanLearnedInfo

Gets the learned information for VLAN 1.

stpBridgeVlanLearnedInfo getNextVlanLearnedInfo

Gets the learned information for the next VLAN.

stpBridgeVlanLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *stpVlan*.

SEE ALSO

stpBridge, *stpInterface*, *stpInterfaceLearnedInfo*, *stpServer*, *stpVlan*, *stpVlanInterfaceLearnedInfo*

NAME - stpCistInterfaceLearnedInfo

stpCistInterfaceLearnedInfo — views the retrieved learned information for a CIST interface associated with an MSTP Bridge.

SYNOPSIS

stpCistInterfaceLearnedInfo *subcommand options*

DESCRIPTION

The *stpCistInterfaceLearnedInfo* makes available learned state information (from the advertising bridge) for the CIST interface associated with the simulated MSTP Bridge in the current *stpBridge* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

interfaceRole

(Read-only.) The advertised Role of the Interface. One of:

Option	Value	Usage
stpInterfaceRoleDisabled	0	The interface is disabled.
stpInterfaceRoleRoot	1	The interface has the shortest path to the root bridge from the current bridge.
stpInterfaceRoleDesignated	2	The interface is on the LAN segment.
stpInterfaceRoleAlternate	3	The interface is not currently part of the active topology, but it could function as an alternate for the designated interface on this bridge.
stpInterfaceRoleBackup	4	The interface is not currently part of the active topology, but it could function as a backup for the root interface on this bridge.

interfaceState

(Read-only.) The advertised State of the Interface. One of:

Option	Value	Usage
stpInterfaceStateDiscarding	0	The interface is discarding MAC frames.
stpInterfaceStateLearning	1	The interface is learning MAC addresses.
stpInterfaceStateForwarding	2	The interface is forwarding MAC frames.

designatedPriority

(Read-only.) The designated priority being advertised.

designatedMacAddress

(Read-only.) The designated MAC address being advertised for the interface.

designatedPortId

(Read-only.) The designated Port ID being advertised.

protocolInterfaceDescription

(Read-only.) The protocol Interface being advertised.

COMMANDS

The **stpCistInterfaceLearnedInfo** command is invoked with the following sub-commands. If no subcommand is specified, returns a list of all subcommands available.

stpCistInterfaceLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *stpMsti*.

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpInterfaceLearnedInfo*, *stpCistInterfaceLearnedInfo*, *stpMsti*, *stpMstiVlan*, *stpServer*

NAME - stpInterface

stpInterface — sets up the parameters associated with an STP interface.

SYNOPSIS

stpInterface *subcommand options*

DESCRIPTION

The *stpInterface* command holds the information related to a single interface on the simulated bridge. Interfaces are added into the *stpInterface* interface list using the *stpBridge* addInterface command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

cost

The administrative path cost assigned to this STP interface. A 4-byte unsigned integer. *(Default = 1)*

enable ***true |false***

If set, enables the use of this interface. *(Default = false)*

enableJitter

If set, then the jitter feature is enabled. *(Default = true)*

enableAutoPickPortNum

If set, then the Auto-Pick Port Number feature is enabled and each STP interface configured for the same bridge will be assigned a unique port number automatically. Additional port numbers for interfaces on the same bridge will be incremented by 1. *(Default = true)*

interBpduGap

The length of time between transmissions of BPDUs. In milliseconds. *(Valid range: 0 msec to 60,000 msec; default = 0 sec)*

interfaceType

The type of STP interface, one of: Shared Link or Point-to-Point. *(Default = Shared Link)*

jitterPercentage

For the Hello Timer. The maximum percentage of +/- variation (jitter) from the Hello message transmission interval.

mstiId

The identifier of the MSTP MSTI. An unsigned integer. *(Valid range: 1 to 4,095)*

name

(Read-only) Name of the interface that will be used as a unique key to retrieve the object.

portNum

The port number associated with the STP interface. If enableAutoPickPortNum is set, each STP interface configured for the same bridge will be assigned a unique port number automatically. If it is not set, the user may assign a value for the port number. An unsigned integer. *(Valid range: 1 to 4,095)*

protocolInterface Description

The *description* option associated with an *stpInterface* when it was created. The IP address and mask are read from the interface entry.

COMMANDS

The **stpInterface** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpInterface setDefault

Sets default values for all options.

EXAMPLES

See examples under *stpServer*.

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterfaceLearnedInfo*, *stpLan*, *stpServer*

NAME - stpInterfaceLearnedInfo

stpInterfaceLearnedInfo — views the retrieved learned information for the parameters associated with an advertising STP Interface.

SYNOPSIS

stpInterfaceLearnedInfo *subcommand options*

DESCRIPTION

The *stpInterfaceLearnedInfo* makes available learned state information from the advertising STP Interface. Refer to STP for an overview of this command.

STANDARD OPTIONS

designatedMacAddress

(Read-only.) The advertised 6-byte MAC Address of the Designated Bridge on the LAN segment.

designatedPortId

(Read-only.) The advertised Port ID of the Designated Bridge's designated Port on the LAN segment.

designatedPriority

(Read-only.) The advertised Priority of the Designated Bridge on the LAN segment.

interfaceRole

(Read-only.) The advertised Role of the Interface. One of:

Option	Value	Usage
stpInterfaceRoleDisabled	0	The interface is disabled.
stpInterfaceRoleRoot	1	The interface has the shortest path to the root bridge from the current bridge.
stpInterfaceRoleDesignated	2	The interface is on the LAN segment.
stpInterfaceRoleAlternate	3	The interface is not currently part of the active topology, but it could function as an alternate for the designated interface on this bridge.
stpInterfaceRoleBackup	4	The interface is not currently part of the active topology, but it could function as a backup for the root interface on this bridge.

interfaceState

(Read-only.) The advertised State of the Interface. One of:

Option	Value	Usage
stpInterfaceStateDiscarding	0	The interface is discarding MAC frames.

Option	Value	Usage
stpInterfaceStateLearning	1	The interface is learning MAC addresses.
stpInterfaceStateForwarding	2	The interface is forwarding MAC frames.

protocolInterface Description

(Read-only.) The descriptive identifier of the advertised protocol interface.

rootCost

(Read-only.) The advertised administrative cost of the path to the Root Bridge.

rootMacAddress

(Read-only.) The advertised 6-byte MAC Address of the Root Bridge.

rootPriority

(Read-only.) The advertised Priority value of the Root Bridge.

COMMANDS

The **stpInterfaceLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpInterfaceLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under *stpServer*.

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpLan*, *stpServer*

NAME - stpLan

stpLan — sets up the parameters associated with an STP LAN.

SYNOPSIS

stpLan *subcommand options*

DESCRIPTION

The *stpLan* command describes a Local Area Network (LAN) associated with an STP Bridge. STP LANs are added into *stpLan* lists using the *stpServer* *addLan* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

count

The number of MAC Addresses in the range. *(Valid range: 1 to 500.)*

enable true |false

If set, enables the use of this STP LAN. *(Default = false)*

enableVlan true |false

If set, enables the use of VLANs in this STP LAN. *(Default = false)*

incrementMacAddress true |false

If set, enables the incrementing of the MAC address. *(Default = false)*

incrementVlan true |false

If set, enables the incrementing of VLANs. *(Default = false)*

startMacAddress

The first MAC Address in the range. A 6-byte MAC address. *(Default = 00:00:00:00:00:00)*

vlanId

The first VLAN identifier. *(Default = 0)*

COMMANDS

The **stpLan** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpLan setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *stpServer*.

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpServer*

NAME - stpMsti

stpMsti — sets up the parameters associated with an MSTP MSTI.

SYNOPSIS

stpMsti *subcommand options*

DESCRIPTION

The *stpMsti* command describes a Multiple Spanning Tree Instance (MSTI) associated with an MSTP Bridge. MSTP MSTIs are added into *stpBridge* lists using the *stpBridge* *addMsti* command. Refer to *STP* for an overview of this command.

The following list is maintained for each MSTI:

- MSTI VLAN Ranges — a list of MSTI VLAN Ranges associated with this *stpMsti*, and constructed with the *stpMstiVlan* command.

STANDARD OPTIONS

enable true |false

If set, enables the use of this MSTI. *(Default = false)*

mstiHops

The number of remaining MSTI hops. *(Valid range: 1 to 255; default = 20)*

mstiId

The identifier for this MST Instance (MSTI). An unsigned integer. *(Valid range: 1 to 4,094)*

mstiInternalRoot PathCost

The cost associated with the MSTI Internal Root Path. A 4-byte unsigned integer. *(Default = 0)*

mstiName

The name of this particular MSTI, which is configured from the list of MSTIs. *(Format: MSTI ID-n)* (Editable by the user)

mstiRegionalRootId

The Regional Root ID value for this MSTI. A 6-byte MAC address. *(Default = 00:00:00:00:00:00)*

mstiRootPriority

The MSTI Root Priority. This is part of the MSTI Regional Root Identifier. An unsigned integer; a multiple of 4096. *(Valid range: 0 to 61440; default = 32768)*

name

(Read-only) The name of the MSTI that will be used as a unique key to retrieve the object.

portPriority

The MSTI Port Priority. This is part of the MSTI Regional Root Identifier. An unsigned integer; a multiple of 16. *(Valid range: 0 to 240; default = 0)*

RemoveAllVlans true |false

If true, removes all VLAN information from the packet. *(default = false)*

COMMANDS

The **stpMsti** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpMsti addVlanRange

Add a MSTI VLAN range to the list.

stpMsti clearAllVlans

Clears all VLANs from the list.

stpMsti configure *option value*

Modify the configuration options of the stpMsti. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for stpMsti.

stpMsti generateTopologyChange

Generates Topology Change BPDUs, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree.

stpMsti getFirstVlanRange

Get the first MSTI VLAN Range in the list.

stpMsti getNextVlanRange

Get the next MSTI VLAN Range in the list.

stpMsti getVlanRange

Get the MSTI VLAN Range in the list.

stpMsti setDefault

Sets default values for all configuration options.

stpMsti setVlanRange

This command may be used to change an individual MSTI VLAN range on the fly, while the STP protocol is running. Changes to the MSTI VLAN range are made with the *stpMstiVlan* command. This command must be followed with a call to *stpServer* write.

stpMsti showVlanRangesNames

Returns the names of MSTI VLAN ranges in the list on the selected port. Calling the *select* command is recommended before calling this command.

EXAMPLES

```
# CONFIGURATION : One Port is configured with TWO Bridges, each
with one
Interface, and one MSTI (they are same Region).
# GET THE CHASSIS LOOPBACK INTERFACE
chassis get “loopback”
set chassis [chassis cget -id]
# GET THE CARD ID
set card 2
card setDefault
card config -txFrequencyDeviation 0
card set $chassis $card
card write $chassis $card
# GET THE PORT Port-1
set port 1
# SET UP THE INTERFACE TABLE FOR THE PORT
interfaceTable select $chassis $card $port
interfaceTable setDefault
interfaceTable config -dhcpV4RequestRate 0
interfaceTable config -dhcpV6RequestRate 0
interfaceTable set
interfaceTable clearAllInterfaces
# CONFIGURE THE TWO INTERFACES
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description
{ProtocolInterface - 02:01 - 1}
interfaceEntry config -macAddress {00 00 34 53
6C AA}
```

```
interfaceEntry config -eui64Id {02 00 34 FF
FE 53 6C AA}
interfaceEntry config -atmEncapsulation
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmMode -1
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 0
interfaceEntry config -vlanPriority 0
interfaceEntry config -enableDhcpV6 false
interfaceTable addInterface interfaceTypeConnected
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
interfaceEntry config -enable true
interfaceEntry config -description
{ProtocolInterface - 02:01 - 2}
interfaceEntry config -macAddress {00 00 34
53 6C AB}
interfaceEntry config -eui64Id {02 00 34
FF FE 53 6C AB}
interfaceEntry config -atmEncapsulation
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmMode -1
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 0
interfaceEntry config -vlanPriority 0
interfaceEntry config -enableDhcpV6 false
interfaceTable addInterface interfaceTypeConnected
```

```
# SELECT THE PORT TO OPERATE
stpServer select $chassis $card $port
# CLEAR ALL BRIDGES AND LANS
stpServer clearAllBridges
stpServer clearAllLans
# CONFIGURE AND ADD FIRST PROTOCOL INTERFACE
stpInterface setDefault
stpInterface config -enable true
stpInterface config -cost 1
stpInterface config -interBpduGap 0
stpInterface config -enableJitter false
stpInterface config -jitterPercentage 0
stpInterface config -interfaceType
stpInterfacePointToPoint
stpInterface config -mstiId -1
stpInterface config -portNum 1
stpInterface config -enableAutoPickPortNum true
stpInterface config -protocolInterfaceDescription
"ProtocolInterface - 02:01 - 1"
stpBridge addInterface Interface1
# CONFIGURE AND ADD FIRST MSTI UNDER FIRST PROTOCOL INTERFACE
stpMsti setDefault
stpMsti config -enable true
stpMsti config -mstiId 1
stpMsti config -mstiRegionalRootId
"00:00:00:00:00:00"
stpMsti config -mstiRootPriority 32768
stpMsti config -mstiInternalRootPathCost 0
stpMsti config -startVLANid 1
stpMsti config -endVLANid 1
stpMsti config -mstiName "MSTI
ID-1"
stpMsti config -mstiHops 20
stpMsti config -portPriority 0
```

```
stpBridge addMsti Msti1
# CONFIGURE AND ADD BRIDGE1 TO STP SERVER
stpBridge setDefault
stpBridge config -enable true
stpBridge config -mode
bridgeMstp
stpBridge config -bridgeMacAddress
“00:00:02:01:00:01”
stpBridge config -bridgeSystemId 0
stpBridge config -bridgePriority 32768
stpBridge config -enableAutoPickBridgeMac true
stpBridge config -rootMacAddress
“00:00:00:00:00:00”
stpBridge config -rootSystemId 0
stpBridge config -rootPriority 32768
stpBridge config -rootCost 0
stpBridge config -helloInterval 2000
stpBridge config -forwardDelay 15000
stpBridge config -maxAge 20000
stpBridge config -portPriority 0
stpBridge config -extRootPriority 32768
stpBridge config -extRootMacAddress
“00:00:00:00:00:00”
stpBridge config -extRootCost 0
stpBridge config -regRootPriority 32768
stpBridge config -regRootMacAddress
“00:00:00:00:00:00”
stpBridge config -regRootCost 0
stpBridge config -mstcConfigName “MSTC
ID-1”
stpBridge config -mstcConfigRevisionNumber 0
stpBridge config -messageAge 0
stpBridge config -cistRemainingHops 20
stpServer addBridge Bridge1
```

```
# CONFIGURE AND ADD SECOND PROTOCOL INTERFACE
stpInterface setDefault
stpInterface config -enable true
stpInterface config -cost 1
stpInterface config -interBpduGap 0
stpInterface config -enableJitter false
stpInterface config -jitterPercentage 0
stpInterface config -interfaceType
stpInterfacePointToPoint
stpInterface config -mstiId -1
stpInterface config -portNum 1
stpInterface config -enableAutoPickPortNum true
stpInterface config -protocolInterfaceDescription
"ProtocolInterface - 02:01 - 2"
stpBridge addInterface Interface2
# CONFIGURE AND ADD FIRST MSTI UNDER SECOND PROTOCOL INTERFACE
stpMsti setDefault
stpMsti config -enable true
stpMsti config -mstiId 1
stpMsti config -mstiRegionalRootId
"00:00:00:00:00:00"
stpMsti config -mstiRootPriority 32768
stpMsti config -mstiInternalRootPathCost 0
stpMsti config -startVLANid 1
stpMsti config -endVLANid 1
stpMsti config -mstiName "MSTI
ID-1"
stpMsti config -mstiHops 20
stpMsti config -portPriority 0
stpBridge addMsti Msti1
# CONFIGURE AND ADD BRIDGE2
stpBridge setDefault
stpBridge config -enable true
stpBridge config -mode
```

```
bridgeMstp
stpBridge config -bridgeMacAddress
"00:00:02:01:00:02"
stpBridge config -bridgeSystemId 0
stpBridge config -bridgePriority 32768
stpBridge config -enableAutoPickBridgeMac true
stpBridge config -rootMacAddress
"00:00:00:00:00:00"
stpBridge config -rootSystemId 0
stpBridge config -rootPriority 32768
stpBridge config -rootCost 0
stpBridge config -helloInterval 2000
stpBridge config -forwardDelay 15000
stpBridge config -maxAge 20000
stpBridge config -portPriority 0
stpBridge config -extRootPriority 32768
stpBridge config -extRootMacAddress
"00:00:00:00:00:00"
stpBridge config -extRootCost 0
stpBridge config -regRootPriority 32768
stpBridge config -regRootMacAddress
"00:00:00:00:00:00"
stpBridge config -regRootCost 0
stpBridge config -mstcConfigName "MSTC
ID-2"
stpBridge config -mstcConfigRevisionNumber 0
stpBridge config -messageAge 0
stpBridge config -cistRemainingHops 20
stpServer addBridge Bridge2
# LET THE PROTOCOL SERVER RESPOND TO STP
protocolServer config -enableStpService true
protocolServer set $chassis $card $port
# START AND SELECT MSTP PROTOCOL
set portList {}
```

```
lappend portList [list 1 2 1]
ixStartStp portList
stpServer select $chassisId $cartdId $portID
# REQUEST FOR LEARNED INFO
stpBridge requestLearnedInfo
# GET CIST LEARNED INFO
stpBridge getCistLearnedInfo
showCmd stpBridgeCistLearnedInfo
# GET CIST INTERFACE LEARNED INFO
stpBridgeCistLearnedInfo getFirstInterfaceLearnedInfo
showCmd stpCistInterfaceLearnedInfo
# GET MSTI LEARNED INFO
stpBridge getFirstMstiLearnedInfo
showCmd stpBridgeMstiLearnedInfo
# GET MSTI INTERFACE LEARNED INFO
stpBridgeMstiLearnedInfo getFirstInterfaceLearnedInfo
showCmd stpMstiInterfaceLearnedInfo
# Disable MSTI1 on BRIDGE1
stpServer getBridge Bridge1
stpBridge getMsti Msti1
stpMsti config -enable false
stpBridge setMsti
stpServer setBridge Bridge1
stpServer write
# Configure MSTI1 with -mstiHops to a New Value
stpBridge getMsti Msti1
stpMsti config -mstiHops 16
stpBridge setMsti
stpBridge updateMstiParameters
stpServer write
# Configure a New Value for First PROTOCOL INTERFACE UNDER BRIDGE1
stpBridge getFirstInterface
stpInterface config -interBpduGap 13
stpBridge setInterface
```

```
stpServer write

# Configure a New Value of Hello Interval of Bridge1

stpBridge config -helloInterval 1000

stpServer setBridge

stpServer write

# MSTI TOPOLOGY CHANGE

stpMsti generateTopologyChnage

# CIST TOPOLOGY CHANGE

stpBridge cistTopologyChange

# BRIDGE TOPOLOLOGY CHANGE

stpBridge generateTopologyChange

# DELETE MSTI1 under Bridge1

stpServer getBridge Bridge1

stpBridge delMsti Msti1

stpServer write
```

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpInterfaceLearnedInfo*, *stpLan*, *stpServer*, *stpMstiVlan*, *stpMstiInterfaceLearnedInfo*, *stpBridgeMstiLearnedInfo*, *stpBridgeCistLearnedInfo*, *stpCistInterfaceLearnedInfo*

NAME - stpMstiInterfaceLearnedInfo

stpMstiInterfaceLearnedInfo — views the retrieved learned information for the parameters associated with an MSTP MSTI Interface.

SYNOPSIS

stpMstiInterfaceLearnedInfo *subcommand options*

DESCRIPTION

The *stpMstiInterfaceLearnedInfo* makes available learned state information from the advertising MSTP MSTI Interface. Refer to STP for an overview of this command.

STANDARD OPTIONS

designatedMac

(Read-only.) The advertised 6-byte MAC Address of the Designated Bridge on the LAN segment.

designatedPortId

(Read-only.) The advertised Port ID of the Designated Bridge's designated Port on the LAN segment.

designatedPriority

(Read-only.) The advertised Priority of the Designated Bridge on the LAN segment.

interfaceRole

(Read-only.) The advertised Role of the Interface. One of:

Option	Value	Usage
stpInterfaceRoleDisabled	0	The interface is disabled.
stpInterfaceRoleRoot	1	The interface has the shortest path to the root bridge from the current bridge.
stpInterfaceRoleDesignated	2	The interface is on the LAN segment.
stpInterfaceRoleAlternate	3	The interface is not currently part of the active topology, but it could function as an alternate for the designated interface on this bridge.
stpInterfaceRoleBackup	4	The interface is not currently part of the active topology, but it could function as a backup for the root interface on this bridge.
stpinterfaceRoleMaster	5	The MSTI interface role has been set as Master, based on all of the BPDUs exchanged between the Ixia port and the DUT.

interfaceState

(Read-only.) The advertised State of the Interface. One of:

Option	Value	Usage
stpInterfaceStateDiscarding	0	The interface is discarding MAC frames.
stpInterfaceStateLearning	1	The interface is learning MAC addresses.
stpInterfaceStateForwarding	2	The interface is forwarding MAC frames.

protocolInterface Description

(Read-only.) The advertised descriptive identifier of the protocol interface.

COMMANDS

The **stpMstiInterfaceLearnedInfo** command is invoked with the following sub-commands. If no subcommand is specified, returns a list of all subcommands available.

stpMstiInterfaceLearnedInfo setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under stpMsti.

SEE ALSO

stpBridge, *stpMsti*, *stpInterface*, *stpServer*, *stpBridgeMstiLearnedInfo*

NAME - stpMstiVlan

stpMstiVlan — sets up the parameters associated with an MSTP MSTI VLAN range.

SYNOPSIS

stpMstiVlan *subcommand options*

DESCRIPTION

The *stpMstiVlan* command describes a Virtual LAN (VLAN) range associated with an MSTP MSTI. MSTI VLAN ranges are added into *stpMsti* lists using the *stpMsti* *addVlanRange* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

endVlanId

The last MSTI VLAN Id in the MSTI VLAN range. An unsigned integer. *(Valid range: 1 to 4094)*

startVlanId

The first MSTI VLAN Id in the MSTI VLAN range. An unsigned integer. *(Valid range: 1 to 4094)*

COMMANDS

The **stpMstiVlan** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpMstiVlan config *option value*

Modify the configuration options of the *stpMstiVlan*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for *stpMstiVlan*.

stpMstiVlan setDefault

Sets default values for all configuration options.

EXAMPLES

See examples under *stpMsti*.

SEE ALSO

stpBridge, *stpMsti*, *stpInterface*, *stpServer*, *stpBridgeMstiLearnedInfo*, *stpMstiInterfaceLearnedInfo*

NAME - stpServer

stpServer — accesses the STP component of the protocol server for a particular port.

SYNOPSIS

stpServer *subcommand options*

DESCRIPTION

The *stpServer* command is necessary in order to access the STP protocol server for a particular port. The *select* subcommand **must** be used before all other STPcommands. Refer to *STP* for an overview.

STANDARD OPTIONS

N/A

COMMANDS

The **stpServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpServer addbridge *bridgeName*

Adds the STP bridge described in the *stpBridge* command to the list of bridges associated with the port. The bridge's entry in the list is given an identifier of *bridgeName*. Specific errors are:

- stpServer select has not been called.
- The bridge parameters in *stpBridge* are invalid.
- A bridge with this bridgeName exists already in the list.

stpServer addLan *lanName*

Adds the STP LAN described in the *stpLan* command to the list of LANs associated with the port. The LAN's entry in the list is given an identifier of *lanName*. Specific errors are:

- stpServer select has not been called.
- The LAN parameters in *stpLan* are invalid.
- A LAN with this lanName exists already in the list.

stpServer clearAllBridges

Deletes all the STP bridges in the list. Specific errors are:

- stpServer select has not been called.
- There is no bridge with this *bridgeName* in the list.

stpServer clearAllLans

Deletes all the STP LANs in the list. Specific errors are:

- stpServer select has not been called.
- There is no LAN with this *lanName* in the list.

stpServer delBridge *[bridgeName]*

Deletes the STP bridge described that has an identifier of *bridgeName*. Specific errors are:

- stpServer select has not been called.
- There is no bridge with this *bridgeName* in the list.

stpServer delLan *[lanName]*

Deletes the STP LAN described that has an identifier of *lanName*. Specific errors are:

- stpServer select has not been called.
- There is no LAN with this *lanName* in the list.

stpServer get

Gets the current configuration of the protocol server for the last selected port.

stpServer getBridge *bridgeName*

Access the STP bridge with an identifier of *bridgeName.* The results may be accessed using the *stpBridge* command. Specific errors are:

- stpServer select has not been called.
- There is no bridge with this *bridgeName* in the list.

stpServer getFirstBridge

Access the first STP bridge in the list*.* The results may be accessed using the *stpBridge* command. Specific errors are:

- stpServer select has not been called.
- There are no bridges in the list.

stpServer getFirstLan

Access the first STP LAN in the list*.* The results may be accessed using the *stpLan* command. Specific errors are:

- stpServer select has not been called.
- There are no LANs in the list.

stpServer getLan *lanName*

Access the STP LAN with an identifier of *lanName.* The results may be accessed using the *stpLan* command. Specific errors are:

- stpServer select has not been called.
- There is no LAN with this *lanName* in the list.

stpServer getNextBridge

Access the next STP bridge in the list. The results may be accessed using the *stpBridge* command. Specific errors are:

- stpServer select has not been called.
- stpServer getFirstBridge has not been called.
- There is no more bridges in the list.

stpServer getNextLan

Access the next STP LAN in the list. The results may be accessed using the *stpLan* command. Specific errors are:

- stpServer select has not been called.
- stpServer getFirstLan has not been called.
- There is no more LANs in the list.

stpServer select *chasID cardID portID*

Accesses the STP component of the protocol server for the indicated port. Specific errors are:

- No connection to the chassis.
- The STP protocol package has not been installed.
- Invalid port specified.

stpServer set

Sets the current configuration of the protocol server on the most recently selected port to its hardware. Call this command before calling *stpServer* cget *option value* to get the value of the configuration option. Specific errors are:

- No connection to a chassis.
- Invalid port number.
- The port is being used by another user.
- The configured parameters are not valid for this port.

stpServer setDefault

Sets default values for all configuration options.

stpServer setBridge *[bridgeName]*

Sets the values for the bridge's entry in the list with an identifier of *bridgeName* based on changes made through the *stpBridge* command. *bridgeName* may only be omitted if *getFirstBridge* and *getNextBridge* were used to select the bridge, in which case the currently selected bridge is set. This command should be used to change a running configuration and must be followed by an *stpServer write* command in order to send these changes to the protocol server. Specific errors are:

- A bridge with this *bridgeName* does not exist in the list.
- Argument is omitted and no bridge is currently selected.

stpServer setLan *[lanName]*

Edit on the fly "lanName." If no lanName is specified, the current one will be modified.

stpServer showBridgeNames

Returns names of bridges in the list on the selected port. Calling select command is recommended before calling this command.

stpServer showLanNames *[lanName]*

Returns names of LANs in the list on the selected port. Calling select command is recommended before calling this command.

stpServer updateBridgeParameters *[bridgeName]*

Updates the current bridge. Get commands need to be called before calling this command.

stpServer write

Sends any changes made with *stpBridge* *setInterface,* or *stpServer* *setBridge* to the protocol server for immediate application. This command **must** be used after those mentioned above in order for their changes to have an effect.

EXAMPLES

```
# Set up the interface table for the port

interfaceTable select $chassis $card $port

interfaceTable clearAllInterfaces

interfaceEntry clearAllItems addressTypeIpV6

interfaceEntry clearAllItems addressTypeIpV4

interfaceIpV4 setDefault

interfaceIpV4 config -gatewayIpAddress

{31.0.1.1}

interfaceIpV4 config -maskWidth 24

interfaceIpV4 config -ipAddress {31.0.1.2}

interfaceEntry addItem addressTypeIpV4

# Configure the interface

interfaceEntry setDefault

interfaceEntry config -enable true

interfaceEntry config -description \

{ProtocolInterface - 06:01 - 1}

interfaceEntry config -macAddress \

{00 00 00 59 20 39}

interfaceEntry config -eui64Id \
{02 00 00 FF FE 59 20 39}
interfaceEntry config -atmEncapsulation \
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmMode -1
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 0
interfaceEntry config -vlanPriority 0
interfaceTable addInterface interfaceTypeConnected
# Select the port to operate
stpServer select $chassis $card $port
# Clear all bridges
stpServer clearAllBridges
# Clear all LANs
stpServer clearAllLans
# Configure LAN 1
stpLan setDefault
stpLan config -enable true
stpLan config -startMacAddress \
"00:00:00:04:45:55"
stpLan config -count 10
# Add LAN 1 to the STP server
stpServer addLan Lan1
# Configure LAN 2
stpLan setDefault
stpLan config -enable true
stpLan config -startMacAddress \
"00:00:00:00:77:88"
stpLan config -count 20
# Add LAN 2 to the STP server
stpServer addLan Lan2
# Configure the STP interface
```

```
stpInterface setDefault
stpInterface config -enable true
stpInterface config -cost 12
stpInterface config -interBpduGap 25
stpInterface config -enableJitter true
stpInterface config -jitterPercentage 15
stpInterface config -interfaceType \
stpInterfacePointToPoint
stpInterface config -protocolInterfaceDescription \
“ProtocolInterface - 06:01 - 1”
# Add the STP interface to the bridge
stpBridge addInterface stpBridgeInterface1
# Configure the bridge
stpBridge setDefault
stpBridge config -enable true
stpBridge config -mode
bridgeRstp
stpBridge config -bridgeMacAddress \
“00:00:06:01:00:01”
stpBridge config -bridgeSystemId 0
stpBridge config -bridgePriority 32768
stpBridge config -enableAutoPickBridgeMac true
stpBridge config -rootMacAddress \
“99:88:77:66:55:44”
stpBridge config -rootSystemId 0
stpBridge config -rootPriority 32768
stpBridge config -rootCost 10
stpBridge config -helloInterval 2000
stpBridge config -forwardDelay 15000
stpBridge config -maxAge 20000
# Add the bridge to the STP server
stpServer addBridge Bridge1
# Let the protocol server respond to ARP, STP
protocolServer setDefault
protocolServer config -enableArpResponse true

protocolServer config -enableStpService true

protocolServer set $chassis $card $port
```

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpInterfaceLearnedInfo*, *stpLan*

NAME - stpVlan

stpVlan — sets up the parameters associated with a PVST+ or RPVST+ VLAN.

SYNOPSIS

stpVlan *subcommand options*

DESCRIPTION

The *stpVlan* command describes a VLAN associated with a PVST+ or RPVST+ Bridge. STP VLANs are added into *stpBridge* lists using the *stpBridge* *addVlan* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

enable true |false

If set, enables the use of this STP VLAN. *(default = false)*

vlanId

The identifier for this Virtual LAN (VLAN). The valid range is 2 to 4094. *(default = 2)*

portMacAddress

The MAC Address for the port associated with this VLAN. *(default = 00:00:00:00:00:00)*

portPathCost

The cost associated with the Path. The valid range is 0 to 4294967295. *(default = 0)*

portPriority

The root priority for this port. The valid range is 0 to 61440. *(default = 32,768)*

vlanPortPriority

The VLAN Port Priority. The valid range is 0 to 63. *(default = 32)*

COMMANDS

The **stpVlan** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

stpMsti generateTopologyChange

After the protocol is up and running, this subcommand generates Topology Change BPDUs, so that MAC addresses are flushed from the DUT and from relevant nodes in the spanning tree.

stpMsti setDefault

Sets default values for all configuration options.

EXAMPLES

```
package require IxTclHal

set hostname loopback

ixConnectToChassis $hostname

set chId [chassis cget -id]

set cardId 1

set portId 1

protocolServer config –enableStpService true

protocolServer set $chId $cardId $portId

stpServer select $chId $cardId $portId

stpLan config –enable true

stpLan config –startMacAddress “11 22 33 00 00 00

stpLan config –endMacAddress “55 00 00 00 11 11

stpServer addLan lan1

stpLan config –enable true

stpLan config –startMacAddress “88 22 33 00 00 00

stpLan config –endMacAddress “99 00 00 00 11 11

stpServer addLan lan2

stpInterface config –enable true

stpInterface config –cost 1

stpBridge addInterface interface1

stpBridge config –enable true

stdBridge config –bridgeSystemId 10

stdBridge config –lanNameList “lan1 lan2

stdServer addBridge bridge1

#Adding Vlan

stpVlan config -VlanId 15

stpVlan config -VlanRootPriority ‘10

stpServer addVlan Vlan1

stpVlan config -VlanId 20

stpVlan config -VlanRootPriority ‘11

stpServer addVlan Vlan2

stdServer write

#Get Lan names
```

```
stpServer select $chId $cardId $portId
stpServer showLanNames
#Get Vlans
stpServer showVlans
# Disable interface interface1.
stpServer select $chId $cardId $portId
stpServer getBridge bridge1
stpBridge getInterface interface1
stpInterface config –enable false
stpBridge setInterface interface1
#get the name for the second lan in the list
stpServer select $chId $cardId $portId
stpServer getFirstLan
stpServer getNextLan
set name [stpLan cget –name]
#get the second Vlan in the list
stpServer getFirstVlan
stpServer getNextVlan
# Request learned Info and get learned info
stpServer select $chId $cardId $portId
stpServer getBridge bridge1
stpBridge requestLearnedInfo
set timer 10
set rxFlag 0
while {$timer} {
if {[stpBridge getLearnedInfo]} {
set rxFlag 1
break;
}
After 1000
Incr timer -1
}
while {$timer} {
if {[stpBridge getFirstVlanLearnedInfo Vlan1]} {
```

```
set rxFlag 1
break;
}
After 1000
Incr timer -1
}
while {$timer} {
if {[stpBridge getNextVlanLearnedInfo Vlan1]} {
set rxFlag 1
break;
}
After 1000
Incr timer -1
}
if { $rxFlag} {
showCmd stpBridgeLearnedInfo
stpBridgeLearnedInfo getFirstinterfaceLearnedInfo
showCmd stpInterfaceLearnedInfo
}
if { $rxFlag} {
showCmd pvstVlanInterfaceLearnedInfo
stpBridgeVlanLearnedInfo getFirstinterfaceLearnedInfo
showCmd stpInterfaceLearnedInfo
}
```

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpInterfaceLearnedInfo*, *stpLan*, *stpServer*, *stpBridgeVlanLearnedInfo*, *stpVlanInterfaceLearnedInfo.*

NAME - stpVlanInterfaceLearnedInfo

stpVlanInterfaceLearnedInfo — views the retrieved learned information for an interface, associated with a VLAN for a PVST+ or RPVST+ Bridge.

SYNOPSIS

stpVlanInterfaceLearnedInfo *subcommand options*

DESCRIPTION

The *stpVlanInterfaceLearnedInfo* makes available learned state information (from the advertising bridge) for the VLAN interface on the PVST+ or RPVST+ Bridge in the current *stpBridge* command. Refer to *STP* for an overview of this command.

STANDARD OPTIONS

interfaceRole

Read-only. The advertised Role of the Interface. One of:

Option	Value	Usage
stpInterfaceRoleDisabled	0	The interface is disabled.
stpInterfaceRoleRoot	1	The interface has the shortest path to the root bridge from the current bridge.
stpInterfaceRoleDesignated	2	The interface is on the LAN segment.
stpInterfaceRoleAlternate	3	The interface is not currently part of the active topology, but it could function as an alternate for the designated interface on this bridge.
stpInterfaceRoleBackup	4	The interface is not currently part of the active topology, but it could function as a backup for the root interface on this bridge.

interfaceState

Read-only. The advertised State of the Interface. One of:

Option	Value	Usage
stpInterfaceStateDiscarding	0	The interface is discarding MAC frames.
stpInterfaceStateLearning	1	The interface is learning MAC addresses.
stpInterfaceStateForwarding	2	The interface is forwarding MAC frames.
stpInterfaceStateListening		(Available for use with PVST+/RPVST+ only)

designatedPriority

(Read-only.) The priority of the advertising Designated PVST+/RPVST+ bridge.

designatedMacAddress

(Read-only.) The 6-byte MAC address of the advertising Designated PVST+/RPVST+ bridge.

designatedPortId

(Read-only.) The Port ID of the advertising Designated PVST+/RPVST+ bridge's port on the LAN segment.

protocolInterfaceDescription

(Read-only.) The descriptive identifier for the advertised protocol interface.

rootCost

(Read-only.) The cost for the shortest path from the advertising bridge to the Regional Root bridge.

rootMac

(Read-only.) The Root bridge MAC address being advertised by the bridge.

rootPriority

(Read-only.) The priority being advertised for the Root bridge.

COMMANDS

The **stpVlanInterfaceLearnedInfo** command is invoked with the following sub-commands. If no subcommand is specified, returns a list of all subcommands available.

stpVlanInterfaceLearnedInfo setDefault

Sets default values for all options.

EXAMPLES

See examples under stpVlan.

SEE ALSO

stpBridge, *stpBridgeLearnedInfo*, *stpInterface*, *stpInterfaceLearnedInfo*, *stpServer*, *stpVlan*, *stpBridgeVlanLearnedInfo*.

NAME - mplsTpServer

mplsTpServer — accesses the mpls-tp component of the protocol server for a particular port.

SYNOPSIS

mplsTpServer *subcommand options*

DESCRIPTION

The *mplsTpServer* command is necessary in order to access the MPLS-TP protocol server for a particular port.

STANDARD OPTIONS

bfdCcChannelType

The bfd cc channel type.

apsChannelType

The asp channel type.

onDemandCvChannelType

The on demand cv channel type.

faultManagementChannelType

The fault management channel type.

lossMeasurementChannelType

The loss measurement channel type.

y1731ChannelType

The y1731 channel type.

pwStatusChannelType

The PW status channel type.

delayManagementChannelType

The delay measurement channel type.

COMMANDS

The **mplsTpServer** command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

mplsTpServer config *option value*

Modifies the configuration options of the mplsTpServer. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mplsTpServer.

mplsTpServer getFirstRouter

Gets the first router from the router list and refreshes the options.

Refer to *mplsTpServer Subcommands* for the list of subcommands.

EXAMPLE

```
##############################################################
# This Script has been generated by Ixia ScriptGen
# Software Version : IxOS 6.00.0.333 EB
##############################################################
package req IxTclHal
ixInitialize $hostname
chassis get $hostname
set chassis [chassis cget -id]
port setFactoryDefaults $chassis $card $port
if {[port setPhyMode $::portPhyModeCopper $chassis $card $port]} {
errorMsg "Error calling port setPhyMode
$::portPhyModeCopper $chassis $card $port"
set retCode $::TCL_ERROR
}
port config -speed 1000
port config -duplex full
port config -flowControl true
port config -directedAddress "01 80 C2 00 00 01"
port config -multicastPauseAddress "01 80 C2 00 00 01"
port config -loopback portNormal
port config -transmitMode
portTxModeAdvancedScheduler
port config -receiveMode [expr
$::portCapture|$::portRxModeWidePacketGroup]
port config -autonegotiate true
port config -advertise100FullDuplex true
port config -advertise100HalfDuplex true
port config -advertise10FullDuplex true
port config -advertise10HalfDuplex true
port config -advertise1000FullDuplex true
port config -portMode portEthernetMode
port config -enableDataCenterMode false
port config -dataCenterMode
eightPriorityTrafficMapping
port config -flowControlType ieee8023x
port config -pfcEnableValueListBitMatrix ""
port config -pfcResponseDelayEnabled 0
port config -pfcResponseDelayQuanta 0
port config -rxTxMode gigNormal
port config -ignoreLink false
port config -advertiseAbilities
portAdvertiseSendAndReceive
port config -timeoutEnable true
port config -negotiateMasterSlave 1
port config -masterSlave portSlave
port config -pmaClock
pmaClockAutoNegotiate
port config -enableSimulateCableDisconnect false
port config -enableAutoDetectInstrumentation false
port config -autoDetectInstrumentationMode
portAutoInstrumentationModeEndOfFrame
port config -enableRepeatableLastRandomPattern false
port config -transmitClockDeviation 0
port config -transmitClockMode portClockInternal
port config -preEmphasis preEmphasis0
port config -transmitExtendedTimestamp 0
port config -operationModeList [list
$::portOperationModeStream]
port config -MacAddress "00 de bb 00 00 01"
port config -DestMacAddress "00 de bb 00 00 02"
port config -name ""
port config -numAddresses 1
port config -enableManualAutoNegotiate false
port config -enablePhyPolling true
port config -enableTxRxSyncStatsMode false
port config -txRxSyncInterval 0
port config -enableTransparentDynamicRateChange false
port config -enableDynamicMPLSMode false
port config -enablePortCpuFlowControl false
port config -portCpuFlowControlDestAddr "01 80 C2 00 00 01"
port config -portCpuFlowControlSrcAddr "00 00 01 00 02 00"
port config -portCpuFlowControlPriority "0 0 0 0 0 0 0 0"
port config -portCpuFlowControlType 0
port config -enableWanIFSStretch false
if {[port set $chassis $card $port]} {
errorMsg "Error calling port set $chassis $card $port"
set retCode $::TCL_ERROR
}
stat setDefault
stat config -mode statNormal
stat config -enableValidStats false
stat config -enableProtocolServerStats true
stat config -enableArpStats true
stat config -enablePosExtendedStats true
stat config -enableDhcpStats false
stat config -enableDhcpV6Stats false
stat config -enableEthernetOamStats false
stat config -enableBgpStats false
stat config -enableIcmpStats true
stat config -enableOspfStats false
stat config -enableIsisStats false
stat config -enableRsvpStats false
stat config -enableLdpStats false
stat config -enableIgmpStats false
stat config -enableOspfV3Stats false
stat config -enablePimsmStats false
stat config -enableMldStats false
stat config -enableStpStats false
stat config -enableEigrpStats false
stat config -enableBfdStats false
stat config -enableCfmStats false
stat config -enableLacpStats false
stat config -enableOamStats false
stat config -enableMplsTpStats true
if {[stat set $chassis $card $port]} {
errorMsg "Error calling stat set $chassis $card $port"
set retCode $::TCL_ERROR
}
packetGroup setDefault
packetGroup config -signatureOffset 48
packetGroup config -signature "08 71 18 05"
packetGroup config -insertSignature false
packetGroup config -ignoreSignature false
packetGroup config -groupId 0
packetGroup config -groupIdOffset 52
packetGroup config -enableGroupIdMask false
packetGroup config -enableInsertPgid true
packetGroup config -groupIdMask 0
packetGroup config -latencyControl cutThrough
packetGroup config -measurementMode
packetGroupModeLatency
packetGroup config -delayVariationMode
delayVariationWithSequenceErrors
packetGroup config -preambleSize 8
packetGroup config -sequenceNumberOffset 44
packetGroup config -sequenceErrorThreshold 2
packetGroup config -insertSequenceSignature false
packetGroup config -allocateUdf true
packetGroup config -enableSignatureMask false
packetGroup config -signatureMask "00 00 00 00"
packetGroup config -enableRxFilter false
packetGroup config -headerFilter "00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00"
packetGroup config -headerFilterMask "00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00"
packetGroup config -enable128kBinMode false
packetGroup config -enableTimeBins false
packetGroup config -numPgidPerTimeBin 32
packetGroup config -numTimeBins 1
packetGroup config -timeBinDuration 1000000
packetGroup config -enableLatencyBins false
packetGroup config -latencyBinList ""
packetGroup config -groupIdMode
packetGroupCustom
packetGroup config -sequenceCheckingMode seqThreshold
packetGroup config -multiSwitchedPathMode
seqSwitchedPathPGID
packetGroup config -enableLastBitTimeStamp false
if {[packetGroup setRx $chassis $card $port]} {
errorMsg "Error calling packetGroup setRx $chassis $card
$port"
set retCode $::TCL_ERROR
}
flexibleTimestamp setDefault
flexibleTimestamp config -type
timestampBeforeCrc
flexibleTimestamp config -offset 23
if {[flexibleTimestamp set $chassis $card $port]} {
errorMsg "Error calling flexibleTimestamp set $chassis
$card $port"
set retCode $::TCL_ERROR
}
capture setDefault
capture config -fullAction lock
capture config -sliceSize 8191
capture config -sliceOffset 0
capture config -captureMode
captureTriggerMode
capture config -continuousFilter 0
capture config -beforeTriggerFilter
captureBeforeTriggerNone
capture config -afterTriggerFilter
captureAfterTriggerFilter
capture config -triggerPosition 1.0
capture config -enableSmallPacketCapture false
if {[capture set $chassis $card $port]} {
errorMsg "Error calling capture set $chassis $card $port"
set retCode $::TCL_ERROR
}
filter setDefault
filter config -captureTriggerDA anyAddr
filter config -captureTriggerSA anyAddr
filter config -captureTriggerPattern anyPattern
filter config -captureTriggerError errAnyFrame
filter config -captureTriggerFrameSizeEnable false
filter config -captureTriggerFrameSizeFrom 12
filter config -captureTriggerFrameSizeTo 12
filter config -captureTriggerCircuit filterAnyCircuit
filter config -captureFilterDA anyAddr
filter config -captureFilterSA anyAddr
filter config -captureFilterPattern anyPattern
filter config -captureFilterError errAnyFrame
filter config -captureFilterFrameSizeEnable false
filter config -captureFilterFrameSizeFrom 12
filter config -captureFilterFrameSizeTo 12
filter config -captureFilterCircuit filterAnyCircuit
filter config -userDefinedStat1DA anyAddr
filter config -userDefinedStat1SA anyAddr
filter config -userDefinedStat1Pattern anyPattern
filter config -userDefinedStat1Error errAnyFrame
filter config -userDefinedStat1FrameSizeEnable false
filter config -userDefinedStat1FrameSizeFrom 12
filter config -userDefinedStat1FrameSizeTo 12
filter config -userDefinedStat1Circuit filterAnyCircuit
filter config -userDefinedStat2DA anyAddr
filter config -userDefinedStat2SA anyAddr
filter config -userDefinedStat2Pattern anyPattern
filter config -userDefinedStat2Error errAnyFrame
filter config -userDefinedStat2FrameSizeEnable 0
filter config -userDefinedStat2FrameSizeFrom 12
filter config -userDefinedStat2FrameSizeTo 12
filter config -userDefinedStat2Circuit filterAnyCircuit
filter config -asyncTrigger1DA anyAddr
filter config -asyncTrigger1SA anyAddr
filter config -asyncTrigger1Pattern anyPattern
filter config -asyncTrigger1Error errAnyFrame
filter config -asyncTrigger1FrameSizeEnable false
filter config -asyncTrigger1FrameSizeFrom 12
filter config -asyncTrigger1FrameSizeTo 12
filter config -asyncTrigger1Circuit filterAnyCircuit
filter config -asyncTrigger2DA anyAddr
filter config -asyncTrigger2SA anyAddr
filter config -asyncTrigger2Pattern anyPattern
filter config -asyncTrigger2Error errAnyFrame
filter config -asyncTrigger2FrameSizeEnable false
filter config -asyncTrigger2FrameSizeFrom 12
filter config -asyncTrigger2FrameSizeTo 12
filter config -asyncTrigger2Circuit filterAnyCircuit
filter config -captureTriggerEnable true
filter config -captureFilterEnable true
filter config -userDefinedStat1Enable false
filter config -userDefinedStat2Enable false
filter config -asyncTrigger1Enable false
filter config -asyncTrigger2Enable false
if {[filter set $chassis $card $port]} {
errorMsg "Error calling filter set $chassis $card $port"
set retCode $::TCL_ERROR
}
filterPallette setDefault
filterPallette config -DA1 "00 00 00
00 00 00"
filterPallette config -DAMask1 "00 00 00
00 00 00"
filterPallette config -DA2 "00 00 00
00 00 00"
filterPallette config -DAMask2 "00 00 00
00 00 00"
filterPallette config -SA1 "00 00 00
00 00 00"
filterPallette config -SAMask1 "00 00 00
00 00 00"
filterPallette config -SA2 "00 00 00
00 00 00"
filterPallette config -SAMask2 "00 00 00
00 00 00"
filterPallette config -pattern1 "DE ED EF
FE AC CA"
filterPallette config -patternMask1 "00 00 00
00 00 00"
filterPallette config -pattern2 00
filterPallette config -patternMask2 00
filterPallette config -patternOffset1 12
filterPallette config -patternOffset2 12
filterPallette config -matchType1 matchUser
filterPallette config -matchType2 matchUser
filterPallette config -patternOffsetType1
filterPalletteOffsetStartOfFrame
filterPallette config -patternOffsetType2
filterPalletteOffsetStartOfFrame
filterPallette config -gfpErrorCondition
gfpErrorsOr
filterPallette config -enableGfptHecError true
filterPallette config -enableGfpeHecError true
filterPallette config -enableGfpPayloadCrcError true
filterPallette config -enableGfpBadFcsError true
filterPallette config -circuitList ""
if {[filterPallette set $chassis $card $port]} {
errorMsg "Error calling filterPallette set $chassis $card
$port"
set retCode $::TCL_ERROR
}
ipAddressTable setDefault
ipAddressTable config -defaultGateway "0.0.0.0"
if {[ipAddressTable set $chassis $card $port]} {
errorMsg "Error calling ipAddressTable set $chassis $card
$port"
set retCode $::TCL_ERROR
}
arpServer setDefault
arpServer config -retries 3
arpServer config -mode arpGatewayOnly
arpServer config -rate 2083333
arpServer config -requestRepeatCount 3
if {[arpServer set $chassis $card $port]} {
errorMsg "Error calling arpServer set $chassis $card
$port"
set retCode $::TCL_ERROR
}
if {[interfaceTable select $chassis $card $port]} {
```

```
errorMsg "Error calling interfaceTable select $chassis
$card $port"
set retCode $::TCL_ERROR
}
interfaceTable setDefault
interfaceTable config -dhcpV4RequestRate 0
interfaceTable config -dhcpV6RequestRate 0
interfaceTable config -dhcpV4MaximumOutstandingRequests 100
interfaceTable config -dhcpV6MaximumOutstandingRequests 100
interfaceTable config -fcoeRequestRate 500
interfaceTable config -fcoeNumRetries 5
interfaceTable config -fcoeRetryInterval 2000
interfaceTable config -fipVersion
fipVersion1
interfaceTable config -enableFcfMac false
interfaceTable config -fcfMacCollectionTime 1000
interfaceTable config -enablePMacInFpma true
interfaceTable config -enableNameIdInVLANDiscovery false
interfaceTable config -enableTargetLinkLayerAddrOption false
if {[interfaceTable set]} {
errorMsg "Error calling interfaceTable set"
set retCode $::TCL_ERROR
}
interfaceTable clearAllInterfaces
if {[interfaceTable write]} {
errorMsg "Error calling interfaceTable write"
set retCode $::TCL_ERROR
}
interfaceEntry clearAllItems addressTypeIpV6
interfaceEntry clearAllItems addressTypeIpV4
interfaceEntry setDefault
interfaceIpV4 setDefault
interfaceIpV4 config -gatewayIpAddress {1.1.1.2}
interfaceIpV4 config -maskWidth 24
```

```
interfaceIpV4 config -ipAddress {1.1.1.1}
if {[interfaceEntry addItem addressTypeIpV4]} {
errorMsg "Error calling interfaceEntry addItem
addressTypeIpV4"
set retCode $::TCL_ERROR
}
dhcpV4Properties removeAllTlvs
dhcpV4Properties setDefault
dhcpV4Properties config -clientId ""
dhcpV4Properties config -serverId
"0.0.0.0"
dhcpV4Properties config -vendorId ""
dhcpV4Properties config -renewTimer 0
dhcpV4Properties config -retryCount 4
dhcpV4Properties config -relayAgentAddress
"0.0.0.0"
dhcpV4Properties config -relayDestinationAddress
"255.255.255.255"
dhcpV6Properties removeAllTlvs
dhcpV6Properties setDefault
dhcpV6Properties config -iaType
dhcpV6IaTypePermanent
dhcpV6Properties config -iaId 0
dhcpV6Properties config -renewTimer 0
dhcpV6Properties config -relayLinkAddress
"0:0:0:0:0:0:0:0"
dhcpV6Properties config -relayDestinationAddress
"FF05:0:0:0:0:0:1:3"
interfaceEntry config -enable true
interfaceEntry config -description {Connected
- ProtocolInterface - 100:01 - 7}
interfaceEntry config -macAddress {00 00 81
54 6C 6A}
interfaceEntry config -mtu 1500
interfaceEntry config -eui64Id {02 00 81
FF FE 54 6C 6A}
interfaceEntry config -atmEncapsulation
atmEncapsulationLLCBridgedEthernetFCS
interfaceEntry config -atmVpi 0
interfaceEntry config -atmVci 32
interfaceEntry config -enableDhcp false
interfaceEntry config -enableDhcpV6 false
interfaceEntry config -enableVlan false
interfaceEntry config -vlanId 1
interfaceEntry config -vlanPriority 0
if {[interfaceTable addInterface 0]} {
errorMsg "Error calling interfaceTable addInterface 0"
set retCode $::TCL_ERROR
}
if {[mplsTpServer select $chassis $card $port]} {
errorMsg "Error calling mplsTpServer select $chassis
$card $port"
set retCode $::TCL_ERROR
}
mplsTpServer clearAllRouters
mplsTpLspPwRange setDefault
mplsTpLspPwRange config -enabled true
mplsTpLspPwRange config -skipZeroVlanId 1
mplsTpLspPwRange config -enableVlan true
mplsTpLspPwRange config -repeatMac 0
mplsTpLspPwRange config -revertive 0
mplsTpLspPwRange config -supportSlowStart 0
mplsTpLspPwRange config -typeOfRange lsp
mplsTpLspPwRange config -rangeRole
rangeRoleWorking
mplsTpLspPwRange config -lspOutgoingLabel 100
mplsTpLspPwRange config -lspIncomingLabel 500
mplsTpLspPwRange config -pwOutgoingLabel 16
mplsTpLspPwRange config -pwIncomingLabel 16
mplsTpLspPwRange config -srcMepId 1
mplsTpLspPwRange config -destMepId 2
mplsTpLspPwRange config -cccvType
cccvBfdCc
mplsTpLspPwRange config -apsType apsIetf
mplsTpLspPwRange config -lspOutgoingLabelStep 1
mplsTpLspPwRange config -lspIncomingLabelStep 1
mplsTpLspPwRange config -pwOutgoingLabelStep 1
mplsTpLspPwRange config -pwIncomingLabelStep 1
mplsTpLspPwRange config -srcMepIdStep 1
mplsTpLspPwRange config -destMepIdStep 1
mplsTpLspPwRange config -vlanIncrementMode
noIncrement
mplsTpLspPwRange config -vlanCount 1
mplsTpLspPwRange config -ipType
addressTypeIpV4
mplsTpLspPwRange config -ipAddressMask 32
mplsTpLspPwRange config -ipAddressStep 1
mplsTpLspPwRange config -alarmType
alarmTypeIetf
mplsTpLspPwRange config -dmType
dmTypeIetf
mplsTpLspPwRange config -lmType
lmTypeIetf
mplsTpLspPwRange config -cccvTrafficClass 7
mplsTpLspPwRange config -typeOfProtectionSwitching
onePlusOneBidirectional
mplsTpLspPwRange config -waitToRevertTime 300
mplsTpLspPwRange config -apsTrafficClass 7
mplsTpLspPwRange config -pwOutgoingLabelStepAcrossLsp 0
mplsTpLspPwRange config -pwIncomingLabelStepAcrossLsp 0
mplsTpLspPwRange config -numberOfLsp 1
mplsTpLspPwRange config -numberOfPwPerLsp 0
mplsTpLspPwRange config -megIdIntegerStep 0
mplsTpLspPwRange config -macPerPw 1
mplsTpLspPwRange config -ipHostPerLsp 10
mplsTpLspPwRange config -cccvInterval 1000.0
mplsTpLspPwRange config -peerLspOrPwRange
"IXIA.0001.0001.0001.0002"
mplsTpLspPwRange config -megIdPrefix "Ixia-
0001"
mplsTpLspPwRange config -vlanTpId 0x8100
mplsTpLspPwRange config -vlanPriority 0
mplsTpLspPwRange config -vlanId 1
mplsTpLspPwRange config -macAddress
"00:00:00:00:00:00"
mplsTpLspPwRange config -description
"IXIA.0001.0001.0001.0001"
mplsTpLspPwRange config -ipAddress
"11.1.1.1"
mplsTpLspPwRange config -srcGlobalId 1
mplsTpLspPwRange config -srcNodeId 1
mplsTpLspPwRange config -srcTunnelNumber 1
mplsTpLspPwRange config -srcTunnelNumberStep 1
mplsTpLspPwRange config -srcLspNumber 1
mplsTpLspPwRange config -srcLspNumberStep 1
mplsTpLspPwRange config -srcAcId 1
mplsTpLspPwRange config -srcAcIdStep 1
mplsTpLspPwRange config -destGlobalId 1
mplsTpLspPwRange config -destNodeId 2
mplsTpLspPwRange config -destTunnelNumberStep 1
mplsTpLspPwRange config -destTunnelNumber 1
mplsTpLspPwRange config -destLspNumber 1
mplsTpLspPwRange config -destLspNumberStep 1
mplsTpLspPwRange config -destAcId 2
mplsTpLspPwRange config -destAcIdStep 1
mplsTpLspPwRange config -onDemandCvTrafficClass 7
mplsTpLspPwRange config -pwStatusTrafficClass 7
mplsTpLspPwRange config -alarmTrafficClass 7
mplsTpLspPwRange config -lmTrafficClass 7
mplsTpLspPwRange config -dmTrafficClass 7
mplsTpLspPwRange config -dmTimeFormat
dmTimeFormatIeee
mplsTpLspPwRange config -lmCounterType
lmCounterType32Bit
mplsTpLspPwRange config -lmInitialTxValue 1
mplsTpLspPwRange config -lmTxStep 1
mplsTpLspPwRange config -lmInitialRxValue 1
mplsTpLspPwRange config -lmRxStep 1
mplsTpInterface addLspPwRange LSPPWRange1
mplsTpLspPwRange setDefault
mplsTpLspPwRange config -enabled true
mplsTpLspPwRange config -skipZeroVlanId 1
mplsTpLspPwRange config -enableVlan true
mplsTpLspPwRange config -repeatMac 0
mplsTpLspPwRange config -revertive 0
mplsTpLspPwRange config -supportSlowStart 0
mplsTpLspPwRange config -typeOfRange lsp
mplsTpLspPwRange config -rangeRole
rangeRoleProtect
mplsTpLspPwRange config -lspOutgoingLabel 1000
mplsTpLspPwRange config -lspIncomingLabel 1500
mplsTpLspPwRange config -pwOutgoingLabel 16
mplsTpLspPwRange config -pwIncomingLabel 16
mplsTpLspPwRange config -srcMepId 1
mplsTpLspPwRange config -destMepId 2
mplsTpLspPwRange config -cccvType
cccvBfdCc
mplsTpLspPwRange config -apsType apsIetf
mplsTpLspPwRange config -lspOutgoingLabelStep 1
mplsTpLspPwRange config -lspIncomingLabelStep 1
mplsTpLspPwRange config -pwOutgoingLabelStep 1
mplsTpLspPwRange config -pwIncomingLabelStep 1
mplsTpLspPwRange config -srcMepIdStep 1
mplsTpLspPwRange config -destMepIdStep 1
mplsTpLspPwRange config -vlanIncrementMode
noIncrement
mplsTpLspPwRange config -vlanCount 1
mplsTpLspPwRange config -ipType
addressTypeIpV4
mplsTpLspPwRange config -ipAddressMask 24
mplsTpLspPwRange config -ipAddressStep 1
mplsTpLspPwRange config -alarmType
alarmTypeIetf
mplsTpLspPwRange config -dmType
dmTypeIetf
mplsTpLspPwRange config -lmType
lmTypeIetf
mplsTpLspPwRange config -cccvTrafficClass 7
mplsTpLspPwRange config -typeOfProtectionSwitching
oneIstoOneBidirectional
mplsTpLspPwRange config -waitToRevertTime 300
mplsTpLspPwRange config -apsTrafficClass 7
mplsTpLspPwRange config -pwOutgoingLabelStepAcrossLsp 0
mplsTpLspPwRange config -pwIncomingLabelStepAcrossLsp 0
mplsTpLspPwRange config -numberOfLsp 5
mplsTpLspPwRange config -numberOfPwPerLsp 0
mplsTpLspPwRange config -megIdIntegerStep 0
mplsTpLspPwRange config -macPerPw 1
mplsTpLspPwRange config -ipHostPerLsp 0
mplsTpLspPwRange config -cccvInterval 1000.0
mplsTpLspPwRange config -peerLspOrPwRange ""
mplsTpLspPwRange config -megIdPrefix "Ixia-
0001"
mplsTpLspPwRange config -vlanTpId 0x8100
mplsTpLspPwRange config -vlanPriority 0
mplsTpLspPwRange config -vlanId 1
mplsTpLspPwRange config -macAddress
"00:00:00:00:00:00"
mplsTpLspPwRange config -description
"IXIA.0001.0001.0001.0002"
mplsTpLspPwRange config -ipAddress
"0.0.0.0"
mplsTpLspPwRange config -srcGlobalId 1
mplsTpLspPwRange config -srcNodeId 1
mplsTpLspPwRange config -srcTunnelNumber 1
mplsTpLspPwRange config -srcTunnelNumberStep 1
mplsTpLspPwRange config -srcLspNumber 1
mplsTpLspPwRange config -srcLspNumberStep 1
mplsTpLspPwRange config -srcAcId 1
mplsTpLspPwRange config -srcAcIdStep 1
mplsTpLspPwRange config -destGlobalId 1
mplsTpLspPwRange config -destNodeId 2
mplsTpLspPwRange config -destTunnelNumberStep 1
mplsTpLspPwRange config -destTunnelNumber 1
mplsTpLspPwRange config -destLspNumber 1
mplsTpLspPwRange config -destLspNumberStep 1
mplsTpLspPwRange config -destAcId 2
mplsTpLspPwRange config -destAcIdStep 1
mplsTpLspPwRange config -onDemandCvTrafficClass 7
mplsTpLspPwRange config -pwStatusTrafficClass 7
mplsTpLspPwRange config -alarmTrafficClass 7
mplsTpLspPwRange config -lmTrafficClass 7
mplsTpLspPwRange config -dmTrafficClass 7
mplsTpLspPwRange config -dmTimeFormat
dmTimeFormatIeee
mplsTpLspPwRange config -lmCounterType
lmCounterType32Bit
mplsTpLspPwRange config -lmInitialTxValue 1
mplsTpLspPwRange config -lmTxStep 1
mplsTpLspPwRange config -lmInitialRxValue 1
mplsTpLspPwRange config -lmRxStep 1
mplsTpInterface addLspPwRange LSPPWRange2
mplsTpInterface setDefault
mplsTpInterface config -enabled true
mplsTpInterface config -dutMacAddress
"ff:ff:ff:ff:ff:ff"
mplsTpInterface config -interfaces
"Connected - ProtocolInterface - 100:01 - 7"
if {[mplsTpRouter addInterface Interface1]} {
errorMsg "Error calling mplsTpRouter addInterface
Interface1"
set retCode $::TCL_ERROR
}
mplsTpRouter setDefault
mplsTpRouter config -enabled true
mplsTpRouter config -routerId "100.1.0.1"
mplsTpRouter config -enableCccvPause false
mplsTpRouter config -cccvPauseTriggerOption
cccvPauseTriggerOptionTx
mplsTpRouter config -enableCccvResume false
mplsTpRouter config -cccvResumeTriggerOption
cccvResumeTriggerOptionTx
mplsTpRouter config -apsTriggerType
apsTriggerTypeForcedSwitch
mplsTpRouter config -lmInterval 1000
mplsTpRouter config -lspTraceRouteTtlLimit 5
mplsTpRouter config -enableLspTraceRoute false
mplsTpRouter config -lmTrafficClass 7
mplsTpRouter config -enableLspPingFecStackValidation true
mplsTpRouter config -enablePwStatusFault false
mplsTpRouter config -alarmType
alarmTriggerTypeIetf
```

```
mplsTpRouter config -dmType
dmTriggerTypeIetf
mplsTpRouter config -dmIterations 10
mplsTpRouter config -lastDmResponseTimeout 1000
mplsTpRouter config -lmTxStep 1000
mplsTpRouter config -enableAlarm false
mplsTpRouter config -counterType
counterType32Bit
mplsTpRouter config -dmPadLen 0
mplsTpRouter config -enableAlarmAis true
mplsTpRouter config -periodicity 5
mplsTpRouter config -dmTrafficClass 7
mplsTpRouter config -enableDmTrigger false
mplsTpRouter config -lspTraceRouteResponseTimeout 1000
mplsTpRouter config -enableAlarmLck false
mplsTpRouter config -alarmTrigger
alarmTriggerStart
mplsTpRouter config -lmIterations 10
mplsTpRouter config -enableAlarmSetLdi true
mplsTpRouter config -dmInterval 1000
mplsTpRouter config -lmInitialRxValue 1000
mplsTpRouter config -lmInitialTxValue 1000
mplsTpRouter config -enableLspPing false
mplsTpRouter config -lspPingTtlValue 255
mplsTpRouter config -lspPingResponseTimeout 1000
mplsTpRouter config -enableApsTrigger false
mplsTpRouter config -dmRequestPaddedReply 0
mplsTpRouter config -lastLmResponseTimeout 1000
mplsTpRouter config -lmRxStep 1000
mplsTpRouter config -dmMode
dmModeResponseExpected
mplsTpRouter config -dmTimeFormat
dmTriggerTimeFormatNtp
mplsTpRouter config -enableLmTrigger false
mplsTpRouter config -lmType
lmTriggerTypeIetf
mplsTpRouter config -lmMode
lmModeResponseExpected
mplsTpRouter config -enableLspTraceRouteFecStackValidation true
mplsTpServer addRouter Router1
mplsTpServer setDefault
mplsTpServer config -onDemandCvChannelType 09
mplsTpServer config -bfdCcChannelType 07
mplsTpServer config -apsChannelType 02
mplsTpServer config -faultManagementChannelType 03
mplsTpServer config -lossMeasurementChannelType 04
mplsTpServer config -delayManagementChannelType 05
mplsTpServer config -y1731ChannelType "7FFA"
mplsTpServer config -pwStatusChannelType 01
if {[mplsTpServer set]} {
errorMsg "Error calling mplsTpServer set"
set retCode $::TCL_ERROR
}
if {[mplsTpServer write]} {
errorMsg "Error calling mplsTpServer write"
set retCode $::TCL_ERROR
}
protocolServer setDefault
protocolServer config -enableArpResponse true
protocolServer config -enablePingResponse false
protocolServer config -enableIgmpQueryResponse false
protocolServer config -enableOspfService false
protocolServer config -enableBgp4Service false
protocolServer config -enableIsisService false
protocolServer config -enableRsvpService false
protocolServer config -enableRipService false
protocolServer config -enableLdpService false
protocolServer config -enableRipngService false
protocolServer config -enableMldService false
protocolServer config -enableOspfV3Service false
protocolServer config -enablePimsmService false
protocolServer config -enableStpService false
protocolServer config -enableEigrpService false
protocolServer config -enableBfdService false
protocolServer config -enableCfmService false
protocolServer config -enableLacpService false
protocolServer config -enableOamService false
protocolServer config -enableMplsTpService true
protocolServer config -enableBgp4CreateInterface false
protocolServer config -enableIsisCreateInterface false
protocolServer config -enableOspfCreateInterface false
protocolServer config -enableRipCreateInterface false
protocolServer config -enableRsvpCreateInterface false
protocolServer config -enableIgmpCreateInterface false
if {[protocolServer set $chassis $card $port]} {
errorMsg "Error calling protocolServer set $chassis $card
$port"
set retCode $::TCL_ERROR
}
oamPort setDefault
oamPort config -enable false
oamPort config -macAddress "00 00 00 00 00
00"
oamPort config -enableLoopback false
oamPort config -enableLinkEvents false
oamPort config -maxOamPduSize 1518
oamPort config -oui "00 00 00"
oamPort config -vendorSpecificInformation "00 00 00 00"
oamPort config -idleTimer 5
oamPort config -enableOptionalTlv false
oamPort config -optionalTlvType 254
oamPort config -optionalTlvValue ""
```

```
if {[oamPort set $chassis $card $port]} {
errorMsg "Error calling oamPort set $chassis $card $port"
set retCode $::TCL_ERROR
}
lappend portList [list $chassis $card $port]
ixWritePortsToHardware portList
ixCheckLinkState portList
#################################################################
Generating streams for all the ports from above
#################################################################
######### Chassis-xm12-4 Card-3 Port-11 #########
chassis get "xm12-4"
set chassis [chassis cget -id]
set card 3
set port 11
streamRegion get $chassis $card $port
if {[streamRegion enableGenerateWarningList $chassis $card $port
0]} {
errorMsg "Error calling streamRegion
enableGenerateWarningList $chassis $card $port 0"
set retCode $::TCL_ERROR
}
if {[port reset $chassis $card $port]} {
errorMsg "Error calling port reset $chassis $card $port"
set retCode $::TCL_ERROR
}
streamRegion generateWarningList $chassis $card $port
ixWriteConfigToHardware portList -noProtocolServer
```

NAME - mplsTpRouter

mplsTpRouter — adds a simulated mpls-tp router.

SYNOPSIS

mplsTprouter *subcommand options*

DESCRIPTION

The *mplsTprouter* command adds a simulated mpls-tp router.

STANDARD OPTIONS

enabled

Enables this simulated mpls-tp router. This can be enabled/disabled based on its value set as true/false.

routerId

The ID of the simulated router, which is expressed as an IP address.

enableCccvPause

Enable cccv pause. This can be enabled/disabled based on its value set as true/false.

Default = false

cccvPauseTriggerOption

The cccv pause trigger option.

enableCccvResume

Enables cccv resume. This can be enabled/disabled based on its value set as true/false.

Default = false

cccvResumeTriggerOption

The cccv resume trigger option.

apsTriggerType

The aps trigger type. Possible values include:

- clear
- forcedSwitch
- manualSwitch
- lockout
- exercise
- freeze

lmInterval

The lm interval.

lspTraceRouteTtlLimit

The lsp trace route ttl limit.

enableLspTraceRoute

Enables lsp trace route.

lmTrafficClass

The lm traffic class value.

enableLspPingFecStackValidation

Enables lsp ping fec stack validation.

enablePwStatusFault

Enables pw fault status.

alarmType

The type of alarm. Possible values include:

- ietf
- y1731

dmType

The DM type. Possible values include:

- ietf
- y1731

dmIterations

The total dm iterations.

lastDmResponseTimeout

The last dm response timeout.

lmTxStep

The increment value for lm transmit.

enableAlarm

Enables alarm.

counterType

The counter type. Possible values include:

- 32Bit
- 64Bit

dmPadLen

The dm pad length.

enableAlarmAis

Enables alarm ais.

periodicity

Indicates the periodicity.

dmTrafficClass

The dm traffic class.

enableDmTrigger

Enables dm trigger.

lspTraceRouteResponseTimeout

The lsp trace route response timeout.

enableAlarmLck

Enables alarm lck.

alarmTrigger

The alarm trigger. Possible values include:

- clear
- start

lmIterations

The lm iterations.

enableAlarmSetLdi

Enables alarm set ldi.

dmInterval

The dm interval value.

lmInitialRxValue

The initial lm receive value.

lmInitialTxValue

The initial lm transmit value.

enableLspPing

Enables lsp ping.

lspPingTtlValue

The lsp ping ttl value.

lspPingResponseTimeout

The lsp ping response timeout.

enableApsTrigger

Enables APS trigger.

dmRequestPaddedReply

The dm request padded reply.

lastLmResponseTimeout

The last lm response timeout.

lmRxStep

The step value for lm receive.

dmMode

The dm mode. Possible values include:

- noResponseExpected
- responseExpected

dmTimeFormat

The dm time format. Possible values include:

- ieee
- ntp

enableLmTrigger

Enables lm trigger.

lmType

The lm type. Possible values include:

- ietf
- y1731

lmMode

The lm mode. Possible values include:

- responseExpected

noResponseExpected

enableLspTraceRouteFecStackValidation

Enables lsp trace route fec stack validation.

COMMANDS

The **mplsTpRouter** command is invoked with the following subcommands. If no sub-command is specified, returns a list of all subcommands available.

mplsTpRouter config *option value*

Modifies the configuration options of the mplsTpRouter. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mplsTpRouter.

mplsTpRouter addRouter

Adds a simulated mpls-tp router.

Refer to *mplsTpRouter Subcommands* for the list of subcommands.

EXAMPLE

See *EXAMPLE* section of *mplsTpServer*

NAME - mplsTpLspPwRange

mplsTpLspPwRange — adds an lsp pw range.

SYNOPSIS

mplsTpLspPwRange *subcommand options*

DESCRIPTION

The *mplsTpLspPwRange* command adds an lsp pw range.

STANDARD OPTIONS

enabled

Enables this LSP PW Range.

typeOfRange

The type of range. Possible values include:

- lsp
- pw

rangeRole

The role of the range. Possible values include:

- none
- working
- protect

peerLspOrPwRange

The type of range.

numberOfLsp

The total number of lsps.

numberOfPwPerLsp

The total number of PWs per LSP.

lspOutgoingLabel

The outgoing LSP label.

lspIncomingLabel

The incoming LSP label.

pwOutgoingLabel

The outgoing PW label.

pwIncomingLabel

The incoming PW label.

cccvInterval

The cccv interval value.

cccvType

The cccv type. Possible values include:

- bfdCc
- y1731

megIdPrefix

The prefix for the meg id.

srcMepId

The source MEP id.

destMepId

The destination MEP id.

ipType

The id type.Possible values include:

- ipv4
- ipv6

ipAddress

The IP address value.

ipAddressMask

The IP address mask.

ipAddressStep

The increment value for the IP address.

trafficGroupId

The traffic group id.

skipZeroVlanId

If true, skips vlands with zero values.

vlanIncrementMode

The increment vaue for the vlan. Possible values include:

- noIncrement
- parallelIncrement
- innerFirst
- outerFirst

vlanTpId

The vlan TP id.

vlanPriority

The vlan priority value.

vlanId

The vlan id.

vlanCount

The vlan count.

enableVlan

If true, enables the vlan.

macAddress

The MAC address.

repeatMac

If true, repeats the MAC addresses.

macPerPw

The total MAC per PW.

lspOutgoingLabelStep

The increment value for the outgoing lsp label.

lspIncomingLabelStep

The increment value for the incoming lsp label.

pwOutgoingLabelStep

The increment value for the outgoing pw label.

pwIncomingLabelStep

The increment value for the incoming pw label.

megIdIntegerStep

The increment value for the MEG.

srcMepIdStep

The increment value for the source MEP.

destMepIdStep

The increment value for the destination MEP.

description

The description of the range.

pwOutgoingLabelStepAcrossLsp

The increment value for the outgoing pw label across lsp.

pwIncomingLabelStepAcrossLsp

The increment value for the incoming pw label across lsp.

alarmType

The type of alarm. Possible values include:

- ietf
- y1731

dmType

The DM type. Possible values include:

- ietf
- y1731

lmType

The LM type. Possible values include:

- ietf
- y1731

ipHostPerLsp

The IP host per lsp.

cccvTrafficClass

The cccv traffic class value.

typeOfProtectionSwitching

The type of switching protection. Possible values include:

- 1:1Unidirectional
- 1+1Unidirectional

- 1:1Bidirectional
- 1+1Bidirectional

revertive

The revertive value.

waitToRevertTime

The wait time to revert.

onDemandCvTrafficClass

The on demand cv traffic class value.

pwStatusTrafficClass

The pw status traffic class value.

alarmTrafficClass

The alarm traffic class value.

dmTimeFormat

The dm time format. Possible values include:

- ieee
- ntp

dmTrafficClass

The dm traffic class value.

lmCounterType

The lm counter type. Possible values include:

- 32Bit
- 64Bit

lmInitialTxValue

The initial lm transmit value.

lmTxStep

The increment value for lm transmit.

lmInitialRxValue

The initial lm receive value.

lmRxStep

The increment value for lm receive.

lmTrafficClass

The lm traffic class value.

destGlobalId

The destination global id.

srcLspNumber

.The source LSP number.

srcTunnelNumberStep

The increment value for the source tunnel number.

srcTunnelNumber

The source tunnel number value.

destNodeId

The destination node id.

destAcIdStep

The increment value for the destination ac id.

srcNodeId

The source node id.

destTunnelNumberStep

The increment value for the destination tunnel number.

destAcId

The destination ac id.

supportSlowStart

If true, support slow start.

pwStatusFaultReplyInterval

The interval value for the pw fault status.

destTunnelNumber

The destination tunnel number.

destLspNumber

The destination LSP number.

destLspNumberStep

The increment value for the destination LSP number.

apsType

The aps types. Possible values include:

- ietf
- y1731

apsTrafficClass

The aps traffic class value.

srcAcIdStep

The increment value for the source ac id.

srcGlobalId

The source global id.

srcAcId

The source ac id.

srcLspNumberStep

The increment value for the source LSP number.

COMMANDS

The **mplsTpLspPwRange** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsTpLspPwRange config *option value*

Modifies the configuration options of the *mplsTpLspPwRange*. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mplsTpLspPwRange.

Refer to mplsTpLspPwRange Subcommands for the list of subcommands.

EXAMPLE

See *EXAMPLE* section of *mplsTpServer*

NAME - mplsTpInterface

mplsTpInterface — holds the information related to a single interface on the simulated router.

SYNOPSIS

mplsTpInterface *subcommand options*

DESCRIPTION

The *mplsTpInterface* command holds the information related to a single interface on the simulated router.

STANDARD OPTIONS

enabled

Enables the use of the simulated interface.

dutMacAddress

The MAC address of the DUT.

interfaces

The number of interfaces.

COMMANDS

The **mplsTpServer** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsTpInterface config *option value*

Modifies the configuration options of the mplsTpInterface. If no *option* is specified, returns a list describing all of the available options (see STANDARD OPTIONS) for mplsTpInterface.

mplsTpInterface addInterface

Adds a simulated interface.

Refer to mplsTpInterface Subcommands for the list of subcommands.

EXAMPLE

See *EXAMPLE* section of *mplsTpServer*

NAME - mplsTpGeneralLearnedInfo

mplsTpGeneralLearnedInfo — Holds lists of the general learned route information.

SYNOPSIS

mplsTpGeneralLearnedInfo *subcommand options*

DESCRIPTION

The *mplsTpGeneralLearnedInfo* command Holds lists of the general learned route information.

STANDARD OPTIONS

incomingLabelOuterInner

(read only) The incoming label for outer and inner.

outgoingLabelOuterInner

(read only) The outgoing label for outer inner.

type

(read only) Indicates the type.

localPwStatus

(read only) The local PW status.

remotePwStatus

(read only) The remote PW status.

role

(read only) Indicates the role.

continuityCheckLocalState

(read only) The continuity check local state. Possible values include:

- na
- bfdDown
- bfdInit
- bfdUp
- y1731Down
- y1731Init
- y1731Up

continuityCheckRemoteState

(read only) The continuity check remote state. Possible values include:

- na
- bfdDown
- bfdInit
- bfdUp
- y1731Down
- y1731Init
- y1731Up

apsLocalFaultPath

(read only) The aps local fault path. Possible values include:

- working
- protect
- both
- none
- na

apsRemoteFaultPath

(read only) The aps remote fault path. Possible values include:

- na
- working
- protect
- both
- none

apsLocalDataPath

(read only) The aps local data path. Possible values include:

- working
- protect
- na

apsRemoteDataPath

(read only) The aps remote data path. Possible values include:

- protect
- na
- working

apsLocalState

(read only) The aps local state. Possible values include:

- na
- apsNoRequest
- apsLockoutOfProtection

- apsSignalFailOnWorking
- apsManualSwitch
- apsWaitToRestore
- apsDoNotRevert
- apsExercise
- apsReverseRequest
- pscNormal
- pscUnavailable
- pscProtectingAdmin
- pscProtectingFailure
- pscWaitToRevert
- pscDoNotRevert
- apsSignalFailOnProtection
- apsForceSwitch

apsRemoteRequestState

(read only) The aps remote request state. Possible values include:

- na
- apsNoRequest
- apsLockoutOfProtection
- apsSignalFailOnWorking
- apsManualSwitch
- apsWaitToRestore
- apsDoNotRevert
- apsExercise
- apsReverseRequest
- pscNormal
- pscUnavailable
- pscProtectingAdmin
- pscProtectingFailure
- pscWaitToRevert
- pscDoNotRevert
- apsSignalFailOnProtection
- apsForceSwitch

aisState

(read only) The ais state.

lckState

(read only) The lck state.

ldi

(read only) The ldi value.

aisTx

(read only) The ais transmit value.

aisRx

(read only) The ais receive value.

lckTx

(read only) The lck transmit value.

lckRx

(read only) The lck receive value.

lastAlarmDuration

(read only) The last alarm duration.

timeSinceLastAlarm

(read only) The time since last alarm.

alarmTypeAis

(read only) The ais alarm type.

alarmTypeLck

(read only) The lck alarm type.

COMMANDS

The **mplsTpGeneralLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsTpGeneralLearnedInfo config cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **mplsTpGeneralLearnedInfo** command.

EXAMPLE

```
mplsTpRouter refreshLearnedInformation

mplsTpRouter getGeneralLearnedInformationList

mplsTpRouter getFirstGeneralLearnedInformation

mplsTpRouter refreshLearnedInformation

mplsTpGeneralLearnedInfo cget -incomingLabelOuterInner

mplsTpGeneralLearnedInfo cget -outgoingLabelOuterInner
mplsTpGeneralLearnedInfo cget -type

mplsTpGeneralLearnedInfo cget -localPwStatus

mplsTpGeneralLearnedInfo cget -remotePwStatus
```

NAME - mplsTpLmLearnedInfo

mplsTpLmLearnedInfo — Holds lists of the lm learned information.

SYNOPSIS

mplsTpLmLearnedInfo *subcommand options*

DESCRIPTION

The *mplsTpLmLearnedInfo* command holds lists of the lm learned information.

STANDARD OPTIONS

outgoingLabelOuterInner

(read only) The outer inner outgoing label.

incomingLabelOuterInner

(read only) The outer inner incoming label.

type

(read only) Indicates the type.

lmQueriesSent

(read only) The number of lm queries sent.

lmResponsesReceived

(read only) The number of lm responses received.

lastLmResponseMyTx

(read only) The last lm transmit response.

lastLmResponseDutRx

(read only) The last lm receive response.

lastLmResponseDutTx

(read only) The last lm transmit response from the DUT.

lmRemoteUsing64Bit

(read only) The lm remote using 64 bit.

COMMANDS

The **mplsTpLmLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsTpLmLearnedInfo config cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **mplsTpLmLearnedInfo** command.

EXAMPLE

```
mplsTpRouter refreshLearnedInformation

mplsTpRouter getLmLearnedInformationList

mplsTpRouter getFirstLmLearnedInformation

mplsTpLmLearnedInfo cget -incomingLabelOuterInner

mplsTpLmLearnedInfo cget -lastLmResponseDutRx

mplsTpLmLearnedInfo cget -lastLmResponseDutTx

mplsTpLmLearnedInfo cget -lmQueriesSent

mplsTpLmLearnedInfo cget -lmRemoteUsing64Bit

mplsTpLmLearnedInfo cget -lmResponsesReceived

mplsTpLmLearnedInfo cget -outgoingLabelOuterInner

mplsTpLmLearnedInfo cget –type

mplsTpRouter getNextLmLearnedInformation
```

NAME - mplsTpDmLearnedInfo

mplsTpDmLearnedInfo — Holds lists of the dm learned information.

SYNOPSIS

mplsTpDmLearnedInfo *subcommand options*

DESCRIPTION

The *mplsTpDmLearnedInfo* command holds lists of the dm learned information.

STANDARD OPTIONS

outgoingLabelOuterInner

(read only) The outer inner outgoing label.

incomingLabelOuterInner

(read only) The outer inner incoming label.

type

(read only) Indicates the type.

dmQueriesSent

(read only) The number of dm queries sent.

dmResponsesReceived

(read only) The number of dm responses received.

averageStrictRtt

(read only) The average number of strict rtt.

averageLooseRtt

(read only) The average number of loose rtt.

minStrictRtt

(read only) The minimum strict rtt.

minLooseRtt

(read only) the minimum loose rtt.

maxStrictRtt

(read only) The maximum strict rtt.

maxLooseRtt

(read only) The maximum loose rtt.

averageStrictRttVariation

(read only) The average strict rtt variation.

averageLooseRttVariation

(read only) The average loose rtt variation.

COMMANDS

The **mplsTpDmLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsTpDmLearnedInfo config cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **mplsTpDmLearnedInfo** command.

EXAMPLE

```
mplsTpRouter refreshLearnedInformation

mplsTpRouter getGeneralLearnedInformationList

mplsTpRouter getFirstDmLearnedInformation

mplsTpDmLearnedInfo cget -averageLooseRTT

mplsTpDmLearnedInfo cget -averageLooseRTTVariation]

mplsTpDmLearnedInfo cget -averageStrictRTT

mplsTpDmLearnedInfo cget -averageStrictRTTVariation

mplsTpDmLearnedInfo cget -dmQueriesSent

mplsTpDmLearnedInfo cget -dmResponsesReceived

mplsTpDmLearnedInfo cget -incomingLabelOuterInner

mplsTpDmLearnedInfo cget -maxLooseRTT

mplsTpDmLearnedInfo cget -maxStrictRTT

mplsTpDmLearnedInfo cget -minLooseRTT

mplsTpDmLearnedInfo cget -minStrictRTT

mplsTpDmLearnedInfo cget -outgoingLabelOuterInner

mplsTpDmLearnedInfo cget –type

mplsTpRouter getNextDmLearnedInformation
```

NAME - mplsTpPingLearnedInfo

mplsTpPingLearnedInfo — Holds lists of the ping learned information.

SYNOPSIS

mplsTpPingLearnedInfo *subcommand options*

DESCRIPTION

The *mplsTpPingLearnedInfo* command holds lists of the ping learned information.

STANDARD OPTIONS

outgoingLabelOuterInner

(read only) The outer inner outgoing label.

incomingLabelOuterInner

(read only) The outer inner incoming label.

type

(read only) Indicates the type.

senderHandle

(read only) The sender handle information.

sequenceNumber

(read only) Indicates the sequence number.

reachability

(read only) Indicates the reachability information.

rtt

(read only) The rtt value.

returnCode

(read only) The return code value.

returnSubcode

(read only) The return subcode value.

COMMANDS

The **mplsTpPingLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsTpPingLearnedInfo config cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **mplsTpPingLearnedInfo** command.

EXAMPLE

```
mplsTpRouter refreshLearnedInformation

mplsTpRouter getGeneralLearnedInformationList

mplsTpRouter getFirstPingLearnedInformation

mplsTpPingLearnedInfo cget -incomingLabelOuterInner

mplsTpPingLearnedInfo cget -outgoingLabelOuterInner

mplsTpPingLearnedInfo cget -reachability

mplsTpPingLearnedInfo cget -returnSubcode

mplsTpPingLearnedInfo cget -senderHandle

mplsTpPingLearnedInfo cget -sequenceNumber

mplsTpPingLearnedInfo cget -type

mplsTpRouter getNextPingLearnedInformation
```

NAME - mplsTpTracerouteLearnedInfo

mplsTpTracerouteLearnedInfo — Holds lists of the ping learned information.

SYNOPSIS

mplsTpTracerouteLearnedInfo *subcommand options*

DESCRIPTION

The *mplsTpTracerouteLearnedInfo* command holds lists of the Traceroute learned information.

STANDARD OPTIONS

outgoingLabelOuterInner

(read only) The outer inner outgoing label.

incomingLabelOuterInner

(read only) The outer inner incoming label.

type

(read only) Indicates the type.

numberOfReplyingHops

(read only) The total number of replying hops.

senderHandle

(read only) The sender handle information.

reachability

(read only) The reachability information.

COMMANDS

The **mplsTpTracerouteLearnedInfo** command is invoked with the following subcommands. If no subcommand is specified, returns a list of all subcommands available.

mplsTpTracerouteLearnedInfo config cget *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **mplsTpTracerouteLearnedInfo** command

EXAMPLE

```
mplsTpRouter refreshLearnedInformation

mplsTpRouter getGeneralLearnedInformationList

mplsTpRouter getTracerouteLearnedInformationList
mplsTpRouter getFirstTracerouteLearnedInformation

mplsTpTracerouteLearnedInfo cget -incomingLabelOuterInner

mplsTpTracerouteLearnedInfo cget -numberOfReplyingHops

mplsTpTracerouteLearnedInfo cget -outgoingLabelOuterInner

mplsTpTracerouteLearnedInfo cget -reachability

mplsTpTracerouteLearnedInfo cget -senderHandle

mplsTpTracerouteLearnedInfo cget -type

mplsTpRouter getNextTracerouteLearnedInformation
```

NAME - elmiUniStatus

elmiUniStatus —

SYNOPSIS

elmiUniStatus *subcommand options*

DESCRIPTION

Attributes

ceVlanIdEvcMapType

enabled

uniIdentifier

uniIdentifierLength

APIs Supported

addBwProfile

delBwProfile

getBwProfile

setBwProfile

getFirstBwProfile

getNextBwProfile

clearAllBwProfiles

showBwProfileNames

NAME - elmiUniStatus

elmiUniStatus —

SYNOPSIS

elmiUniStatus *subcommand options*

DESCRIPTION

Attributes

APIs Supported

Index