Cisco ME 1200 Series Carrier Ethernet Access Devices UCS Controller Configuration Guide, Cisco IOS 15.6(1)SN and Later Releases

First Published: November 20, 2015

Last Modified: February 05, 2016

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

This preface contains information about the Cisco ME 1200 Series Carrier Ethernet Access Device.

Audience

This guide is for the person configuring the Cisco ME 1200 Series Carrier Ethernet Access Devices, hereafter known as Cisco ME 1200 NID.

Document Conventions

This document uses the following conventions:

Convention	Description
^ or Ctrl	Both the ^ symbol and Ctrl represent the Control (Ctrl) key on a keyboard. For example, the key combination **^D** or **Ctrl-D** means that you hold down the Control key while you press the D key. (Keys are indicated in capital letters but are not case sensitive.)
bold font	Commands and keywords and user-entered text appear in **bold** font.
Italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
`Courier font`	Terminal sessions and information the system displays appear in `courier` font.
`Bold Courier font`	Bold Courier font indicates text that the user must enter.
[x]	Elements in square brackets are optional.

<table>
<tr><th>Convention</th><th>Description</th></tr>
<tr><td>...</td><td>An ellipsis (three consecutive nonbolded periods without spaces) after a syntax element indicates that the element can be repeated.</td></tr>
<tr><td>|</td><td>A vertical line, called a pipe, indicates a choice within a set of keywords or arguments.</td></tr>
<tr><td>[x | y]</td><td>Optional alternative keywords are grouped in brackets and separated by vertical bars.</td></tr>
<tr><td>{x | y}</td><td>Required alternative keywords are grouped in braces and separated by vertical bars.</td></tr>
<tr><td>[x {y | z}]</td><td>Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.</td></tr>
<tr><td>string</td><td>A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.</td></tr>
<tr><td>< ></td><td>Nonprinting characters such as passwords are in angle brackets.</td></tr>
<tr><td>[ ]</td><td>Default responses to system prompts are in square brackets.</td></tr>
<tr><td>!, #</td><td>An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.</td></tr>
</table>

Reader Alert Conventions

This document uses the following conventions for reader alerts:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Tip Means *the following information will help you solve a problem.*

Warning **Means *reader be warned.* In this situation, you might perform an action that could result in bodily injury.**

Related Documentation

These documents provide information about the switches and are available from this Cisco.com site:

http://www.cisco.com/c/en/us/support/switches/me-1200-series-carrier-ethernet-access-devices/tsd-products-support-general-information.html

- *Release Notes for the Cisco ME 1200 Series Carrier Ethernet Access Devices*

Note Before installing, configuring, or upgrading the switch, see the release notes on Cisco.com for the latest information.

- *Cisco ME 3800x and ME 3600x Switches Software Configuration Guide*
- *Cisco Regulatory Compliance and Safety Information for Cisco ME 1200 Series Carrier Ethernet Access Devices*

For information on supported MIBs, see ftp://ftp.cisco.com/pub/mibs/ME1200-MIBS/.

CHAPTER 1

Installing UCS NID Controller

This document describes the pre-requisites of the UCS controller server, its software requirements and their installations.

Prerequisites for Installing UCS Controller

Prerequisites for UCS Server:

- Model: UCS C200 M2 server (Can be any 1RU UCS server or blade server).
- CPU 2 Intel Xeon E5520 2 processor sockets, 8 CPU cores @2.27GHz.
- RAM upto 192 GB.
- Disk Storage: Database store upto 460GB for hosting all VMs.

Prerequisites for Virtual Machine hosting UCS NID Controller:

- OS: RHEL v 6.5 or 7, CentOS 5.2
- RAM: 4GB
- NICS: 1 or more Virtual NICs
- Storage: 40GB per VM

Prerequisites for UCS NID Controller software:

- Python package (version 2.4.3 or higher)
- Sqlite (SQLite version 3.3.6)

Installing UCS Controller

Perform the following tasks to install the UCS controller:

1. Extract the tar file.
2. Execute cd ucs-ctrlr.
3. Run "bash".
4. Run"source ucsenv.sh".
5. Execute "nidctrlr" to get to the device manager shell.
6. Once application starts, user is prompted for login and password. To create a new username, initially login as dummy root.
    - User name: root
    - Password: root

    This will prompt a new username and password to be created and used for subsequent sessions.
7. Once authentication is successful, NIDs can be added statically to the database by specifying NID IP Address.
8. To configure a specific nid, execute command "configure nid <nid-id>" and configure/review/commit required operations on NID.

CHAPTER 2

Configuration Management

This chapter helps you to get started and describes how to configure the initial switch configuration for the Cisco ME 1200 NID. This chapter also describes how to manage Cisco ME 1200 NID configurations.

Prerequisites for Managing Configurations

- NID must be added to the UCS controller.
- NID must be accessible from the UCS controller.

Restrictions for Managing Configurations

- The option **show run** command is not supported.

Information About Configuration Management

Configuration management on ME1200 stores the configurations in XML format. A startup-config.xml file is generated containing all relevant configuration to be applied on the ME1200. A current running-config.xml can also be generated and copied to a TFTP server. This complete XML configuration file can be viewed using a suitable XML editor.

Understanding the Boot Process

The Cisco ME 1200 NID device is not connected to any network soon after it is unpacked. To start your Cisco ME 1200 NID, you need to follow the procedures in the hardware installation guide about installing and powering on the switch. This document describes login and setting up the initial configuration (IP address, subnet mask, default gateway, secret and Telnet passwords, and so forth) of the Cisco ME 1200 NID.

The boot loader provides access to the flash file system before the operating system is loaded. Normally, the boot loader is used only to load, uncompress, and launch the operating system. After the boot loader gives the operating system control of the CPU, the boot loader is not active until the next system reset or power-on.

Before you can assign switch information, make sure you have connected a PC or terminal to the console port, and configured the PC or terminal-emulation software baud rate and character format to match these of the switch console port:

- Baud rate default is 115200.
- Data bits default is 8.
- Stop bits default is 1.
- Parity settings default is none.

Table 1: Default Boot Configuration

Feature	Default Setting
Operating system software image	The device attempts to automatically boot the system using information in the BOOT environment variable. If the variable is not set, the Cisco ME 1200 NID attempts to load and execute the first executable image it can by performing a recursive, depth-first search throughout the flash file system. In a depth-first search of a directory, each encountered subdirectory is completely searched before continuing the search in the original directory.
Configuration file	Configured devices use the startup-config.xml file stored on the system board in flash memory. A new switch has no configuration file.

Getting Started

Perform the following tasks to get started with the Cisco ME 1200 NID:

1. Add the Cisco ME 1200 NID using the **platform nid-controller** command on the UCS controller. To dynamically configure one or more Cisco ME1200 NIDs to the UCS Controller, see Auto Discovery of Cisco ME 1200 NIDs, on page 63.
2. Verify if the NID is added on the controller using the **show platform nid-controller** command on the UCS controller.
3. Create Layer 2 VLANs on the NID.
4. Modify switchport mode as Trunk on the NID.
5. Assign IP address to VLAN Interface.
6. Configure Default IP Route.
7. Create Startup-config.xml file.

Step 1—Adding the Cisco ME 1200 NID to the UCS Controller

Note Effective Cisco IOS Release 15.4(2)SN, Cisco ME3600 switch will act as controller for the Cisco ME 1200 NID.

Effective Cisco IOS Release 15.5(1)SN, the Cisco ASR 920 Series Routers can also act as controller for the Cisco ME 1200 NID.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**platform nid-controller add 192.0.2.1** **Example:** `UCS(config)# platform nid-controller add 192.0.2.1`	Adds the Cisco ME 1200 NID using the **platform nid-controller** command on the controller. The Cisco ME 1200 NID is shipped with the factory default IP address 192.0.2.1.
Step 2	**end** **Example:** `UCS(config)# end`	Exits to the Privileged EXEC mode.

Configuration Example

```
UCS# configure terminal
UCS(config)# platform nid-controller add 192.0.2.1
UCS(config)# end
```

Step 2—Verifying the Cisco ME 1200 NID Addition onto the UCS Controller

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**show platform nid-controller nids** **Example:** `UCS# show platform nid-controller nids`	Displays the Cisco ME 1200 NID IP addresses that are added to the controller.

	Command or Action	Purpose
Step 3	**end** **Example:** `UCS# end`	Exits the Privileged EXEC mode.

Configuration Example

```
UCS# configure terminal
UCS# show platform nid-controller nids

NID_ID IP Address   NID Type  Discovery GROUPS

1      1.1.1.1       ME1200   STATIC     g1
2      7.25.17.223   ME1200   STATIC     g2,g4
3      7.25.16.220   ME1200   STATIC     g1,g2,g4

UCS# end
```

Step 1—Creating Layer 2 VLANs on the NID

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 2	**createVlanCommand createVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 100-105`	Creates the VLAN list. The valid values are from 1 to 4095.
Step 3	**createVlanCommand review** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand review`	Displays the configuration.
Step 4	**createVlanCommand commit** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand commit`	Sends the configuration to the NID.
Step 5	**ProvisionPortVlanPortTypeshow** **Example:** `UCS(ProvisionPortVlanPortType)# showVlans`	Displays the Vlan lists.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 100-105
UCS(ProvisionPortVlanPortType)# createVlanCommand review

Commands in queue:
     createVlanCommand createVlanReq vlan_list 100-105

UCS(ProvisionPortVlanPortType)# createVlanCommand commit

     Vlan Creation Commit Success!!!

UCS(ProvisionPortVlanPortType)# exit
```

Step 2—Modifying Switchport Mode as Trunk

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 2	**modifySwPort modifySWPortConfig interface** *interface_id* **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interaface 4`	Configure the switchport configuration on the defined interface.
Step 3	**modifySwPort modifySWPortConfig mode trunk** {**allowed vlan** {**add** {**all** | **vlan_list** *vlan_list* } | **remove** {**all** | **vlan_list** *vlan_list* }} | {**native vlan** *vlan_list* } **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 100-105`	Sets the mode to TRUNK. • **allowed—**Sets the allowed VLAN characteristics when interface is in trunk mode. • **add—**Adds either all VLANs or specified VLANs to the current list. • **remove—**Rremoves either all VLANs or specified VLANs from the current list. • *vlan_id*—The VLAN ID. The valid values are from 0 to 4095.

	Command or Action	Purpose
Step 4	**modifySwPort review** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort review`	Displays the configuration.
Step 5	**modifySwPort commit** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort commit`	Sends the configuration to the NID.
Step 6	**ProvisionPortVlanPortTypeshow** **Example:** `UCS(ProvisionPortVlanPortType)# showSwPort`	Displays the commit, flush or review commands in queue for switchport configurtion.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interaface 4
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 100-105
UCS(ProvisionPortVlanPortType)# modifySwPort review

Commands in queue:
   modifySwPort modifySWPortConfig interaface 4
   modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 100-105

UCS(ProvisionPortVlanPortType)# modifySwPort commit

   ModifySwPort_Output.modifySwPortConfigResp = 0

   Modify SwitchPort Commit Success!!!

UCS(ProvisionPortVlanPortType)# exit
```

Step 3— Assigning IP Address to VLAN Interface

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanType`	Enters the ProvisionPortVlanPortType mode.

	Command or Action	Purpose
Step 2	**createIntVlan createIntVlanReq vlan_id** *vlan_id* **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan)_id 100`	Creates the interface VLAN list.

Step 3	**createIntVlan createIntVlanReq** {**address** {**ipv4** {**dhcp** | **ipv4-address**} | **ipv6 ipv6-address** *ipv6-address*} | **vlan_id** *vlan_id*}
	
	Example:
	`UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address address 22.22.22.3`
	`UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address mask 255.255.255.0`
	`UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv6 ipv6_address 2001:4::1/64`
		Creates the interface VLAN on the specified IPv4 or IPv6 address, or VLAN ID.

Step 4	**createIntVlan review** **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan review`	Displays the createIntVlan configuration.
Step 5	**createIntVlan commit** **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan commit`	Sends createIntVlan configuration to the Cisco ME 1200 NID .
Step 6	**ProvisionPortVlanPortTypeshow** **Example:** `UCS(ProvisionPortVlanPortType)# showIntVlan`	Displays the commit, flush or review the commands for VLAN interfaces.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

```
Example 1: IPv4
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan_Id 100
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address
address 22.22.22.3
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address
mask 255.255.255.0
UCS(ProvisionPortVlanPortType)# createIntVlan review

Commands in queue:
   createIntVlan createIntVlanReq vlan_id 100
   createIntVlan createIntVlanReq address ipv4 ipv4_address address 22.22.22.3
   createIntVlan createIntVlanReq address ipv4 ipv4_address mask 255.255.255.0

UCS(ProvisionPortVlanPortType)# createIntVlan commit
```

```
   CreateIntVlan_Output.createIntVlanResp = 0

   Create Interface Vlan Commit Success!!!

UCS(ProvisionPortVlanPortType)# exit
Example 2: IPv6
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan_Id 100
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv6 ipv6-address
2001:4::1/64
UCS(ProvisionPortVlanPortType)# createIntVlan review

Commands in queue:
   createIntVlan createIntVlanReq vlan_id 100
   createIntVlan createIntVlanReq address ipv6 ipv6-address 2001:4::1/64

UCS(ProvisionPortVlanPortType)# createIntVlan commit

   CreateIntVlan_Output.createIntVlanResp = 0

   Create Interface Vlan Commit Success!!!

UCS(ProvisionPortVlanPortType)# exit
```

Step 4—Configuring IP Route

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.

Step 2	**setiproute setIpRouteReq** {**gateway_ip** *WORD* | **ipv4_address** *WORD* | **ipv4_mask** *WORD*}
	
	Example:
	`UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_address 10.0.144.0`
	
	`UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_mask 255.255.255.0`
	
	`UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq gateway_ip 10.0.0.1`
		Configures the IP Route.
		• **gateway_ip—**Specifies the gateway IPv4 address.
		  ◦ *WORD*—IPv4 address.
		• **ipv4_address—**Specifies the IPv4 Network/Address.
		  ◦ *WORD*—IPv4 Network/Address.
		• **ipv4_mask—**Specifies the IPv4 mask.
		  ◦ *WORD*—IPv4 mask.

Step 3	**setiproute review** **Example:** `UCS(ProvisionNIDMgmtType)# setiproute review`	Displays the configuration.

	Command or Action	Purpose
Step 4	**getiproute commit** **Example:** `UCS(ProvisionNIDMgmtType)# setiproute commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_address 10.0.144.0
UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_mask 255.255.255.0
UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq gateway_ip 10.0.0.1

UCS(ProvisionNIDMgmtType)# setiproute review
Commands in Queue:
   setIpRoute setIpRouteReq ipv4_address 10.0.144.0
   setIpRoute setIpRouteReq ipv4_mask 255.255.255.0
   setIpRoute setIpRouteReq gateway_ip 10.0.0.1

UCS(ProvisionNIDMgmtType)# setiproute commit

   Setiproute Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Step 5—Creating Startup-config.xml File

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.

Step 2	**copyConfig copyConfigRequest** {**src** {**default_config** | **flash** *WORD* | **running_config** | **startup_config** | **tftp** *WORD*} | **dst** {**flash** *WORD* | **running_config** | **startup_config** | **tftp** *WORD*}}
	
	Example:
	`UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest src running_config`
	`UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest dst startup_config`
		Copies the configuration.
		• **src**—Specifies the source location.
		◦ **default**—Copies to the default-config file.
		◦ **flash**—Copies onto the flash.
		◦ *WORD*—Filename. The format is flash:<filename>. For example, flash:ToTest.

	Command or Action	Purpose
		◦ **running_config—**Copies to the running-config file. ◦ **startup_config—**Copies to the startup-config file. ◦ **tftp—**Copies to the TFTP server. ◦ *WORD*—TFTP filename. The format is tftp://server/path-and-filename. For example, tftp://10.0.0.221/ToTest. • **dst—**Specifies the destination location. ◦ **flash—**Copies onto the flash. ◦ *WORD*—Filename. The format is flash:<filename>. For example, flash:ToTest. ◦ **running_config—**Copies to the running-config file. ◦ **startup_config—**Copies to the startup-config file. ◦ **tftp—**Copies to the TFTP server. ◦ *WORD*—TFTP filename. The format is tftp://server/path-and-filename. For example, tftp://10.0.0.221/ToTest.
Step 3	**copyConfig review** **Example:** `UCS(ProvisionConfigMGMTPortType)# copyConfig review`	Displays the configuration.
Step 4	**copyConfig commit** **Example:** `UCS(ProvisionConfigMGMTPortType)# copyConfig commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionConfigMGMTPortType)# exit`	Exits the ProvisionConfigMGMTPortType mode.

Configuration Example

```
UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest src running_config
UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest dst startup_config
```

```
UCS(ProvisionConfigMGMTPortType)# copyConfig review
Commands in Queue:
   copyConfig copyConfigRequest src running-config
   copyConfig copyConfigRequest dst startup-config

UCS(ProvisionConfigMGMTPortType)# copyConfig commit

  CopyConfig Commit Success!!!

UCS(ProvisionConfigMGMTPortType)# exit
```

How to Manage Configurations

Listing Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 2	**listConfigs listConfigsReq** **Example:** `UCS(ProvisionConfigMGMTPortType)# listConfigs listConfigsReq`	Lists the configuration.
Step 3	**listConfigs review** **Example:** `UCS(ProvisionConfigMGMTPortType)# listConfigs review`	Displays the configuration.
Step 4	**listConfigs commit** **Example:** `UCS(ProvisionConfigMGMTPortType)# listConfigs commit`	Fetches listing of flash configuration on the NID.
Step 5	**exit** **Example:** `UCS(ProvisionConfigMGMTPortType)# exit`	Exits the ProvisionConfigMGMTPortType mode.

Configuration Example

```
UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# listConfigs listConfigsReq
UCS(ProvisionConfigMGMTPortType)# listConfigs review

Commands in Queue:
   listConfigs listConfigsReq
```

```
UCS(ProvisionConfigMGMTPortType)# listConfigs commit

   ListConfigs_Output.configFiles.files[0].fileName = 'default-config'
   ListConfigs_Output.configFiles.files[0].fileSize = '    1100'
   ListConfigs_Output.configFiles.files[0].timeStamp = '1970-01-01 00:00:00'
   ListConfigs_Output.configFiles.files[0].permissions = 'r-'
   ListConfigs_Output.configFiles.files[1].fileName = 'startup-config'
   ListConfigs_Output.configFiles.files[1].fileSize = '    1552'
   ListConfigs_Output.configFiles.files[1].timeStamp = '1970-01-01 00:04:44'
   ListConfigs_Output.configFiles.files[1].permissions = 'rw'
   ListConfigs_Output.configFiles.files[2].fileName = 'startup-config.xml'
   ListConfigs_Output.configFiles.files[2].fileSize = '  149016'
   ListConfigs_Output.configFiles.files[2].timeStamp = '2014-03-25 10:15:58'
   ListConfigs_Output.configFiles.files[2].permissions = 'rw'
   ListConfigs_Output.configFiles.files[3].fileName = 'Totest'
   ListConfigs_Output.configFiles.files[3].fileSize = '  149016'
   ListConfigs_Output.configFiles.files[3].timeStamp = '2014-03-25 10:20:31'
   ListConfigs_Output.configFiles.files[3].permissions = 'rw'

   ListConfigs Commit Success!!!

UCS(ProvisionConfigMGMTPortType)# exit
```

Verifying Configuration Version

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionConfigMGMPortType** **Example:** `UCS# ProvisionConfigMGMPortType`	Enters the ProvisionConfigMGMPortType mode.
Step 2	**showVersion showVersionReq** **Example:** `UCS(ProvisionConfigMGMTPortType)# showVersion showVersionReq`	Displays the configuration.
Step 3	**showVersion review** **Example:** `UCS(ProvisionConfigMGMTPortType)# showVersion review`	Displays the configuration.
Step 4	**showVersion commit** **Example:** `UCS(ProvisionConfigMGMTPortType)# showVersion commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionConfigMGMTPortType)# exit`	Exits the ProvisionConfigMGMTPortType mode.

Configuration Example

Note The Active.Image is the current image and Alternative.Image is the backup image. While upgrading the image, you can choose to swap Active.Image with Alternate.Image.

```
UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# showVersion showVersionReq
UCS(ProvisionConfigMGMTPortType)# showVersion review

Commands in Queue:
   showVersion showVersionReq

UCS(ProvisionConfigMGMTPortType)# showVersion commit

   ShowVersion_Output.showVersionResp.Active.Image = 'me1200-universal-mz.154-2.SN.dat'
   ShowVersion_Output.showVersionResp.Active.Version = 'ME1200 OS Software Build 15.4-2.SN'

   ShowVersion_Output.showVersionResp.Active.Date = 'Fri Mar 21 10:08:34 PDT 2014'
   ShowVersion_Output.showVersionResp.Alternative.Image = 'me1200-universal-mz.dat'
   ShowVersion_Output.showVersionResp.Alternative.Version = 'ME1200 OS Software Build
15.4-2.SN'
   ShowVersion_utput.showVersionResp.Alternative.Date = 'Fri Mar 21 05:56:50 PDT 2014'

   ShowVersion Commit Success!!!

UCS(ProvisionConfigMGMTPortType)# exit
```

Copying Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionConfigMGMPortType** **Example:** `UCS# ProvisionConfigMGMPortType`	Enters the ProvisionConfigMGMPortType mode.

Step 2	**copyConfig copyConfigRequest** {**src** {**default_config** | **flash** *WORD* | **running_config** | **startup_config** | **tftp** *WORD*} | **dst** {**flash** *WORD* | **running_config** | **startup_config** | **tftp** *WORD*}}
	
	Example:
	`UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest src running_config`
	`UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest dst startup_config`
	In this example, the Source is the running-config, and the Destination is the startup-config. When you use these commands for the first time on the Cisco ME 1200 NID, the NID creates the startup-config.xml file in the flash,
		Copies the configuration.
		• **src**—Specifies the source location.
		◦ **default**—Copies to the default-config file.
		◦ **flash**—Copies onto the flash.
		◦ *WORD*—Filename. The format is flash:<filename>. For example, flash:ToTest.
		◦ **running_config**—Copies to the running-config file.
		◦ **startup_config**—Copies to the startup-config file.
		◦ **tftp**—Copies to the TFTP server.

	Command or Action	Purpose
	which is used during the device boot-up. When the device reloads for the first time, it uses the startup-config.xml file.	◦ *WORD*—TFTP filename. The format is tftp://server/path-and-filename. For example, tftp://10.0.0.221/ToTest. • **dst**—Specifies the destination location. ◦ **flash**—Copies onto the flash. ◦ *WORD*—Filename. The format is flash:<filename>. For example, flash:ToTest. ◦ **running_config**—Copies to the running-config file. ◦ **startup_config**—Copies to the startup-config file. ◦ **tftp**—Copies to the TFTP server. ◦ *WORD*—TFTP filename. The format is tftp://server/path-and-filename. For example, tftp://10.0.0.221/ToTest.
Step 3	**copyConfig review** **Example:** `UCS(ProvisionConfigMGMTPortType)# copyConfig review`	Displays the configuration.
Step 4	**copyConfig commit** **Example:** `UCS(ProvisionConfigMGMTPortType)# copyConfig commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionConfigMGMTPortType)# exit`	Exits the ProvisionConfigMGMTPortType mode.

Configuration Example

```
UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest src running_config
UCS(ProvisionConfigMGMTPortType)# copyConfig copyConfigRequest dst startup_config
UCS(ProvisionConfigMGMTPortType)# copyConfig review

Commands in Queue:
   copyConfig copyConfigRequest src running_config
   copyConfig copyConfigRequest dst startup_config

UCS(ProvisionConfigMGMTPortType)# copyConfig commit
```

```
   CopyConfig Commit Success!!!
UCS(ProvisionConfigMGMTPortType)# exit
```

Note When the running-config file is copied to the TFTP server, by default, it stores the file in the XML format. You need not mention the XML extension explicitly. This hold good vice versa as well.

Note When the Source is TFTP and the Destination is running-config, the TFTP file *appends* to the existing running-config, and does not overwrite the running-config file.

Deleting Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionConfigMGMPortType** **Example:** `UCS# ProvisionConfigMGMPortType`	Enters the ProvisionConfigMGMPortType mode.
Step 2	**deleteConfFile configName** {**configFile***WORD*} **Example:** `UCS(ProvisionConfigMGMTPortType)# deleteConfFile configName configFile flash:ToTest`	Deletes the configuration. • **configFile**—Specifies the configuration file to be deleted. ◦ *WORD*—File name. The format is **flash:filename**.
Step 3	**deleteConfFile review** **Example:** `UCS(ProvisionConfigMGMTPortType)# deleteConfFile review`	Displays the configuration.
Step 4	**deleteConfFile commit** **Example:** `UCS(ProvisionConfigMGMTPortType)# deleteConfFile commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionConfigMGMTPortType)# exit`	Exits the ProvisionConfigMGMTPortType mode.

Configuration Example

```
UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# deleteConfFile configName configFile flash:ToTest
UCS(ProvisionConfigMGMTPortType)# deleteConfFile review

Commands in Queue:
   deleteConfFile configName configFile flash:ToTest

UCS(ProvisionConfigMGMTPortType)# deleteConfFile commit

   DeleteConfFile Commit Success!!!

UCS(ProvisionConfigMGMTPortType)# exit
```

What to Do Next

Use the **listConfigs listConfigsReq** command to verify the delete action.

```
UCS(ProvisionConfigMGMTPortType)# listConfigs listConfigsReq
UCS(ProvisionConfigMGMTPortType)# listConfigs review
UCS(ProvisionConfigMGMTPortType)# listConfigs commit
```

Reloading the System

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 2	**reloadSystem reloadSystemReq** {**last_saved**} **Example:** `UCS(ProvisionConfigMGMTPortType)# reloadSystem reloadSystemReq last_saved`	Reloads the configuration. • **last_saved—**Reloads from the last saved configuration.
Step 3	**reloadSystem review** **Example:** `UCS(ProvisionConfigMGMTPortType)# reloadSystem review`	Displays the configuration.
Step 4	**reloadSystem commit** **Example:** `UCS(ProvisionConfigMGMTPortType)# reloadSystem commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionConfigMGMTPortType)# exit`	Exits the ProvisionConfigMGMTPortType mode.

Configuration Example

```
UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# reloadSystem reloadSystemReq last_saved
UCS(ProvisionConfigMGMTPortType)# reloadSystem review

Commands in Queue:
   reloadSystem reloadSystemReq last_saved

UCS(ProvisionConfigMGMTPortType)# reloadSystem commit

   ReloadSystem Commit Success!!!

UCS(ProvisionConfigMGMTPortType)# exit
```

Note To reboot the system with the last saved changes, copy the configurations from running-config (source) to startup-config.xml (destination) file before you reload the system. This ensures the system boots-up with the latest configuration.

Upgrading Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ProvisionConfigMGMPortType** **Example:** `UCS# ProvisionConfigMGMPortType`	Enters the ProvisionConfigMGMPortType mode.
Step 2	**upgradeImage upgradeImageRequest** {**swap** | **upgrade** {**tftp** *WORD*}} **Example:** `UCS(ProvisionConfigMGMTPortType)# upgradeImage upgradeImageRequest upgrade tftp tftp://<TFTP Server address>/<Path and file name>`	Upgrades the configuration. • **swap—**Swaps the configuration between Active and Alternate firmware images. **Note** When the Cisco ME1200 NID is upgraded, the previous image is stored as a Backup image in the flash. Use the **upgradeImage upgradeImageRequest swap** command to load the system with the old image. To view the Active and Alternative (backup) firmware images, see the . • **upgrade—**Upgrades the image. ◦ **tftp—**Specifies the TFTP location. ◦ *WORD*—TFTP details. Enter the tftp://server/path-and-filename.

	Command or Action	Purpose
Step 3	**upgradeImage review** **Example:** `UCS(ProvisionConfigMGMTPortType)# upgradeImage review`	Displays the configuration.
Step 4	**upgradeImage commit** **Example:** `UCS(ProvisionConfigMGMTPortType)# upgradeImage commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionConfigMGMTPortType)# exit`	Exits the ProvisionConfigMGMTPortType mode.

Configuration Example

```
Example 1: Upgrade

UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# upgradeImage upgradeImageRequest upgrade tftp tftp://<TFTP
 Server address>/<Path and file name>
ucs(ProvisionConfigMGMTPortType)# upgradeImage review

Commands in Queue:
   upgradeImage upgradeImageRequest upgrade tftp tftp://<TFTP Server add>/<Path and file
name>

UCS(ProvisionConfigMGMTPortType)# upgradeImage commit

   UpgradeImage Commit Success!!!

UCS(ProvisionConfigMGMTPortType)# exit

Example 2: Swap


UCS# ProvisionConfigMGMTPortType
UCS(ProvisionConfigMGMTPortType)# upgradeImage upgradeImageRequest swap
UCS(ProvisionConfigMGMTPortType)# upgradeImage review

Commands in Queue:
   upgradeImage upgradeImageRequest swap

UCS(ProvisionConfigMGMTPortType)# upgradeImage commit

   UpgradeSwap commit success !!!!

UCS(ProvisionConfigMGMTPortType)# exit
```

CHAPTER 3

Administering the Cisco ME 1200 NID

This chapter describes how to perform one-time operations to administer the Cisco ME 1200 NID.

For more information, see Administering the Switch.

Prerequisites for Administering the NID

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.

How to Administer the Cisco ME 1200 NID

Configuring the System Clock

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.

	Command or Action	Purpose

Step 3	**setclockConfig clockConfig** {**summerTime** {**endDate** *WORD* | **mode** {**disabled** | **nonRecurring** | **recurring**} | **name** *WORD* | **offSet** *Offset* | **startDate** *WORD*} | **timeZone** {**acronym** *WORD* | **hrOffSet** *hours* | **minOffSet** *mins*}}
	
	Example:
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime endDate 3-31-2016-23-59`
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime mode nonRecurring`
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime name MyClock`
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime offSet 3`
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime startDate 3-31-2014-23-59`
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig timeZone acronym IST`
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig timeZone hrOffSet 5`
	`UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig timeZone minOffSet 30`
		Configures the Cisco ME 1200 NID clock.
		• **summerTime—**Configures the summer (daylight savings) time.
		◦ **endDate—**Specifies the end date format.
		◦ *WORD*—end date depends on the mode. For recurring mode, the format is **week-day-month-hrs:min**.Where,
		• week ranges from 1 to 5.
		• day ranges from 1 to 7.
		• month ranges from 1 to 12.
		• hrs ranges from 0-23.
		• min ranges from 0-59.
		For nonrecurring mode, the format is **month-day-year-hrs:min**.
		• month ranges from 1 to 12.
		• day ranges from 1 to 31.
		• year ranges from 2000-2097.
		• hrs ranges from 0-23.
		• min ranges from 0-59.
		◦ **mode—**Specifies the day light saving time mode.
		◦ **disabled—**Disables the day light saving time.
		◦ **nonRecurring—**Specifies the standard mode.
		◦ **recurring—**Specifies the recurring mode.
		◦ **name—**Specifies the name of time zone in summer.
		◦ *WORD*—Clock name.
		◦ **offSet—**Specifies the Offset to add in minutes.
		◦ *Offset*—offset time. The range is from 1 to 1440 minutes.
		◦ **startDate—**Specifies the start date format.
		◦ *WORD*—start date depends on the mode. For recurring mode, the format is **week-day-month-hrs:min**.Where,

	Command or Action	Purpose
		• week ranges from 1 to 5. • day ranges from 1 to 7. • month ranges from 1 to 12. • hrs ranges from 0-23. • min ranges from 0-59. For nonrecurring mode, the format is **month-day-year-hrs:min**. Where, • month ranges from 1 to 12. • day ranges from 1 to 31. • year ranges from 2000-2097. • hrs ranges from 0-23. • min ranges from 0-59. • **timeZone**—Configures the time zone. ◦ **acronym**—Specifies the name of time zone. ◦ *WORD*—time zone name. ◦ **hrOffSet**—Specifies the off set hours from Universal Time Coordinated (UTC). ◦ *hours*—off set hour from UTC. The range is from minus(-) 23 to 23. ◦ **minOffSet**—Specifies the offset minutes from UTC. ◦ *mins*—off set minutes from UTC. The range is from 0-59.
Step 4	**setclockConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# setclockConfig review`	Displays the configuration.
Step 5	**setclockConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# setclockConfig commit`	Sends the configuration to the NID.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
Switch# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime endDate 3-31-2016-23-59
UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime mode nonRecurring
UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime name MyClock
UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime offSet 3
UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig summerTime startDate 3-31-2014-23-59

UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig timeZone acronym IST
UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig timeZone hrOffSet 5
UCS(ProvisionNIDMgmtType)# setClockConfig clockConfig timeZone minOffSet 30

UCS(ProvisionNIDMgmtType)# setclockConfig review
Commands in queue:
    setClockConfig clockConfig summerTime endDate 3-31-2016-23-59
    setClockConfig clockConfig summerTime mode nonRecurring
    setClockConfig clockConfig summerTime name MyClock
    setClockConfig clockConfig summerTime offSet 3
    setClockConfig clockConfig summerTime startDate 3-31-2014-23-59
    setClockConfig clockConfig timeZone acronym IST
    setClockConfig clockConfig timeZone hrOffSet 5
    setClockConfig clockConfig timeZone minOffSet 30

UCS(ProvisionNIDMgmtType)# setclockConfig commit
    SetClockConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Viewing the System Clock

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.

	Command or Action	Purpose
Step 3	**getClockConfig detailClock** **Example:** `UCS(ProvisionNIDMgmtType)# getClockConfig detailClock`	Displays the clock details.
Step 4	**getClockConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# getClockConfig review`	Displays the configuration.
Step 5	**getClockConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# getClockConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmt Type mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# getClockConfig detailClock
UCS(ProvisionNIDMgmtType)# getClockConfig review

Commands in queue:
    getClockConfig detailClock

UCS(ProvisionNIDMgmtType)# getClockConfig commit

    GetClockConfig_Output.clockConfig.timeZone.hrOffSet = 5
    GetClockConfig_Output.clockConfig.timeZone.minOffSet = 30
    GetClockConfig_Output.clockConfig.timeZone.acronym = 'IST'
    GetClockConfig_Output.clockConfig.summerTime.name = ''
    GetClockConfig_Output.clockConfig.summerTime.mode.t = 1
    GetClockConfig_Output.clockConfig.summerTime.mode.u.disabled = ''
    GetClockConfig_Output.clockConfig.summerTime.startDate = ''
    GetClockConfig_Output.clockConfig.summerTime.endDate = ''
    GetClockConfig_Output.clockConfig.summerTime.offSet = 1

    GetClockConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Verifying System Clock Settings

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**showclock showClockReq** {**show_clock**} **Example:** `UCS(ProvisionNIDMgmtType)# showclock showClockReq show_clock`	Displays the clock details.
Step 4	**showclock review** **Example:** `UCS(ProvisionNIDMgmtType)# showclock review`	Displays the configuration.
Step 5	**showclock commit** **Example:** `UCS(ProvisionNIDMgmtType)# showclock commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# showclock showClockReq show_clock
UCS(ProvisionNIDMgmtType)# showclock review

Commands in queue:
    showClock showClockReq show_clock

UCS(ProvisionNIDMgmtType)# showclock commit
    ShowClock_Output.showClockResp.clock_info = 'System Time   : 1970-01-02T19:17:07+05:30'


    ShowClock Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Clearing IP ARP Entries

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**clearIpArpEntries clearIpArpEntriesReq** {**all**} **Example:** `UCS(ProvisionNIDMgmtType)# clearIpArpEntries clearIpArpEntriesReq all`	Clears the IP ARP entries.
Step 4	**clearIpArpEntries review** **Example:** `UCS(ProvisionNIDMgmtType)# clearIpArpEntries review`	Displays the configuration.
Step 5	**clearIpArpEntries commit** **Example:** `UCS(ProvisionNIDMgmtType)# clearIpArpEntries commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# clearIpArpEntries clearIpArpEntriesReq all
UCS(ProvisionNIDMgmtType)# clearIpArpEntries review

Commands in queue:
    clearIpArpEntries clearIpArpEntriesReq all

UCS(ProvisionNIDMgmtType)# clearIpArpEntries commit

    ClearIpArpEntries_Output.clearIpArpEntriesResp = 0

    ClearIpArpEntries Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Verifying IP ARP Entries

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**showIpArp showIpArpEntriesReq** {**all**} **Example:** `UCS(ProvisionNIDMgmtType)# showIpArp showIpArpEntriesReq all`	Displays the IP ARP details.
Step 4	**showIpArp review** **Example:** `UCS(ProvisionNIDMgmtType)# showIpArp review`	Displays the configuration.
Step 5	**showIpArp commit** **Example:** `UCS(ProvisionNIDMgmtType)# showIpArp commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# showIpArp showIpArpEntriesReq all
UCS(ProvisionNIDMgmtType)# showIpArp review

Commands in queue:
    showIpArpEntriesReq all

UCS(ProvisionNIDMgmtType)# showIpArp commit

    ShowIpArp_Output.showIpArpEntriesResp.arp_entry[0] = '10.0.0.1 via
VLAN10:00-00-0c-07-ac-03'
    ShowIpArp_Output.showIpArpEntriesResp.arp_entry[1] = '10.0.10.21 via
VLAN10:e9-ed-f3-78-27-c0'

    ShowIpArp Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Configuring IP Route Global Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**ipRoutingGlobalConfig ipRoutingGlobalConfigReq** {**disable** | **enable**} **Example:** `UCS(ProvisionNIDMgmtType)# ipRoutingGlobalConfig ipRoutingGlobalConfigReq enable`	Configures the IP routing global configuration. • **disable**—Disables the IP Routing. • **enable**—Enables the IP Routing.
Step 4	**ipRoutingGlobalConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# ipRoutingGlobalConfig review`	Displays the configuration.
Step 5	**ipRoutingGlobalConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# ipRoutingGlobalConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmt Type mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# ipRoutingGlobalConfig ipRoutingGlobalConfigReq enable
UCS(ProvisionNIDMgmtType)# ipRoutingGlobalConfig review

Commands in queue:
    ipRoutingGlobalConfig ipRoutingGlobalConfigReq enable

UCS(ProvisionNIDMgmtType)# ipRoutingGlobalConfig commit

    IpRoutingGlobalConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Configuring IP Route

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.

Step 3	**setiproute setIpRouteReq** {**gateway_ip** *WORD* |**ipv4_address** *WORD* | **ipv4_mask** *WORD*}
	
	Example:
	`UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_address 10.0.144.0`
	`UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_mask 255.255.255.0`
	`UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq gateway_ip 10.0.0.1`
		Configures the IP Route.
		• **gateway_ip—**Specifies the gateway IPv4 address.
		◦ *WORD*—IPv4 address.
		• **ipv4_address—**Specifies the IPv4 Network/Address.
		◦ *WORD*—IPv4 Network/Address.
		• **ipv4_mask—**Specifies the IPv4 mask.
		◦ *WORD*—IPv4 mask.

Step 4	**setiproute review** **Example:** `UCS(ProvisionNIDMgmtType)# setiproute review`	Displays the configuration.
Step 5	**getClockConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# setiproute commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_address 10.0.144.0
```

```
UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq ipv4_mask 255.255.255.0
UCS(ProvisionNIDMgmtType)# setIpRoute setIpRouteReq gateway_ip 10.0.0.1

UCS(ProvisionNIDMgmtType)# setiproute review
Commands in Queue:
   setIpRoute setIpRouteReq ipv4_address 10.0.144.0
   setIpRoute setIpRouteReq ipv4_mask 255.255.255.0
   setIpRoute setIpRouteReq gateway_ip 10.0.0.1

UCS(ProvisionNIDMgmtType)# setiproute commit

   Setiproute Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Viewing IP Route

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**showiproute showIpRouteReq** {**all**} **Example:** `UCS(ProvisionNIDMgmtType)# showiproute showIpRouteReq all`	Displays the IP route details. • **all**—Specifies the IP route entries.
Step 4	**showiproute review** **Example:** `UCS(ProvisionNIDMgmtType)# showiproute review`	Displays the configuration.
Step 5	**showiproute commit** **Example:** `UCS(ProvisionNIDMgmtType)# showiproute commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# showiproute showIpRouteReq all
```

```
UCS(ProvisionNIDMgmtType)# showiproute review

Commands in queue:
    showIpRoute showIpRouteReq all

UCS(ProvisionNIDMgmtType)# showiproute commit

    ShowIpRoute_Output.showIpRouteResp.ip_route_entry[0] = '0.0.0.0/0 via 10.25.0.1 [UP
GATEWAY HW_RT]'
    ShowIpRoute_Output.showIpRouteResp.ip_route_entry[1] = '10.25.0.0/16 via  [UP HW_RT]'
    ShowIpRoute_Output.showIpRouteResp.ip_route_entry[2] = '127.0.0.1/32 via 127.0.0.1 [UP
 HOST]'
   ShowIpRoute_Output.showIpRouteResp.ip_route_entry[3] = '202.153.0.0/16 via 7.25.0.1 [UP
 GATEWAY HW_RT]'
   ShowIpRoute_Output.showIpRouteResp.ip_route_entry[4] = '224.0.0.0/4 via 127.0.0.1 [UP]'


    ShowIpRoute Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Removing IP Route

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.

Step 3	**removeiproute removeIpRouteReq** {**gateway_ip** *WORD* |**ipv4_address** *WORD* |**ipv4_mask** *WORD*}
	
	Example:
	`UCS(ProvisionNIDMgmtType)# removeiproute removeIpRouteReq ipv4_address 10.0.144.0`
	`UCS(ProvisionNIDMgmtType)# removeiproute removeIpRouteReq ipv4_mask 255.255.255.0`
	`UCS(ProvisionNIDMgmtType)# removeiproute removeIpRouteReq gateway_ip 10.0.0.1`
		Removes the IP Route.
		• **gateway_ip—**Specifies the gateway IPv4 address.
		  ◦ *WORD*—IPv4 address.
		• **ipv4_address—**Specifies the IPv4 Network/Address.
		  ◦ *WORD*—IPv4 Network/Address.
		• **ipv4_mask—**Specifies the IPv4 mask.
		  ◦ *WORD*—IPv4 mask.

Step 4	**removeIpRoute review** **Example:** `UCS(ProvisionNIDMgmtType)# removeIpRoute review`	Displays the configuration.

	Command or Action	Purpose
Step 5	**removeIpRoute commit** **Example:** `UCS(ProvisionNIDMgmtType)# removeIpRoute commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmt Type mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# removeiproute removeIpRouteReq ipv4_address 10.0.144.0
UCS(ProvisionNIDMgmtType)# removeiproute removeIpRouteReq ipv4_mask 255.255.255.0
UCS(ProvisionNIDMgmtType)# removeiproute removeIpRouteReq gateway_ip 10.0.0.1

UCS(ProvisionNIDMgmtType)#removeIpRoute review

Commands in queue:
    removeiproute removeIpRouteReq ipv4_address 10.0.144.0
    removeiproute removeIpRouteReq ipv4_mask 255.255.255.0
    removeiproute removeIpRouteReq gateway_ip 10.0.0.1

UCS(ProvisionNIDMgmtType)# removeIpRoute commit

    Removeiproute Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Configuring IP DNS Proxy Request

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**setipdnsProxyConfig setIpDNSProxyConfigReq** {**dns_proxy**} **Example:** `UCS(ProvisionNIDMgmtType)# setipdnsProxyConfig setIpDNSProxyConfigReqdns_proxy`	Configures the IP DNS proxy request. • **dns_proxy—**Configures the DNS proxy service.

	Command or Action	Purpose
Step 4	**setipdnsProxyConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# setipdnsProxyConfig review`	Displays the configuration.
Step 5	**setipdnsProxyConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# setipdnsProxyConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# setipdnsProxyConfig setIpDNSProxyConfigReq dns_proxy
UCS(ProvisionNIDMgmtType)# setipdnsProxyConfig review

Commands in queue:
    setIpDnsProxyConfig setIpDNSProxyConfigReq dns_proxy

UCS(ProvisionNIDMgmtType)# setipdnsProxyConfig commit

    SetIpDnsProxyConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Removing IP DNS Proxy Request Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**removeipdnsProxyConfig removeIpDnsProxyConfigReq** {**dns_proxy**}	Removes the IP DNS proxy configuration.

	Command or Action	Purpose
	Example: `UCS(ProvisionNIDMgmtType)# removeipdnsProxyConfig removeIpDnsProxyConfigReq dns_proxy`	
Step 4	**removeipdnsProxyConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# removeipdnsProxyConfig review`	Displays the configuration.
Step 5	**removeipdnsProxyConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# removeipdnsProxyConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits to the ProvisionNIDMgmt Type mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# removeipdnsProxyConfig removeIpDnsProxyConfigReq dns_proxy
UCS(ProvisionNIDMgmtType)# removeipdnsProxyConfig review

Commands in queue:
   removeIpDnsProxyConfig removeIpDnsProxyConfigReq dns_proxy

UCS(ProvisionNIDMgmtType)# removeipdnsProxyConfig commit

   RemoveIpDnsProxyConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Configuring the Name Server

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.

	Command or Action	Purpose
Step 3	**setnameServerConfig setNameServerConfigReq** {**dhcp** {**vlan_interface***vlan_id*} |**ipv4_address** *WORD*} **Example:** `UCS(ProvisionNIDMgmtType)# setNameServerConfig setNameServerConfigReq ipv4_address 10.0.0.5`	Configures the name server. • **dhcp—**Specifies the Dynamic Host Configuration Protocol. ◦ **vlan_interface—**Select an VLAN interface to configure. ◦ *vlan_id***—**Vlan ID. The range is from 1 to 4093. • **ipv4_address—**Specifies IPv4 unicast address. ◦ *WORD* **—**IPv4 unicast address.
Step 4	**setnameServerConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# setnameServerConfig review`	Displays the configuration.
Step 5	**setnameServerConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# setnameServerConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# setNameServerConfig setNameServerConfigReq ipv4_address 10.0.0.5
UCS(ProvisionNIDMgmtType)# setNameServerConfig review

Commands in queue:
    setNameServerConfig setNameServerConfigReq ipv4_address 10.0.0.5

UCS(ProvisionNIDMgmtType)# setNameServerConfig commit

    nid_create_SetNameServerConfig_req_file 7421

    SetNameServerConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Verifying the Name Server

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**shownameServerConfig showNameSeverConfigReq** {**config**} **Example:** `UCS(ProvisionNIDMgmtType)# shownameServerConfig showNameSeverConfigReq config`	Displays the name server details. • **config—**Displays the name server configuration.
Step 4	**shownameServerConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# shownameServerConfig review`	Displays the configuration.
Step 5	**shownameServerConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# shownameServerConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmtType mode.

Configuration Example

```
UCS(ProvisionNIDMgmtType)# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# shownameServerConfig showNameSeverConfigReq config
UCS(ProvisionNIDMgmtType)# shownameServerConfig review

Commands in queue:
   shownameServerConfig showNameSeverConfigReq config

UCS(ProvisionNIDMgmtType)# shownameServerConfig commit
ShowNameServerConfig_Output.showNameServerConfigResp.name_server_config = 'Current DNS
server is 7.0.0.3 set by STATIC.'

  ShowNameServerConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Removing the Name Server

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNIDMgntType** **Example:** `UCS# ProvisionNIDMgntType`	Enters the ProvisionNIDMgntType mode.
Step 3	**removenameServerConfig removeNameServerConfigReq** {**name_server**} **Example:** `UCS(ProvisionNIDMgmtType)# removenameServerConfig removeNameServerConfigReq name_server`	Removes the name server. • **name_server—**Specifies the domain name system removal.
Step 4	**removenameServerConfig review** **Example:** `UCS(ProvisionNIDMgmtType)# removenameServerConfig review`	Displays the configuration.
Step 5	**getClockConfig commit** **Example:** `UCS(ProvisionNIDMgmtType)# removenameServerConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNIDMgmtType)# exit`	Exits the ProvisionNIDMgmt Type mode.

Configuration Example

```
UCS# ProvisionNIDMgmtType
UCS(ProvisionNIDMgmtType)# removenameServerConfig removeNameServerConfigReq name_server
UCS(ProvisionNIDMgmtType)# removenameServerConfig review

Commands in queue:
    removenameServerConfig removeNameServerConfigReq name_server

UCS(ProvisionNIDMgmtType)# removenameServerConfig commit

    RemoveNameServerConfig Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

CHAPTER 4

Configuring Notifications

This chapter describes how to configure notifications on Cisco ME 1200 NID.

Prerequisites for Configuring Notifications

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.
- Configuring username and enabling password on the UCS controller is a must for HTTP authentication.

Restrictions for Configuring Notifications

- Only the first three notification lists are supported. They are:
    - Notifications_list 1 for HeartBeat notifications.
    - Notifications_list 2 for Config Change Trap notifications.
    - Notifications_list 3 for Asynchronous failure notifications.

Note In case, the other notification lists are configured, the configuration does not get applied.

Information About Notifications

Notifications are asynchronous messages generated by NID based on the events. The notifications are transported over HTTP POST request. The NID acts as an HTTP client and the controller (ME3600 or any other HTTP server) acts as a server.

Types of Notifications

The Cisco ME 1200 NID supports three types of notifications:

- HeartBeat Notifications
- Config Change Trap Notifications
- Asynchronous Failure Notifications

HeartBeat Notifications

Heartbeat notifications are used to check the liveliness of the NID. This corresponds to the module_id = 1, and the notification_id = 1. You can use the **setNotificationReceiver setNotificationReceiver_req heartbeat 30** command to configure the HeartBeat to 30 seconds. If the controller does not receive the heartbeat notification within 30 seconds, the controller waits for an additional 60 seconds before announcing Cisco ME 1200 NID as DOWN. For more information, see .

Config Change Trap Notifications

The registered server receives configuration change notifications when any other controller does prime changes or accesses the NID through XML. This corresponds to the module_id = 2, and the notification_id = 1. For more information, see .

Asynchronous Failure Notifications (ICLI failures)

The controller receives asynchronous failure notifications, when there are ICLI failures. Every commit action (SOAP action) on the controller is translated into multiple ICLI commands on NID, and if there are failures during the ICLI command execution, an asynchronous failure notification is sent to the controller that has registered to receive the asynchronous failure notification. This corresponds to the module_id = 3 , and the notification_id = 1. For more information, see .

How to Configure Notifications

Setting Password on the Controller

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**username** *user_name* **Example:** `UCS# username guest123`	Configures the username. • *user_name*—username.
Step 3	**enable password** *password* **Example:** `UCS# enable password lab123`	Configures the password. • *password*—password.
Step 4	**exit** **Example:** `UCS# exit`	Exits to the privileged EXEC mode.

Configuration Example

```
UCS# configureNID
UCS# username guest123
UCS# enable password lab123
UCS# exit
```

Viewing Notifications

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionNotifications** **Example:** `UCS# ProvisionNotifications`	Enters the Notifications mode.
Step 3	**getNotificationList listAllNotifications_req** **Example:** `UCS(ProvisionNotifications)# getNotificationList listAllNotifications_req`	Displays the supported notifications list for that module.
Step 4	**getNotificationList review** **Example:** `UCS(ProvisionNotifications)# getNotificationList review`	Displays the configuration.
Step 5	**getNotificationList commit** **Example:** `UCS(ProvisionNotifications)# getNotificationList commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNotifications)# exit`	Exits the Notifications mode.

Configuration Example

```
UCS# ProvisionNotifications
UCS(ProvisionNotifications)# getNotificationList  listAllNotifications_req
UCS(ProvisionNotifications)# getNotificationList review

Commands in queue:
    getNotificationList listAllNotifications_req

UCS(ProvisionNotifications)# getNotificationList commit

    GetNotificationList_Output.listAllNotifications_resp.notification[0].module_id = 1
   GetNotificationList_Output.listAllNotifications_resp.notification[0].module_description
 = 'HeartBeat'
    GetNotificationList_Output.listAllNotifications_resp.notification[0].notification_id =
 1

GetNotificationList_Output.listAllNotifications_resp.notification[0].notification_description
 = 'Heartbeat Notifications'
    GetNotificationList_Output.listAllNotifications_resp.notification[1].module_id = 2
   GetNotificationList_Output.listAllNotifications_resp.notification[1].module_description
 = 'ConfigChangeTrap'
    GetNotificationList_Output.listAllNotifications_resp.notification[1].notification_id =
 1

GetNotificationList_Output.listAllNotifications_resp.notification[1].notification_description
 = 'Config Change Notifications'
    GetNotificationList_Output.listAllNotifications_resp.notification[2].module_id = 3
   GetNotificationList_Output.listAllNotifications_resp.notification[2].module_description
 = 'AsyncNotification'
    GetNotificationList_Output.listAllNotifications_resp.notification[2].notification_id =
 1
```

```
GetNotificationList_Output.listAllNotifications_resp.notification[2].notification_description
 = 'ICLI command failure'

    GetNotificationList Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Configuring Notifications

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNotifications** **Example:** `UCS# ProvisionNotifications`	Enters the Notifications mode.

Step 3	**setNotificationReceiver setNotificationReceiver_req** {**heartbeat** *heartbeat* |**http_BA_password** *WORD* | **http_BA_user** *WORD* | **http_file_path** *WORD* | **http_server_address** *WORD*}
	
	Example:
	`UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req heartbeat 30`
	`UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req http_BA_password lab123`
	
	`UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req http_BA_user guest123`
	`UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req http_file_path myfiles/nid_notification`
	`UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req http_server_address 10.20.30.40`
		Configures the values for receiving the notifications.
		• **heartbeat—**Specifies the rate at which keepalive packets are expected (in seconds).
		• *heartbeat*—Heartbeat rate. The range is from 1 to 65535.
		• **http_BA_password—**Specifies the HTTP basic authentication password.
		• *WORD*—Password.
		• **http_BA_user—**Specifies the HTTP basic authentication user.
		• *WORD*—user.
		• **http_file_path—**Specifies the name of a document/resource. For example: /nid_notification.
		• *WORD*—name of a document.
		• **http_server_address—**Specifies the HTTP server IP address.
		• *WORD*—HTTP Server IP address.

	Command or Action	Purpose
Step 4	**setNotificationReceiver review** **Example:** `UCS(ProvisionNotifications)# setNotificationReceiver review`	Displays the configuration.
Step 5	**setNotificationReceiver commit** **Example:** `UCS(ProvisionNotifications)# setNotificationReceiver commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNotifications)# exit`	Exits the Notifications mode.

Configuration Example

```
UCS# ProvisionNotifications
UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req heartbeat
 30
UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req
http_BA_password lab123
UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req http_BA_user
 guest123
UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req
http_file_path myfiles/nid_notification
UCS(ProvisionNotifications)# setNotificationReceiver setNotificationReceiver_req
http_server_address 10.20.30.40
UCS(ProvisionNotifications)# setNotificationReceiver review

Commands in queue:
    setNotificationReceiver setNotificationReceiver_req heartbeat 30
    setNotificationReceiver setNotificationReceiver_req http_BA_password lab123
    setNotificationReceiver setNotificationReceiver_req http_BA_user guest123
    setNotificationReceiver setNotificationReceiver_req http_file_path
myfiles/nid_notification
    setNotificationReceiver setNotificationReceiver_req http_server_address 10.20.30.40

UCS(ProvisionNotifications)# setNotificationReceiver commit

    SetNotificationReceiver_Output.setNotificationReceiver_resp.registerOK = true
    SetNotificationReceiver_Output.setNotificationReceiver_resp.cookie = 5120

    SetNotificationReceiver Commit Success!!!!

UCS(ProvisionNIDMgmtType)# exit
```

Note Use this procedure to generate multiple cookies. In the above mentioned example, 5120 is the cookie value.

Registering for HeartBeat Notification

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNotifications** **Example:** `UCS# ProvisionNotifications`	Enters the Notifications mode.

Step 3	**registerForNotification regForNotification_req** {**cookie** *cookie value* | **notifications_list** *List of notifications* {**enable** {**enable** | **disable**} | **module_id** *Module ID* | **notification_id** *Notification ID* }}
	
	Example:
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req cookie 5120`
	
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 1 enable enable`
	
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 1 module_id 1`
	
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 1 notification_id 1`
	Note For the Heartbeat notification, the notification_list is 1, the module_id is 1, and the notification_id is 1.
		Lists all the registered notifications under the generated cookie.
		
		• **cookie**—Specifies the notification cookie with unique cookie value.
		• *cookie value*—cookie value. The range is from 1 to 65535.
		• **notifications_list**—Specifies the list of notifications.
		• *List of notifications*—list of notification ranges from 1 to 10, where, the supported lists are from 1 to 3.
		• **enable**—Enables or disables the notification.
		• **module_id**—Specifies the module ID from which notifications are desired.
		• *Module ID*—Module ID. The range is from 1 to 65536.
		• **notification_id**—Specifies the notification ID.
		• *Notification ID*—list of notification ranges from 1 to 10, wherein the supported lists are from 1 to 3.

Step 4	**registerForNotification review** **Example:** `UCS(ProvisionNotifications)# registerForNotification review`	Displays the configuration.
Step 5	**registerForNotification commit** **Example:** `UCS(ProvisionNotifications)# registerForNotification commit`	Sends the configuration to the NID.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(ProvisionNotifications)# exit`	Exits the Notification mode.

Configuration Example

```
UCS# ProvisionNotifications
UCS(ProvisionNotifications)# registerForNotification regForNotification_req cookie 5120
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 1 enable enable
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 1 module_id 1
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 1 notification_id 1
UCS(ProvisionNotifications)# registerForNotification review

Commands in queue:
    registerForNotification regForNotification_req cookie 5120
    registerForNotification regForNotification_req notifications_list 1 enable enable
    registerForNotification regForNotification_req notifications_list 1 module_id 1
    registerForNotification regForNotification_req notifications_list 1 notification_id 1

UCS(ProvisionNotifications)# registerForNotification commit

    RegisterForNotification_Output.regForNotification_resp = 0
    RegisterForNotification Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Registering for Config Change Trap Notification

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNotifications** **Example:** `UCS# ProvisionNotifications`	Enters the Notifications mode.
Step 3	**registerForNotification regForNotification_req** {**cookie** *cookie value* | **notifications_list** *List of notifications* {**enable** {**enable** | **disable**} | **module_id** *Module ID* | **notification_id** *Notification ID* }}	Lists all the registered notifications under the generated cookie. • **cookie—**Specifies the notification cookie with unique cookie value.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req cookie 5120`
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 2 enable enable`
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 2 module_id 2`
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 2 notification_id 1`
	Note For the ConfigChangeTrap notification, the notification_list is 2, the module_id is 2, and the notification_id is 1.
		• *cookie value*—cookie value. The range is from 1 to 65535.
		• **notifications_list—**Specifies the list of notifications.
		• *List of notifications*—list of notification ranges from 1 to 10, where, the supported lists are from 1 to 3.
		• **enable—**Enables or disables the notification.
		• **module_id—**Specifies the module ID from which notifications are desired.
		• *Module ID*—Module ID. The range is from 1 to 65536.
		• **notification_id—**Specifies the notification ID.
		• *Notification ID*—list of notification ranges from 1 to 10, wherein the supported lists are from 1 to 3.

Step 4	**registerForNotification review** **Example:** `UCS(ProvisionNotifications)# registerForNotification review`	Displays the configuration.
Step 5	**registerForNotification commit** **Example:** `UCS(ProvisionNotifications)# registerForNotification commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNotifications)# exit`	Exits the Notifications mode.

Configuration Example

```
UCS# ProvisionNotifications
UCS(ProvisionNotifications)# registerForNotification regForNotification_req cookie 5120
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 2 enable enable
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 2 module_id 2
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 2 notification_id 1
UCS(ProvisionNotifications)# registerForNotification review

Commands in queue:
    registerForNotification regForNotification_req cookie 5120
    registerForNotification regForNotification_req notifications_list 2 enable enable
    registerForNotification regForNotification_req notifications_list 2 module_id 2
    registerForNotification regForNotification_req notifications_list 2 notification_id 1
```

```
UCS(ProvisionNotifications)# registerForNotification commit

    RegisterForNotification_Output.regForNotification_resp = 0
    RegisterForNotification Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Registering for Asynchronous Failure Notification

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNotifications** **Example:** `UCS# ProvisionNotifications`	Enters the Notifications mode.

Step 3	**registerForNotification regForNotification_req** {**cookie** *cookie value* | **notifications_list** *List of notifications* {**enable** {**enable** | **disable**} | **module_id** *Module ID* | **notification_id** *Notification ID* }}
	
	Example:
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req cookie 5120`
	
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 3 enable enable`
	
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 3 module_id 3`
	
	`UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list 3 notification_id 1`
	Note For the Asynchronous failure notification, the notification_list is 3, the module_id is 3, and the notification_id is 1.
		Lists all the registered notifications under the generated cookie.
		
		• **cookie**—Specifies the notification cookie with unique cookie value.
		• *cookie value*—cookie value. The range is from 1 to 65535.
		• **notifications_list**—Specifies the list of notifications.
		• *List of notifications*—list of notification ranges from 1 to 10, where, the supported lists are from 1 to 3.
		• **enable**—Enables or disables the notification.
		• **module_id**—Specifies the module ID from which notifications are desired.
		• *Module ID*—Module ID. The range is from 1 to 65536.
		• **notification_id**—Specifies the notification ID.
		• *Notification ID*—list of notification ranges from 1 to 10, wherein the supported lists are from 1 to 3.

	Command or Action	Purpose
Step 4	**registerForNotification review** **Example:** `UCS(ProvisionNotifications)# registerForNotification review`	Displays the configuration.
Step 5	**registerForNotification commit** **Example:** `UCS(ProvisionNotifications)# registerForNotification commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNotifications)# exit`	Exits the Notifications mode.

Configuration Example

```
UCS# ProvisionNotifications
UCS(ProvisionNotifications)# registerForNotification regForNotification_req cookie 5120
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 3 enable enable
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 3 module_id 3
UCS(ProvisionNotifications)# registerForNotification regForNotification_req notifications_list
 3 notification_id 1
UCS(ProvisionNotifications)# registerForNotification review

Commands in queue:
    registerForNotification regForNotification_req cookie 5120
    registerForNotification regForNotification_req notifications_list 3 enable enable
    registerForNotification regForNotification_req notifications_list 3 module_id 3
    registerForNotification regForNotification_req notifications_list 3 notification_id 1

UCS(ProvisionNotifications)# registerForNotification commit

    RegisterForNotification_Output.regForNotification_resp = 0
    RegisterForNotification Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Listing Notifications

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionNotifications** **Example:** `UCS# ProvisionNotifications`	Enters the Notifications mode.
Step 3	**listRegisteredNotification listNotificationsRegistered** {**cookie** *cookie value*} **Example:** `UCS(ProvisionNotifications)# listRegisteredNotification listNotificationsRegistered cookie 5120`	Lists all the registered notifications under the generated cookie. • **cookie**—Specifies the notification cookie with unique cookie value. • *cookie value*—cookie value. The range is from 1 to 65535.
Step 4	**listRegisteredNotification review** **Example:** `UCS(ProvisionNotifications)# listRegisteredNotification review`	Displays the configuration.
Step 5	**listRegisteredNotification commit** **Example:** `UCS(ProvisionNotifications)# listRegisteredNotification commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNotifications)# exit`	Exits the Notifications mode.

Configuration Example

```
UCS# ProvisionNotifications
UCS(ProvisionNotifications)# listRegisteredNotification listNotificationsRegistered cookie
 5120
UCS(ProvisionNotifications)# listRegisteredNotification review

Commands in queue:
    listRegisteredNotification listNotificationsRegistered cookie 5120

UCS(ProvisionNotifications)# listRegisteredNotification commit

   ListRegisteredNotification_Output.regForNotification_req.notifications_list[0].module_id
 = 1

ListRegisteredNotification_Output.regForNotification_req.notifications_list[0].notification_id
 = 1
    ListRegisteredNotification_Output.regForNotification_req.notifications_list[0].enable
= true
   ListRegisteredNotification_Output.regForNotification_req.notifications_list[1].module_id
 = 2

ListRegisteredNotification_Output.regForNotification_req.notifications_list[1].notification_id
 = 1
    ListRegisteredNotification_Output.regForNotification_req.notifications_list[1].enable
= true
```

```
    ListRegisteredNotification_Output.regForNotification_req.notifications_list[2].module_id
 = 3

ListRegisteredNotification_Output.regForNotification_req.notifications_list[2].notification_id
 = 1
    ListRegisteredNotification_Output.regForNotification_req.notifications_list[2].enable
= true
    ListRegisteredNotification_Output.regForNotification_req.cookie = 5120

    ListRegisteredNotification Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

Deleting Notifications

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionNotifications** **Example:** `UCS# ProvisionNotifications`	Enters the Notifications mode.
Step 3	**delNotificationReceiver delNotifReceiver_req** {**cookie** *cookie value*} **Example:** `UCS(ProvisionNotifications)# delNotificationReceiver delNotifReceiver_req cookie 5120`	Deletes the notifications under the cookie. • **cookie**—Specifies the notification cookie with unique cookie value. • *cookie value*—cookie value.
Step 4	**delNotificationReceiver review** **Example:** `UCS(ProvisionNotifications)# delNotificationReceiver review`	Displays the configuration.
Step 5	**delNotificationReceiver commit** **Example:** `UCS(ProvisionNotifications)# delNotificationReceiver commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionNotifications)# exit`	Exits the Notifications mode.

Configuration Example

```
UCS# ProvisionNotifications
UCS(ProvisionNotifications)# delNotificationReceiver delNotifReceiver_req cookie 5120
UCS(ProvisionNotifications)# delNotificationReceiver review

Commands in queue:
    delNotificationReceiver delNotifReceiver_req cookie 5120

UCS(ProvisionNotifications)# delNotificationReceiver commit

    DelNotificationReceiver_Output.delNotifReceiver_resp = false

    DelNotificationReceiver Commit Success!!!

UCS(ProvisionNIDMgmtType)# exit
```

What to Do Next

After deleting the notification, use the **listRegisteredNotification listNotificationsRegistered** {**cookie** *cookie value*} command to verify if the delete operation is successful.

```
UCS(ProvisionNotifications)# listRegisteredNotification listNotificationsRegistered cookie
 cookie value
UCS(ProvisionNotifications)# listRegisteredNotification review
UCS(ProvisionNotifications)# listRegisteredNotification commit
```

CHAPTER 5

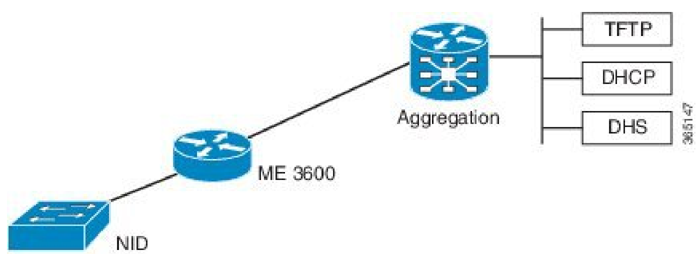
Zero Touch Provisioning

Zero Touch Provisioning (ZTP) automates configuration of Cisco ME 1200 Series Carrier Ethernet Access Device (hereafter known as Cisco ME 1200 NID) when it is deployed either in standalone operating mode or through a directly connected upstream user premise equipment such as Cisco ME 3600/ME 3800x devices or the Cisco ASR920. When connected through ME 3600x device, provisioning of CE Services can be done from the remote controller mode. Otherwise, CLI on Cisco ME 1200 NID can be accessed using SSH to provision CE Services once ZTP process is completed .

The ZTP process is activated by pressing the ZTP pinhole reset button found on the front of the Cisco ME 1200 NID. This minimizes manual operator intervention and helps reduce customers' initial deployment costs.

Note The only interface for the ZTP is the ZTP button and the status LED, both found on the front of the Cisco ME 1200 NID.

Figure 1: Cisco ME 1200 Topology-Typical Deployment



Restrictions for ZTP

• ZTP inherits the security levels of the protocols it uses. Therefore, ZTP must be used in a trusted environment, where all security concerns are handled by protocols or technologies it uses.

- ZTP is not supported over IPv6.

ZTP Activation

Pressing the ZTP reset button triggers a series of steps that result in provisioning the Cisco ME 1200 NID with a complete, operational configuration.

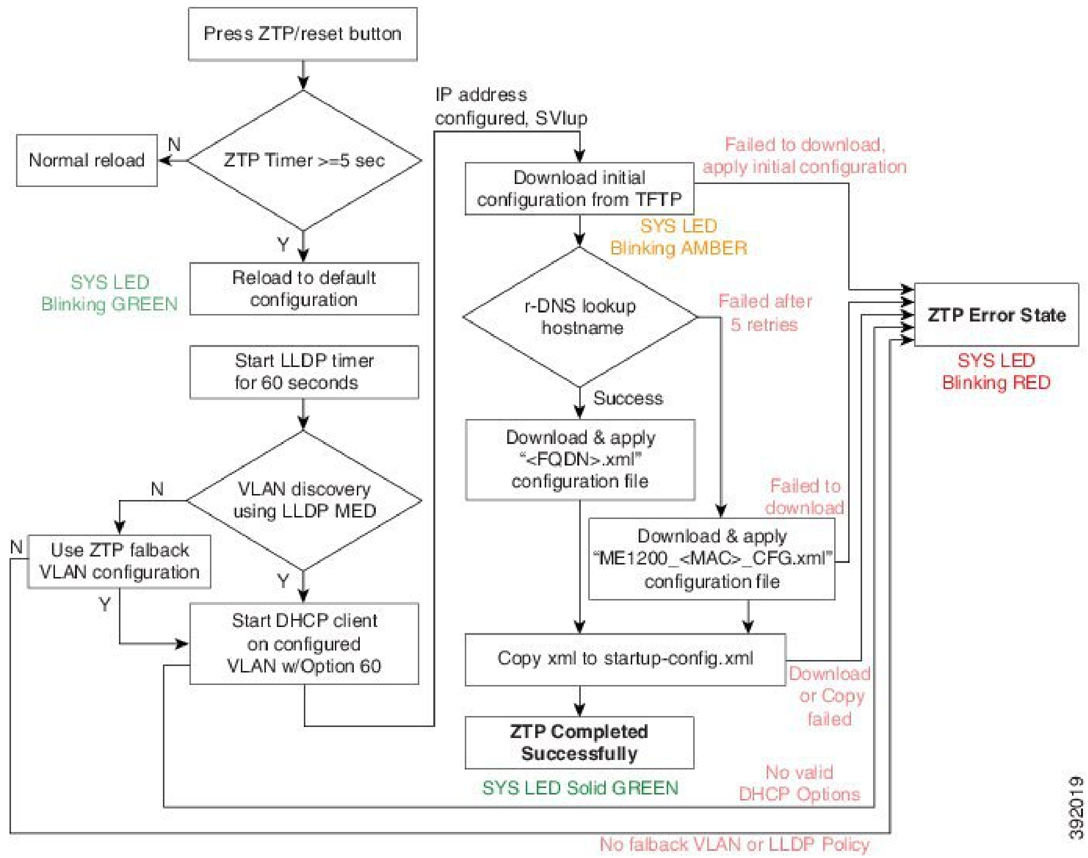
1. Start ZTP.
2. Restore to factory defaults.
3. Get management VLAN Configuration from LLDP-MED.
4. Start the DHCP client on VLAN interface.
5. Download and apply the initial configuration file from a location provided by the DHCP client.
6. Use reverse DNS to obtain the host name of the device.
7. Download and apply a specific configuration file.

Note Ensure that the initial and specific configuration files are present on the TFTP server before starting the ZTP process.

8. Copy the running configuration to startup configuration.

The following figure depicts the process pictorially:

Figure 2: ZTP Activation Process



This process is explained in detail in the following sections.

ME 3600x device series Configuration

On the Cisco ME 3600x Series Ethernet Access Switch, following DHCP server, management VLAN configuration is required:

- Configure the DHCP pool and add the TFTP and DNS configurations through DHCP options. The following is a sample DHCP pool:

```
ip dhcp excluded-address 7.6.0.1 7.6.19.51
ip dhcp excluded-address 7.6.19.64 7.6.255.255
ip dhcp pool ztp-test
 network 7.6.0.0 255.255.255.0
 default-router 7.6.0.10                    -> Adding local SVI IP as default gateway
 for Cisco ME 1200 NID
 option 60 ascii ME1200-00-3A-99-FD-45-34     -> Adding Cisco ME 1200 NID MAC
 option 43 ip 7.0.0.221                       -> Adding TFTP server
 option 67 ascii "ME1200_CFG"                 -> Adding initial configuration-file
 dns-server 7.0.0.217
```

- Enable LLDP-MED network Policy TLV on the ME 3600x device.

```
Switch(config)# interface gigabit 0/1          -> Physical port on UPE connected to
```

```
Cisco ME 1200 NID
Switch(config-if)# lldp med-tlv-select network-policy
Switch(config-if)# lldp transmit
Switch(config-if)# lldp receive
Switch(config-if)# exit
```

- Run LLDP.

```
Switch(config-if)# lldp run
```

- Configure an interface connecting to the 7.6.x.x switch and allowing VLAN (e.g. VLAN 10) to be used as management VLAN to ensure reachability to network gateway, or DHCP, TFTP, and DNS servers.
- Configure the interface connecting to Cisco ME 1200 NID as trunk.
- Assign management VLAN with the interface connecting to Cisco ME 1200 NID using the following command in global config mode

```
Switch(config)# platform nid-controller assign vlan 10 gigabitEthernet 1/1
```

Step 1—Start ZTP

The ZTP activation is started by pressing the ZTP reset button for five seconds or more.

Note If the ZTP reset button is pressed for less than five seconds, a cold reload is issued.

Note When one instance of the ZTP activation is active, another instance cannot be started. Wait for the ZTP activation to complete (check ZTP status using LED to determine if it was successful or moved to error state) before starting the process again.

Step 2—Reload Defaults

When ZTP activation is triggered, it causes the Cisco ME 1200 NID to reload with a default configuration that includes LLDP MED endpoint connectivity mode on all ports.

The Cisco ME 1200 NID may have links on several ports when the ZTP activation is started. All such ports are candidates for being used for the ZTP activation, and must be ready to receive the LLDP-MED TLV, where a packet is sent on a point-to-point link with a well-known multicast destination MAC.

The LLDP-MED information is sent every 30 seconds. To ensure that all ports receive the LLDP-MED TLV, a 60-second timer starts the count down for LLDP initialization on all ports.

Step 3—Get Management VLAN Configuration

VLAN Discovery using remote ME 3600x device

The first step after ZTP reload is the discovery of a management VLAN between the User-Facing Premise Equipment such as the Cisco ME 3600X Series Ethernet Access Switch, and one or more Cisco ME 1200 NIDs. In this step, the LLDP-MED is used to acquire the management VLAN configuration.

The Cisco ME 1200 NID can be configured to act either as an end-point or a connectivity link. The default configuration type is an end-point, where all ports are scanned for received LLDP-MED broadcast. However, only those ports that have received a network policy with voice application type are considered. These ports are configured accordingly, and a VLAN interface is created on the defined VLAN.

If If none of the ports received LLDP-MED TLVs after 60 seconds, or interface on ME 3600x is not configured to allow a specific management VLAN, ZTP process tries to configure a fallback VLAN as described below. Hence it is better to check configurations and ensure there is no network connectivity issue while in this step.

VLAN Discovery in standalone operating mode

If Cisco ME 1200 NID is deployed without ME 3600x or directly connected switch upstream, which does not support LLDP-MED Network Policy TLV, a fallback mechanism is used to complete VLAN discovery. This assumes that an external DHCP server is present on the network, which can support DHCP Option 60 and 43, and allocate IP address to ME1200 (Note: Option 60 unique identifier string will be of the form "ME1200-00-3A-99-FD-45-34", where 00:3a:99:fd:45:34 is a sample MAC address of the Cisco ME1200.

The fallback VLAN discovery can be described as follows:

1. When ZTP is triggered using the push button on Cisco ME 1200 NID, a special, default configuration is loaded , which includes the following commands:
   ```
   ztp fallback vlan 1-4095 frame-type tagged interface Gi 1/1-6
   ztp fallback vlan 1 frame-type untagged interface Gi 1/1-6
   ```
   This causes DHCP replies coming from an external DHCP server to be processed by Cisco ME1200 NID.
2. If a DHCP OFFER containing a VLAN tag in the range of 1-4095, or DHCP OFFER on VLAN 1 (untagged) is received on any one of the ports 1-6, it is used to determine management VLAN.
3. To avoid a flood of DHCP messages from being intercepted by ME1200, only DHCP replies containing DHCP Option 60 & 43 are intercepted.

The VLAN that is determined from this fallback mechanism is used in subsequent steps of the ZTP process.

Note The default fallback VLAN configuration can also be modified by user and the modified configuration allowed to persist across reloads. From the DHCP pool network range and deployment, user may know *a priori* of the VLAN or range of VLANs from which OFFERs are sent by DHCP server. Hence it is recommended to modify fallback VLAN configuration from default to a reasonable VLAN range. Example 1:

If DHCP server is on VLAN 400 and it's connecting port type is tagged, following configuration change during initial Cisco ME1200 NID configuration will cause DHCP OFFERs on VLAN 400 to be processed when ZTP is triggered in standalone mode.
```
ztp fallback vlan 400 frame-type tagged interface Gi 1/1-6
```
If DHCP server is directly connected to one of the interfaces (say, interface 5), this configuration can be modified further as:
```
ztp fallback vlan 400 frame-type tagged interface Gi 1/5
```

Example 2:

If DHCP server is on VLAN 100 and it's connecting port type is untagged, following configuration change during initial Cisco ME1200 NID configuration causes DHCP OFFERs on VLAN 100 to be processed when ZTP is triggered in standalone mode.
```
ztp fallback vlan 100 frame-type untagged interface Gi 1/1-6
```

Note Only a single VLAN can be configured if port-type is untagged.

If user wishes to disable fallback VLAN configuration, following command can be configured.
```
no ztp fallback vlan
```

Once configuration is modified for above steps, user can copy running-config to startup-config prior to ZTP reset. This will cause modified fallback configuration to be present when ZTP process is restarted.

ZTP for Cisco ME 1200 NIDs in Linear Topology

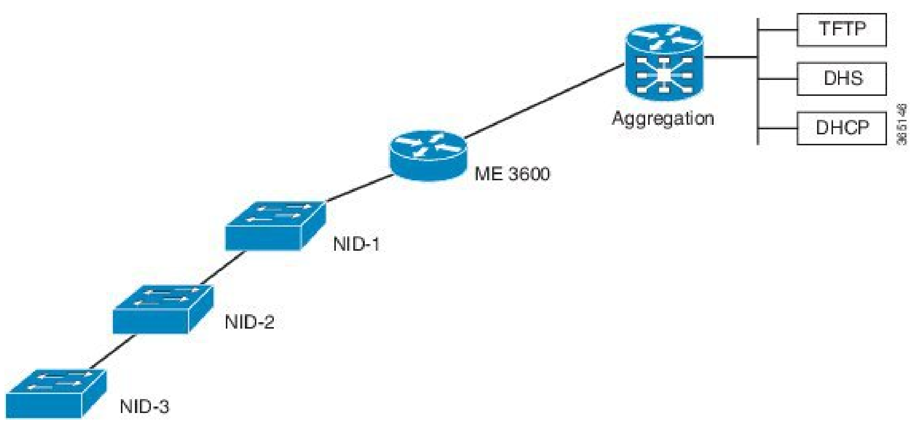
If the network topology involves one or more Cisco ME 1200 NID downstream from the Cisco ME 1200 NID connected to the ME 3600x device, the following steps are required to ensure ZTP works as expected. In this case, each Cisco ME 1200 NID is connected to the upstream Cisco ME 1200 NID through a point-to-point link.

For ZTP to work in this topology, you must trigger ZTP reset on the downstream Cisco ME 1200 NID after ZTP has successfully completed on the upstream Cisco ME 1200 NID.

In addition, before ZTP button is pressed on the downstream Cisco ME 1200 NID, the upstream Cisco ME 1200 NID which has just completed ZTP successfully requires a change in LLDP-MED device type—from endpoint to network connectivity. This single manual step is required to further propagate LLDP towards the downstream Cisco ME 1200 NID.

Consider the following topology:

Figure 3: Cisco ME 1200 NID in a Linear Topology



Configuration on Cisco ME 1200 NID1 Before Starting the ZTP Process on Cisco ME 1200 NID2:

First, on Cisco ME 1200 NID2, configure a LLDP MED media VLAN policy for voice application-type, with frame-type and VLAN set to the same value as the upstream Cisco ME 1200 NID on which ZTP has completed. For example,

```
Switch(config)# lldp med media-vlan-policy 1 voice tagged 10 l2-priority 0 dscp 0
```

Second, on the upstream ME1200 NID1, modify the interface connected to Cisco ME 1200 NID2, in this case GigabitEthernet 1/4, to LLDP MED connectivity type. Also, associate this interface to the same media VLAN policy configured on Cisco ME 1200 NID2. A sample configuration is provided below.

```
interface GigabitEthernet 1/4
 switchport mode trunk
 lldp med media-vlan policy-list 1   -> Assigning media VLAN policy
 lldp med type connectivity          -> Configuring NID1 as network device
 no spanning-tree
 lldp transmit                       -> LLDP transmission is enabled
 lldp receive                        -> LLDP reception is enabled
```

Tip The above configuration can be included in the Cisco ME 1200 NID1 final configuration file to avoid manual configuration after ZTP on Cisco ME 1200 NID1.

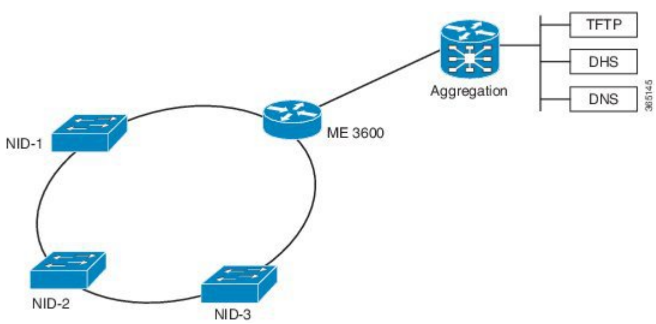
Next Steps

1. Start ZTP on Cisco ME 1200 NID1.
2. When ZTP is finished on Cisco ME 1200 NID-1, start ZTP on Cisco ME 1200 NID2.

ZTP for Cisco ME 1200 NIDs in a Ring Topology

Consider the following topology:

Figure 4: LLDP-MED in a G.8032 Ring Topology



In this deployment, while Cisco ME 1200 NID-1 receives VLAN through LLDP-MED Network Policy TLV and initiates DHCP Discovery, the Cisco ME 1200 NIDs that are downstream to Cisco ME 1200 NID-1 do not receive LLDP-MED TLVs. This is because LLDP-MED TLVs are sent only between endpoint devices and are not propagated beyond Cisco ME 1200 NID-1.

To allow LLDP-MED TLVs to be propagated to all downstream Cisco ME 1200 NIDs once Cisco ME 1200 NID-1 completes VLAN discovery, modify the port configuration in the same manner as the linear chain topology deployment.

Step 4—Start the DHCP Client on the VLAN Interface

A DHCP client is started on all the VLAN interfaces created in the previous step. To identify itself as a device undergoing ZTP, the DHCP client on Cisco ME 1200 NID adds DHCP Option 60 to the DHCPDISCOVER/DHCPREQUEST messages on the newly-discovered management VLAN, that it sends to the DHCP server.

The Option 60 Vendor Class Identifier in DHCPDISCOVER/DHCPREQUEST message is encoded as a unique ASCII string formed by concatenating the string "*ME1200*" with the complete Cisco ME 1200 NID MAC address in the form similar to ME1200-XX- XX-XX-XX-XX-XX, for example, ME1200-00-01-C1-00-00-00. In addition, as part of the parameters list sent in DHCPREQUEST, Cisco ME 1200 NID also requests the DHCP server to send following options:

- Option 43—This option is used by client to accept the DHCP ACK only from DHCP server or the ME 3600x device specifically configured for it. The Vendor Specific Information in Option 43 is the IP address of the TFTP server that contains the configuration file.

- Option 67—This is the startup configuration filename.
- Option 3—Default gateway
- Option 6—DNS Server

Note Options 60, 43, and 67 are not used when the DHCP client is used in a non-ZTP mode.

Wait for the DHCP Client(s) to Enter the Bound State

When multiple DHCP clients have been started, the first client to reach bound state with requested DHCP options is used further in the ZTP activation process.

Note If no DHCP client reaches bound state with the requested options within 120 seconds, ZTP activation process enters error state.

Once the connectivity is established between the Cisco ME 1200 NID and the ME 3600x device, the Cisco ME 1200 NID can be remotely managed from the UPE NID Controller.

A database of NID instances, MAC addresses, IP addresses, TFTP server, attached physical ports is maintained on the ME 3600x device. This information can be used for management of selected Cisco ME 1200 NIDs.

The association between the Cisco ME 1200 NID and ME 3600x device is maintained by sending and receiving periodic IP-based heartbeat messages.

Step 5—Download and Apply the Initial Configuration

When the DHCP client on the Cisco ME 1200 NID receives the DHCPACK, it uses the information from Option 43 and Option 67 from the DHCP client to download the initial configuration file. This configuration file is intended as a pre-staging configuration, containing basic reachability information such as the gateway, TFTP, DNS server, or the default VLAN configuration so that one or more NIDs can be added to network prior to ZTP auto-configuration. But if there is no requirement, this can be an empty configuration file. The intention of this step is to ensure that ME1200 NID can be reachable to gateway, TFTP, DNS servers. The filename should be specified as an ASCII string using Option 67 in DHCP pool configuration on ME 3600x device.

If the download operation fails or if the configuration could not be applied, the ZTP process enters the error state.

Note The value of the Option 67 field will be taken as the configuration filename, including any file extension.

Step 6—Reverse DNS Lookup to Obtain Hostname

Using the IP address of the DHCP client and the DNS server provided by the DHCP client the Cisco ME 1200 NID performs a reverse DNS query to derive its host name. When the host name is derived, it is added to the current running configuration.

In case of failure, the reverse DNS process is retried five times. After five retries, the host name is configured with the MAC address of the device encoded in the format as: 00_01_C1_00_00_00 (hex string values in uppercase), where 00:01:C1:00:00:00 is a sample ME1200 NID MAC address. This allows ZTP process to continue.

Step 7—Download and Apply Specific Configuration

To perform Reverse DNS lookup, the DNS server must include the definition for a forward zone and a reverse zone. Specifically, the forward zone must include definitions for:

- Authoritative name server (NS record) containing fully-qualified domain name (FQDN) to be used for response.
- Address (A) record with hostname-IP address mapping.

The reverse zone must include the PTR record with IP address-name mapping.

Note FQDN is set as hostname at the end of this step.

The following is a sample DNS configuration:

```
NS    nid1.example.com
A    192.168.2.100
nid1    IN    A 192.168.2.100
nid2    IN    A 192.168.2.101
nid3    IN    A 192.168.2.102
nid4    IN    A 192.168.2.103
...

2.168.192.in-addr.arpa.
PTR    server.example.com.
101    IN    PTR    nid2.example.com.
102    IN    PTR    nid3.example.com.
103    IN    PTR    nid4.example.com.
```

The specific configuration of the device is downloaded from the same TFTP server as the initial configuration. If reverse DNS lookup fails to retrieve a hostname, then ZTP activation process looks for a filename with the following format on TFTP server:

```
NID_MAC_ADDR_CFG.xml
```

For example, 00_3B_99_FE_5E_00_CFG.xml(hex string values in uppercase)

If there is no file stored in .xml format, then ZTP process enters the error state. In the error state, status LED is set to blinking red.

Note It is recommended that user makes 2 file copies of the saved, intended configuration - one named as per the FQDN, such as nid2.example.com.xml, and the other based on the MAC address, such as 00_3B_99_FE_5E_00_CFG.xml.

Step 8—Copy Running Configuration to Startup Configuration

The first time ZTP is performed, you must store a default xml configuration in the fqdn.xml derived from rDNS and in NID_MAC_ADDR_CFG.xml. The following is the content of this default xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://new.webservice.namespace" xmlns:SOAP-ENV="http
://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <run_cfg_resp>
      </run_cfg_resp>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Only after this configuration is applied on Cisco ME 1200 NID, Step 8 will generate a complete XML configuration and store the file as flash:startup-config.xml. This file can be used to replace the default configuration in fqdn.xml and NID_MAC_ADDR_CFG.xml as required.

As a last step in ZTP activation, the running configuration (which was the result of Cisco ME 1200 NID-specific configuration applied after reverse DNS and TFTP download) is copied to startup configuration (flash:startup-config.xml). This ensures that running configuration is persistent.

If there is an error in copying the configuration, the ZTP process enters the error state.

The ZTP process has completed, the status LED is set to solid green.

CHAPTER 6

Auto Discovery of Cisco ME 1200 NIDs

This chapter describes how the UCS controller performs Auto Discovery of Cisco ME 1200 NIDs which are reachable in the network. The UCS controller has to maintain the device information and monitor the notifications sent by the discovered NIDs. Key steps performed during Auto Discovery are:

- Auto Discovery of NID Devices after ZTP using ProvisionNotifications configuration template.
- Addition of dynamic entries for each registered NID in UCS Controller database.
- Monitoring the health of each NID using Heartbeat Notifications after discovery.

Information about Auto Discovery

The sequence of steps that lead to Auto Discovery and monitoring of NIDs is as follows:

1 NID acquires IP address using ZTP and downloads the configuration XML file from TFTP server.

To perform Auto Discovery, operator needs to first manually edit the two configuration XML files applicable to each Cisco ME 1200 NID and described in ZTP chapter, steps 7 and 8 (for example, <fqdn>.xml and <NID_MAC_ADDR_CFG>.xml). This step needs to be performed prior to ZTP trigger so that there is no manual intervention during ZTP process. These files are usually stored on TFTP server and downloaded at the end of the ZTP process. For more details about these configuration files, see Zero Touch Provisioning, on page 53 chapter.

a Open the XML configuration file stored on the TFTP server using any XML editor, and search for the <notification_client_register_config> XML tag with <soap_action>setNotificationReceiver</soap_action>. Replace this XML tag as follows by filling details of the UCS Controller IP address, login username and password. A sample XML is provided for better understanding below.

```
<notification_client_register_config>
             <soap_action>setNotificationReceiver</soap_action>
             <xml_file_content>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://new.webservice.namespace"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <setNotificationReceiver_req>
         <http_server_address>7.3.0.251</http_server_address>
         <http_file_path>/nid_notification</http_file_path>
         <http_BA_user>niduser</http_BA_user>
         <http_BA_password>lab</http_BA_password>
         <heartbeat>30</heartbeat>
      </setNotificationReceiver_req>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope></xml_file_content>
         </notification_client_register_config>
```

b In the same configuration file, search for the XML tag <notification_client_list_config> and replace as follows:

```
<notification_client_list_config>
            <soap_action>registerForNotification</soap_action>
            <xml_file_content>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://new.webservice.namespace"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <regForNotification_req>
         <notifications_list>
            <module_id>1</module_id>
            <notification_id>1</notification_id>
            <enable>true</enable>
         </notifications_list>
         <cookie>5120</cookie>
      </regForNotification_req>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope></xml_file_content>
         </notification_client_list_config>
```

2. Once ZTP is triggered, after the above configurations are added to configuration XML files stored in TFTP server for a specific NID, the Cisco ME 1200 NIDs will register themselves for sending notifications to the UCS controller.
3. NID starts sending HTTP POST notifications periodically.
4. The UCS controller will process the notification messages. If the notification is discovery message, it adds an entry into NID database for the first time. Subsequent messages are considered as heart beat messages.
5. A timer will start after receiving the first discovery message. If three consecutive discovery messages are not received, the connection to that particular NID will be torn down and entry in NID database will be removed.
6. On NID reload, it has to register for notifications on boot up and should start sending discovery notifications.
7. On controller reload, it waits for the notifications. It has to start the heartbeat timer for the NIDs present in database.

CHAPTER 7

Configuring Synchronous Ethernet

The Cisco ME 1200 NID support Synchronous Ethernet (SyncE), which is the PHY-layer frequency-synchronization solution for IEEE 802.3 links. It is an evolution of the conventional Ethernet and Ethernet + SDH and SONET-based synchronization. SyncE is used to synchronize and send clock information to remote sites on the network. Each network element along the synchronization path must support SyncE. SyncE provides only frequency synchronization, not related to time or space.

Prerequisites for Configuring SyncE

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.

Restrictions for Configuring SyncE

- The port number three (3) cannot be nominated to source number one (1).

Information About Synchronous Ethernet

This chapter describes the Synchronous Ethernet features, standards, and limitations in the Cisco ME 1200 Series Carrier Ethernet Access Device. This chapter also describes procedures to configure Synchronous Ethernet.

Synchronous Ethernet Overview

A separate external time-division multiplexing (TDM) circuit is required to provide synchronized timing to multiple remote network elements (NEs) for packet transport networks like Cisco Carrier Packet Transport system. The Synchronous Ethernet (SycnE) feature addresses this requirement by providing effective timing to the remote NEs through a packet network without using an external circuit for timing.

With Ethernet equipment gradually replacing existing Synchronous Optical Networking (SONET) and Synchronous Digital Hierarchy (SDH) equipment in service-provider networks, frequency synchronization is required to provide high-quality clock synchronization over Ethernet ports. The SyncE feature provides the required synchronization at the physical level. Operation messages maintain SyncE links and ensure that a node always derives timing from the most reliable source. SyncE uses the Ethernet Synchronization Message Channel (ESMC) to enable traceability of the best clock source to correctly define the timing source and prevent a timing loop.

The Cisco ME 1200 Series Carrier Ethernet Access Device supports Synchronous Ethernet (SyncE), which is the physical layer frequency-synchronization solution for IEEE 802.3 links. SyncE is defined by the ITU-T standards such as G.8261, G.8262, G.8264, and G.781. It is an evolution of the conventional Ethernet and Ethernet + SDH and SONET-based synchronization. SyncE is used to synchronize and send clock information to remote sites on the network. For SyncE to work, each network element along the synchronization path must support SyncE. SyncE provides only frequency synchronization, not related to time or space.

Understanding SyncE

SyncE provides the Ethernet physical layer network (PHY) level frequency distribution of known common precision frequency references. Clocks for use in SyncE are compatible with the clocks used in the SONET/SDH synchronization network. To achieve network synchronization, synchronization information is transmitted through the network via synchronous network connections with performance of egress clock. In SONET/SDH the communication channel for conveying clock information is SSM, and in SyncE it is the ESMC.

SyncE is a standard for distribution of frequency over Ethernet links. Other standards (IEEE Std. 1588 Precision Time Protocol [PTP], IETF Network Time Protocol [NTP], and so on) have been and are being developed or enhanced for high-quality time distribution and Adaptive Clock Recovery (ACR) requirements.

To maintain the timing chain in SONET/SDH, operators often use SSM. Information provided by SSM Quality Levels (SSM-QL) helps a node derive timing from the most reliable source and prevent timing loops. The SONET/SDH header has a QL information present in the S1 bytes of its header. Hence, the SONET/SDH does not require any specific channel for QL information exchange. As the Ethernet does not have the QL information in its header, it requires ESMC for QL information. Because Ethernet networks are not required to be synchronous on all links or in all locations, a specific channel, the ESMC channel defined in G.8264, provides this service. ESMC is composed of the standard Ethernet header for an organization-specific slow protocol, the ITU-T OUI; a specific ITU-T subtype; an ESMC-specific header; a flag field; and a type, length, value (TLV) structure: the use of flags and TLVs aimed at improving the management of Synchronous Ethernet links and the associated timing change.

For more information, see Configuring Synchronous Ethernet.

SyncE Standards

- ITU-T G.8261: Timing and synchronization aspects in packet network

- ITU-T G.8262: Timing characteristics of Synchronous Ethernet equipment slave clock
- ITU-T G.8264: Distribution of timing through packet networks
- ITU-T G.781: Synchronization layer functions

Understanding SyncE Protocols

Network clocking uses the Synchronization Status Messages (SSM) mechanism to exchange the Quality Level (QL) of the clock between the network elements. In Ethernet, Ethernet Synchronization Message Channel (ESMC) is used for SSM exchange.

The two important protocols used for SyncE are:

- Synchronization Status Messages (SSM)
- Ethernet Synchronization Messaging Channel (ESMC)

Synchronization Status Messages (SSM)

Network elements use Synchronization Status Messages (SSM) to inform the neighboring elements about the Quality Level (QL) of the clock. The non-ethernet interfaces such as optical interfaces and SONET/T1/E1 SPA framers uses SSM. The key benefits of the SSM functionality:

- Prevents timing loops.
- Provides fast recovery when a part of the network fails.
- Ensures that a node derives timing from the most reliable clock source.

Ethernet Synchronization Messaging Channel (ESMC)

To maintain a logical communication channel in synchronous network connections, ethernet relies on a channel called Ethernet synchronization Messaging Channel (ESMC). This is based on IEEE 802.3 Organization Specific Slow Protocol standards. ESMC relays the SSM code that represents the Quality Level (QL) of the Ethernet Equipment Clock (EEC) in a physical layer.

The ESMC packets are received only for those ports configured as clock sources and transmitted on all the SyncE interfaces in the system. These packets are then processed by the Clock selection algorithm and are used to select the best clock. The Tx frame is generated based on the QL value of the selected clock source and sent to all the enabled SyncE ports.

Understanding SyncE Clocks

Clock Selection Algorithm

The clock selection algorithm selects the best available synchronization source from the nominated sources. This algorithm exhibits nonrevertive behavior among the clock sources with the same QL value, and always selects the signal with the best QL value. For clock option SDH, the default is revertive, and for clock option SONET, the default is nonrevertive.

The following parameters contribute to the selection process:

- Quality level (QL)
- Signal fail through QL-FAILED

- Priority
- External commands (Manual, Auto-revertive and so on)

Clock Selection Modes

A clock selection is said to be the best, when the clock source is configured with the highest QL and with the highest priority (for the ones with equal QL).

The following are different clock selection modes:

- **Manual—**the clock selector is manually set to the chosen clock source. If the manually selected clock source fails, then, the clock selector goes to the holdover state.
- **Selected—**the clock selector selects the clock manually, however, the highest priority selected clock source becomes the Source.
- **NonRevertive—**the clock selector selects the best clock source only done when the selected clock fails.
- **Revertive—**the selection of the best clock source is constantly searched for.
- **Holdover—**the clock selector is forced to the holdover state.
- **Freerun—**the clock selector is forced to the free run state.

Manual mode is used to force selection of a specific source. It is also used to switch back to the primary source if auto-nonrevertive mode is selected and the failure is cleared. Selected mode is used to freeze the current clock source, in case of a failure on switchover.

How to Configure SyncE

Configuring SyncE Global Defaults

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**setSyncEglobalDefaultConfig set_global_default_config** **Example:** `UCS(SyncE)# setSyncEglobalDefaultConfig set_global_default_config`	Sets the global configuration to defaults. This means that the SyncE feature is not configured on the device.

	Command or Action	Purpose
Step 4	**setSyncEglobalDefaultConfig review** **Example:** `UCS(SyncE)# setSyncEglobalDefaultConfig review`	Displays the configuration.
Step 5	**setSyncEglobalDefaultConfig commit** **Example:** `UCS(SyncE)# setSyncEglobalDefaultConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS(SyncE)# SyncE
UCS(SyncE)# setSyncEglobalDefaultConfig set_global_default_config
UCS(SyncE)# setSyncEglobalDefaultConfig review

Commands in queue:
    setSyncEglobalDefaultConfig set_global_default_config

UCS(SyncE)# setSyncEglobalDefaultConfig commit

    SetSyncEglobalDefaultConfig Commit Success!!!

UCS(SyncE)# exit
```

Viewing SyncE Global Defaults

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**getSyncEglobalconfig get_global_config** **Example:** `UCS(SyncE)# getSyncEglobalconfig get_global_config`	Displays the SyncE global configuration details.

	Command or Action	Purpose
Step 4	**getSyncEglobalconfig review** **Example:** `UCS(SyncE)# getSyncEglobalconfig review`	Displays the configuration that are in queue.
Step 5	**getSyncEglobalconfig commit** **Example:** `UCS(SyncE)# getSyncEglobalconfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# getSyncEglobalconfig get_global_config
UCS(SyncE)# getSyncEglobalconfig review

Commands in queue:
    getSyncEglobalConfig get_global_config

UCS(SyncE)# getSyncEglobalconfig commit

   GetSyncEglobalConfig_Output.synce_global_conf.clock_select_config.t = 5
   GetSyncEglobalConfig_Output.synce_global_conf.clock_select_config.u.revertive = ''
   GetSyncEglobalConfig_Output.synce_global_conf.wait_to_restore = 5
   GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_holdover.t = 1
   GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_holdover.u.QL_NONE = ''
   GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_freerun.t = 1
   GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_freerun.u.QL_NONE = ''
   GetSyncEglobalConfig_Output.synce_global_conf.EEC_Option.t = 1
   GetSyncEglobalConfig_Output.synce_global_conf.EEC_Option.u.EEC1 = ''

   GetSyncEglobalConfig Commit Success!!!

UCS(SyncE)# exit
```

Configuring SyncE Clock Defaults

This task configures the SyncE configurations to defaults.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**setSyncEclockDefaultConfig set_synce_clock_config_defaults_req** **Example:** `UCS(SyncE)# setSyncEclockDefaultConfig set_synce_clock_config_defaults_req`	Set SyncE default Clock configurations.
Step 4	**setSyncEclockDefaultConfig review** **Example:** `UCS(SyncE)# setSyncEclockDefaultConfig review`	Displays the configuration.
Step 5	**setSyncEclockDefaultConfig commit** **Example:** `UCS(SyncE)# setSyncEclockDefaultConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# setSyncEclockDefaultConfig set_synce_clock_config_defaults_req
UCS(SyncE)# setSyncEclockDefaultConfig review

Commands in queue:
    setSyncEclockDefaultConfig set_synce_clock_config_defaults_req

UCS(SyncE)# setSyncEclockDefaultConfig commit

    SetSyncEclockDefaultConfig Commit Success!!!

UCS(SyncE)# exit
```

Viewing SyncE Clock Defaults

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**getSyncEclockdefaultConfig get_synce_clock_config_defaults_req** **Example:** `UCS(SyncE)# getSyncEclockdefaultConfig get_synce_clock_config_defaults_req`	Displays the SyncE default Clock configurations.
Step 4	**getSyncEclockdefaultConfig review** **Example:** `UCS(SyncE)# getSyncEclockdefaultConfig review`	Displays the configuration.
Step 5	**getSyncEclockdefaultConfig commit** **Example:** `UCS(SyncE)# getSyncEclockdefaultConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# getSyncEclockdefaultConfig get_synce_clock_config_defaults_req
UCS(SyncE)# getSyncEclockdefaultConfig review

Commands in queue:
getSyncEclockDefaultConfig get_synce_clock_config_defaults_req

UCS(SyncE)# getSyncEclockdefaultConfig commit

  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].state = false
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].port = 1
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].priority = 0
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].SSM_overwrite.t = 1
```

```
 GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].SSM_overwrite.u.QL_NONE
 = ''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].hold_off.t = 1
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].hold_off.u.disabled
 = ''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].aneg_mode.t = 1
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[0].aneg_mode.u.none =
''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].state = false
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].port = 2
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].priority = 0
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].SSM_overwrite.t = 1
 GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].SSM_overwrite.u.QL_NONE
 = ''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].hold_off.t = 1
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].hold_off.u.disabled
 = ''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].aneg_mode.t = 1
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[1].aneg_mode.u.none =
''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].state = false
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].port = 3
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].priority = 0
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].SSM_overwrite.t = 1
 GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].SSM_overwrite.u.QL_NONE
 = ''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].hold_off.t = 1
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].hold_off.u.disabled
 = ''
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].aneg_mode.t = 1
  GetSyncEclockDefaultConfig_Output.clock_sel_config.source_configs[2].aneg_mode.u.none =
''
 GetSyncEclockDefaultConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_1_UNI
 = false
 GetSyncEclockDefaultConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_2_UNI
 = false
 GetSyncEclockDefaultConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_3_UNI
 = false
 GetSyncEclockDefaultConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_4_UNI
 = false
 GetSyncEclockDefaultConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_5_UNI
 = false
 GetSyncEclockDefaultConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_6_UNI
 = false

  GetSyncEclockDefaultConfig Commit Success!!!

UCS(SyncE)# exit
```

Configuring the Clock Source

Configuring Clock Source

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.

Step 3	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs* **port** *Physical port* }
	
	Example:
	`UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 port 2`
		Configures the clock source on the port.
		• **source_configs—**Specifies the source configurations.
		  ◦ *Physical port***—**Physical port. The range is from 1 to 6.
		• **port—**Specifies the physical port.
		  ◦ *source_configs***—**nominate a port number to be the clock source. The range is from 1 to 2.

Step 4	**setSyncEclockConfig review** **Example:** `UCS(SyncE)# setSyncEclockConfig review`	Displays the configuration.
Step 5	**setSyncEclockConfig commit** **Example:** `UCS(SyncE)# setSyncEclockConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 port 2
UCS(SyncE)# setSyncEclockConfig review

Commands in queue:
  setSyncEclockConfig clock_sel_config source_configs 0 port 2

UCS(SyncE)# setSyncEclockConfig commit

  SetSyncEclockConfig Commit Success!!!

UCS(SyncE)# exit
```

Viewing Clock Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**getSyncEclockConfig get_clock_config** **Example:** `UCS(SyncE)# getSyncEclockConfig get_clock_config`	Displays clock configuration.
Step 4	**setSyncEclockConfig review** **Example:** `UCS(SyncE)# setSyncEclockConfig review`	Displays the configuration.
Step 5	**getSyncEclockConfig commit** **Example:** `UCS(SyncE)# getSyncEclockConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# getSyncEclockConfig get_clock_config
UCS(SyncE)# setSyncEclockConfig review

Commands in queue:
 getSyncEclockConfig get_clock_config

UCS(SyncE)# getSyncEclockConfig commit

   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].state = true
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].port = 4
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].priority = 1
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].SSM_overwrite.t = 2
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].SSM_overwrite.u.QL_PRC =
'0'
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].hold_off.t = 2
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].hold_off.u.value = 800
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].aneg_mode.t = 1
```

```
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[0].aneg_mode.u.none = ''
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].state = true
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].port = 3
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].priority = 0
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].SSM_overwrite.t = 2
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].SSM_overwrite.u.QL_PRC =
'0'
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].hold_off.t = 2
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].hold_off.u.value = 1000
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].aneg_mode.t = 1
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[1].aneg_mode.u.none = ''
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].state = false
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].port = 3
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].priority = 0
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].SSM_overwrite.t = 1
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].SSM_overwrite.u.QL_NONE =
 ''
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].hold_off.t = 1
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].hold_off.u.disabled = ''
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].aneg_mode.t = 1
   GetSyncEClockConfig_Output.clock_sel_config.source_configs[2].aneg_mode.u.none = ''
   GetSyncEClockConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_1_UNI =
false
   GetSyncEClockConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_2_UNI =
false
   GetSyncEClockConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_3_UNI =
true
   GetSyncEClockConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_4_UNI =
true
   GetSyncEClockConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_5_UNI =
false
   GetSyncEClockConfig_Output.clock_sel_config.ssm_enable_ports.GigabitEthernet_6_UNI =
false

   GetSyncEClockConfig Commit Success!!!

UCS(SyncE)# exit
```

Overwriting the Quality Level (QL)

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**setSyncEclockConfig clock_sel_config** {**source_configs** | **ssm_enable_ports** {**GigabitEthernet_1_UNI** | **GigabitEthernet_2_UNI** | **GigabitEthernet_3_UNI** | **GigabitEthernet_4_UNI** | **GigabitEthernet_5_UNI** | **GigabitEthernet_6_UNI**} {**disable** | **enable**}}}	Enters the SyncE clock configuration to select the UNI ports. Here selecting the physical port 4: • **GigabitEthernet_1_UNI—**Physical port 1. • **GigabitEthernet_2_UNI—**Physical port 2.

	Command or Action	Purpose
	Example: `UCS(SyncE)# setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_4_UNI enable`	• **GigabitEthernet_3_UNI—**Physical port 3. • **GigabitEthernet_4_UNI—**Physical port 4. • **GigabitEthernet_5_UNI—**Physical port 5. • **GigabitEthernet_6_UNI—**Physical port 6. • **disable—**Disables the SSM on the configured port. • **enable—**Enables the SSM on the configured port.
Step 4	**setSyncEclockConfig clock_sel_config** {**source_configs** *source_configs* **port** *Physical port* } **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 port 4`	Configures the clock source on the port: • *source_configs*—Nominates a clock source, either 1 or 2. • *Physical port*—Physical port. The range is from 1 to 6.
Step 5	**setSyncEclockConfig clock_sel_config** {**source_configs** *source_configs* **SSM_overwrite** {**QL_DNU** | **QL_EEC1** | **QL_EEC2** | **QL_INV** | **QL_NONE** | **QL_PRC** | **QL_SSUA QL_SSUB**}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 SSM_overwrite QL_PRC`	Selects QL value to overwrite any received QL in an SSM message • *source_configs*—Nominate a port number to be the clock source. The range is from 1 to 2.
Step 6	**setSyncEclockConfig review** **Example:** `UCS(SyncE)# setSyncEclockConfig review`	Displays the configuration.
Step 7	**setSyncEclockConfig commit** **Example:** `UCS(SyncE)# setSyncEclockConfig commit`	Sends the configuration to the NID.
Step 8	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_4_UNI
enable
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 port 4
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 SSM_overwrite QL_PRC
UCS(SyncE)# setSyncEclockConfig review
```

```
Commands in queue:
   setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_4_UNI enable
   setSyncEclockConfig clock_sel_config source_configs 1 port 4
   setSyncEclockConfig clock_sel_config source_configs 0 SSM_overwrite QL_PRC

UCS(SyncE)# setSyncEclockConfig commit

   SetSyncEclockConfig Commit Success!!!

UCS(SyncE)# exit
```

Understanding Clock Redundancy

On the Cisco ME 1200 NID, it is possible to configure up to two clock sources. Any Ethernet port can act as a clock source. For the Cisco ME 1200 NID, external clock input does not exit. Based on the priority and Quality level (QL) of the clock sources, the best source is selected.

To select the best source, nominate the clock sources, and then set priorities for each of them. Enable SSM on ports used for synchronization. Note that QL overwrites the priority. That means, if port 2 receives QL-PRC and port 1 receives only QL-EEC1, and even though port 1 has higher priority than port 2, the port 2 is selected as QL overwrites.

Configuring Clock Redundancy

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**setSyncEclockConfig clock_sel_config** {**source_configs** | **ssm_enable_ports** {**GigabitEthernet_1_UNI** | **GigabitEthernet_2_UNI** | **GigabitEthernet_3_UNI** | **GigabitEthernet_4_UNI** | **GigabitEthernet_5_UNI** | **GigabitEthernet_6_UNI**} {**enable** | **disable**}}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_5_UNI enable`	Enters the SyncE clock configuration to select the UNI ports. Here selecting the physical port 5. • **GigabitEthernet_1_UNI**—Physical port 1. • **GigabitEthernet_2_UNI**—Physical port 2. • **GigabitEthernet_3_UNI**—Physical port 3. • **GigabitEthernet_4_UNI**—Physical port 4. • **GigabitEthernet_5_UNI**—Physical port 5. • **GigabitEthernet_6_UNI**—Physical port 6. • **disable**—Disables the SSM on the configured port. • **enable**—Enables the SSM on the configured port.

	Command or Action	Purpose
Step 4	**setSyncEclockConfig clock_sel_config** {**source_configs** | **ssm_enable_ports** {**GigabitEthernet_1_UNI** | **GigabitEthernet_2_UNI** | **GigabitEthernet_3_UNI** | **GigabitEthernet_4_UNI** | **GigabitEthernet_5_UNI** | **GigabitEthernet_6_UNI**} {**enable** | **disable**}}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_6_UNI enable`	Enters the SyncE clock configuration to select the UNI ports. Here selecting the physical port 6. • **GigabitEthernet_1_UNI—**Physical port 1. • **GigabitEthernet_2_UNI—**Physical port 2. • **GigabitEthernet_3_UNI—**Physical port 3. • **GigabitEthernet_4_UNI—**Physical port 4. • **GigabitEthernet_5_UNI—**Physical port 5. • **GigabitEthernet_6_UNI—**Physical port 6. • **disable—**Disables the SSM on the configured port. • **enable—**Enables the SSM on the configured port.
Step 5	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs* **port** *Physical port*} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 port 5`	Configures the clock source on the port. Here the configuration is done on port 5, and the *source_config* is set to 1. • *source_configs***—**Nominate a port number to be the clock source. The range is from 1 to 2. • *Physical port***—**Physical port. The range is from 1 to 6.
Step 6	**setSyncEclockConfig clock_sel_config** {**source_configs** {**priority** *priority*}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 priority 0`	Sets the clock priority. Here the clock priority is set to 0. • *priority***—**Clock priority value. Either 0 or 1.
Step 7	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs* **state** {**enable** | **disable**}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 state enable`	Enables or Disables the clock source. • *source_configs***—**nominate a port number to be the clock source. The range is from 1 to 2.
Step 8	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs***hold_off** {**disabled** | | **value** {**300 msec to 1800 msec**}}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 hold_off value 1000`	Sets the Hold-off timer value. Active loss of clock source is delayed by the selected amount of time. The clock selector changes the clock source if the loss of clock condition is cleared within this time. • *source_configs***—**nominate a port number to be the clock source. The range is from 1 to 2.

	Command or Action	Purpose
Step 9	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs* **port** *Physical port*} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 port 5`	Configures the clock source on the port. Here the configuration is done on port 6, and the source_config is set to 0. • *source_configs*—Nominate a port number to be the clock source. The range is from 1 to 2. • *Physical port*—Physical port. The range is from 1 to 6.
Step 10	**setSyncEclockConfig clock_sel_config** {**source_configs** {**priority** *priority* }} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 priority 1`	Sets the clock priority. Here the clock priority is set to 1. • *priority*—Clock priority value. Either 0 or 1.
Step 11	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs* **state** {**enable** | **disable**}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 state enable`	Enables or Disables the clock source. • *source_configs*—Nominate a port number to be the clock source. The range is from 1 to 2.
Step 12	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs***hold_off** {**disabled** | **value** {**300 msec to 1800 msec**}}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 hold_off value 800`	Sets the Hold-off timer value. Active loss of clock source is delayed by the selected amount of time. The clock selector changes the clock source if the loss of clock condition is cleared within this time. • *source_configs*—Nominate a port number to be the clock source. The range is from 1 to 2.
Step 13	**setSyncEclockConfig review** **Example:** `UCS(SyncE)# setSyncEclockConfig review`	Displays the configuration.
Step 14	**setSyncEclockConfig commit** **Example:** `UCS(SyncE)# setSyncEclockConfig commit`	Sends the configuration to the NID.
Step 15	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_5_UNI
enable
UCS(SyncE)# setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_6_UNI
enable
```

```
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 port 5
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 priority 0
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 state enable
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 hold_off value 1000
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 port 6
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 priority 1
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 state enable
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 hold_off value 800

UCS(SyncE)# setSyncEclockConfig review

Commands in queue:
 setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_5_UNI enable
 setSyncEclockConfig clock_sel_config ssm_enable_ports GigabitEthernet_6_UNI enable
 setSyncEclockConfig clock_sel_config source_configs 1 port 5
 setSyncEclockConfig clock_sel_config source_configs 1 priority 0
 setSyncEclockConfig clock_sel_config source_configs 1 state enable
 setSyncEclockConfig clock_sel_config source_configs 1 hold_off value 1000
 setSyncEclockConfig clock_sel_config source_configs 0 port 6
 setSyncEclockConfig clock_sel_config source_configs 0 priority 1
 setSyncEclockConfig clock_sel_config source_configs 0 state enable
 setSyncEclockConfig clock_sel_config source_configs 0 hold_off value 800

UCS(SyncE)# setSyncEclockConfig commit

 SetSyncEclockConfig Commit Success!!!

UCS(SyncE)# exit
```

Understanding SyncE Timers

You can manage syncE timers by changing the priority of the clock sources. You can also influence selection by modifying the following timers:

- WTR (Wait to restore) Timer
- Hold-off Timer

WTR Timer

The WTR time is activated on the falling edge of a clock source failure (in Revertive mode). This means that the clock source is first available for clock selection after WTR Time (can be cleared).

Hold-off Timer

In the Hold-off timer, the active loss of clock source is delayed by the selected amount of time. The clock selector does not change the clock source if the loss of clock condition is cleared within this time.

Configuring SyncE Timers

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**setSyncEglobalConfig synce_global_conf** {**EEC_Option** | **SSM_QL_for_freerun** | **SSM_QL_for_holdover** | **clock_select_config** | **wait_to_restore**}	Enters the SyncE global configuration. • **EEC_Option—**Selects PLL EEC option. • **SSM_QL_for_freerun—**Transmits SSM QL value when clock selector is in Free Run Mode. • **SSM_QL_for_holdover—**Transmits SSM QL value when clock selector is in Hold Over State. • **clock_select_config—**Selection mode of nominated clock sources. • **wait_to_restore—**Select the wait to restore time.
Step 4	**setSyncEglobalConfig synce_global_conf wait_to_restore** *wait to restore time* **Example:** `UCS(SyncE)# setSyncEglobalConfig synce_global_conf wait_to_restore 1`	Enters the wait to restore time. • *wait to restore time***—**Restore time. The range is from 0 to 12 minutes; enter the value zero to disable.
Step 5	**setSyncEglobalConfig synce_global_conf clock_select_config** {**freerun** | **holdover** | **manual** *manually set* | **nonrevertive** | **revertive** | **selected**} **Example:** `UCS(SyncE)# setSyncEglobalConfig synce_global_conf clock_select_config revertive`	Enters the selection mode of nominated clock sources. • **freerun—**Selector is forced in free run. • **holdover—**Selector is forced in holdover. • **manual—**Selector is manually set to chosen clock source. ◦ *manually set***—**Clock source.The range is from 1 to 2. • **nonrevertive—**Automatic clock selection, selecting best clock source nonrevertively. • **revertive—**Automatic clock selection, selecting best clock source revertively. • **selected—**Manual clock selection, selecting pt selected clock source.
Step 6	**setSyncEglobalConfig synce_global_conf SSM_QL_for_holdover** {**QL_DNU** | **QL_EEC1** | **QL_EEC2** | **QL_INV** | **QL_NONE** | **QL_PRC** | **QL_SSUA QL_SSUB**} **Example:** `UCS(SyncE)# setSyncEglobalConfig synce_global_conf SSM_QL_for_holdover QL_EEC1`	Transmits SSM QL value when clock selector is in Hold Over State. • **QL_DNU—**SSM QL value is QL_DNU. • **QL_EEC1—**SSM QL value is QL_EEC1. • **QL_EEC2—** SSM QL value is QL_EEC2.

	Command or Action	Purpose
		• **QL_INV**—SSM QL value is QL_INV. • **QL_NONE**—SSM QL value is QL_NONE. • **QL_PRC**—SSM QL value is QL_PRC. • **QL_SSUA**—SSM QL value is QL_SSUA. • **QL_SSUB**—SSM QL value is QL_SSUB.
Step 7	**setSyncEglobalConfig synce_global_conf SSM_QL_for_freerun** {**QL_DNU** | **QL_EEC1** | **QL_EEC2** | **QL_INV** | **QL_NONE** | **QL_PRC** | **QL_SSUA QL_SSUB**} **Example:** `UCS(SyncE)# setSyncEglobalConfig synce_global_conf SSM_QL_for_freerun QL_EEC2`	Transmits SSM QL value when clock selector is in Free Run Mode. • **QL_DNU**—SSM QL value is QL_DNU. • **QL_EEC1**—SSM QL value is QL_EEC1. • **QL_EEC2**— SSM QL value is QL_EEC2. • **QL_INV**—SSM QL value is QL_INV. • **QL_NONE**—SSM QL value is QL_NONE. • **QL_PRC**—SSM QL value is QL_PRC. • **QL_SSUA**—SSM QL value is QL_SSUA. • **QL_SSUB**—SSM QL value is QL_SSUB.
Step 8	**setSyncEglobalConfig synce_global_conf EEC_Option**{**EEC1** | **EEC2**} **Example:** `UCS(SyncE)# setSyncEglobalConfig synce_global_conf EEC_Option EEC2`	Selects PLL EEC option. • **EEC1**—DPLL bandwidth is 3.5 Hz. • **EEC2**—DPLL bandwidth is 0.1 Hz.
Step 9	**setSyncEglobalConfig review** **Example:** `UCS(SyncE)# setSyncEglobalConfig review`	Displays the configuration.
Step 10	**setSyncEglobalConfig commit** **Example:** `UCS(SyncE)# setSyncEglobalConfig commit`	Sends the configuration to the NID.
Step 11	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# setSyncEglobalConfig synce_global_conf wait_to_restore 1
UCS(SyncE)# setSyncEglobalConfig synce_global_conf clock_select_config revertive
UCS(SyncE)# setSyncEglobalConfig synce_global_conf SSM_QL_for_holdover QL_EEC1
UCS(SyncE)# setSyncEglobalConfig synce_global_conf SSM_QL_for_freerun QL_EEC2
UCS(SyncE)# setSyncEglobalConfig synce_global_conf EEC_Option EEC2
UCS(SyncE)# setSyncEglobalConfig review

Commands in queue:
    setSyncEglobalConfig synce_global_conf wait_to_restore 1
    setSyncEglobalConfig synce_global_conf clock_select_config revertive
    setSyncEglobalConfig synce_global_conf SSM_QL_for_holdover QL_EEC1
    setSyncEglobalConfig synce_global_conf SSM_QL_for_freerun QL_EEC2
    setSyncEglobalConfig synce_global_conf EEC_Option EEC2

UCS(SyncE)# setSyncEglobalConfig commit

    SetSyncEglobalConfig Commit Success!!!

UCS(SyncE)# exit
```

Viewing SyncE Timers

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**getSyncEglobalConfig get_global_config** **Example:** `UCS(SyncE)# getSyncEglobalConfig get_global_config`	Displays the SyncE global configuration.
Step 4	**getSyncEglobalConfig review** **Example:** `UCS(SyncE)# getSyncEglobalConfig review`	Sends the configuration to the NID.
Step 5	**getSyncEglobalConfig commit** **Example:** `UCS(SyncE)# getSyncEglobalConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# getSyncEglobalConfig get_global_config
UCS(SyncE)# getSyncEglobalConfig review

Commands in queue:
  getSyncEglobalConfig get_global_config

UCS(SyncE)# getSyncEglobalConfig commit

  GetSyncEglobalConfig_Output.synce_global_conf.clock_select_config.u.revertive = '0'
  GetSyncEglobalConfig_Output.synce_global_conf.wait_to_restore = 1
  GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_holdover.t = 6
  GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_holdover.u.QL_EEC1 = '0'
  GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_freerun.t = 1
  GetSyncEglobalConfig_Output.synce_global_conf.SSM_QL_for_freerun.u.QL_NONE = ''
  GetSyncEglobalConfig_Output.synce_global_conf.EEC_Option.t = 1
  GetSyncEglobalConfig_Output.synce_global_conf.EEC_Option.u.EEC1 = ''

  GetSyncEglobalConfig Commit Success!!!

UCS(SyncE)# exit
```

Understanding ANEG Mode

The Auto-negotiation (ANEG) mode is relevant for 1000BaseT ports only. To recover the clock from a port, the clock must be negotiated to the Slave mode. To distribute the clock, the port must be negotiated to the Master mode.

Following are the different ANEG modes that can be activated on a clock source port:

- **Prefer Slave—**the port negotiates to the Slave mode.
- **Prefer Master—**the port negotiates to the Master mode.
- **Forced Slave—**the port is forced to the Master mode.

Note The port in the **Locked** state always remains negotiated to the **Slave**.

Configuring ANEG mode

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs* **port** *Physical port*} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 port 2`	Configures the clock source on the port. • *source_configs*—Nominate a port number to be the clock source. The range is from 1 to 2. • *Physical port*—Physical port. The range is from 1 to 6.
Step 4	**setSyncEclockConfig clock_sel_config** {**source_configs***source_configs***aneg_mode** {**forced_slave** | **none** |**prefer_master** |**prefer_slave**}} **Example:** `UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 aneg_mode prefer_master`	Configures the ANEG mode that is relevant to ports1 and 2, which are 1000 base T.
Step 5	**setSyncEclockConfig review** **Example:** `UCS(SyncE)# setSyncEclockConfig review`	Displays the configuration.
Step 6	**setSyncEclockConfig commit** **Example:** `UCS(SyncE)# setSyncEclockConfig commit`	Sends the configuration to the NID.
Step 7	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 1 port 2
UCS(SyncE)# setSyncEclockConfig clock_sel_config source_configs 0 aneg_mode prefer_master
UCS(SyncE)# setSyncEclockConfig review

Commands in queue:
  setSyncEclockConfig clock_sel_config source_configs 1 port 2
  setSyncEclockConfig clock_sel_config source_configs 0 aneg_mode prefer_master

UCS(SyncE)# setSyncEclockConfig commit

 SetSyncEclockConfig Commit Success!!!

UCS(SyncE)# exit
```

Verifying SyncE Status

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**syncE** **Example:** `UCS# SyncE`	Enters the syncE mode.
Step 3	**showNetworkClock show_synce_status** **Example:** `UCS(SyncE)# showNetworkClock show_synce_status`	Displays the SyncE status.
Step 4	**exit** **Example:** `UCS(SyncE)# exit`	Exits the SyncE mode.

Configuration Example

```
UCS# SyncE
UCS(SyncE)# showNetworkClock show_synce_status

  ShowNetworkClock_Output.show_network_clock.selector_state.t = 2
  ShowNetworkClock_Output.show_network_clock.selector_state.u.holdover = ''
  ShowNetworkClock_Output.show_network_clock.alarm_state[0].clock_source = 1
  ShowNetworkClock_Output.show_network_clock.alarm_state[0].LOCS = false
  ShowNetworkClock_Output.show_network_clock.alarm_state[0].SSM = false
  ShowNetworkClock_Output.show_network_clock.alarm_state[0].WTR = false
  ShowNetworkClock_Output.show_network_clock.alarm_state[1].clock_source = 2
  ShowNetworkClock_Output.show_network_clock.alarm_state[1].LOCS = true
  ShowNetworkClock_Output.show_network_clock.alarm_state[1].SSM = false
  ShowNetworkClock_Output.show_network_clock.alarm_state[1].WTR = false
  ShowNetworkClock_Output.show_network_clock.alarm_state[2].clock_source = 3
  ShowNetworkClock_Output.show_network_clock.alarm_state[2].LOCS = true
  ShowNetworkClock_Output.show_network_clock.alarm_state[2].SSM = false
  ShowNetworkClock_Output.show_network_clock.alarm_state[2].WTR = false

  ShowNetworkClock Commit Success!!!

UCS(SyncE)# exit
```

CHAPTER 8

Configuring Ethernet Virtual Connections

Ethernet Virtual Connection (EVC) as an association between two or more user network interfaces that identifies a point-to-point or multipoint-to-multipoint path within the service provider network. An EVC is a conceptual service pipe within the service provider network. A bridge domain is a local broadcast domain that is VLAN-ID-agnostic. An ethernet flow point (EFP) service instance is a logical interface that connects a bridge domain to a physical port or to an EtherChannel group in a router.

The Cisco ME 1200 NID supports the application software control modules and interfaces related to EVC.

How to Configure Ethernet Virtual Circuit

Configuring Ethernet Virtual Circuit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionEVC** **Example:** `UCS# ProvisionEVC`	Enters the ProvisionEVC mode.

Step 3	**addEVC evcConfiguration** {**instance***evc_instance_id* | **learning** {**enable** | **disable**} |**nni_ports** | **nni_vid** *nni_vid_outer_tag* | **policer_id***policer_id*}
	
	Example:
	`UCS(ProvisionEVC)# addEVC evcConfiguration nni_vid 101`
	`UCS(ProvisionEVC)# addEVC evcConfiguration learning enable`
	`UCS(ProvisionEVC)# addEVC evcConfiguration nni_ports GigabitEthernet_6_NNI enable`
	`UCS(ProvisionEVC)# addEVC evcConfiguration policer_id 1`
		Adds the EVE configuration.

Step 4	**addEVC review** **Example:** `UCS(ProvisionEVC)# addEVC review`	Reviews the addEVC configuration.
Step 5	**addEVC commit** **Example:** `UCS(ProvisionEVC)# addEVC commit`	Sends the addEVC configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionEVC)# exit`	Exits the ProvisionEVC mode.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addEVC evcConfiguration instance 7
UCS(ProvisionEVC)# addEVC evcConfiguration nni_vid 101
UCS(ProvisionEVC)# addEVC evcConfiguration learning enable
UCS(ProvisionEVC)# addEVC evcConfiguration nni_ports GigabitEthernet_6_NNI enable
UCS(ProvisionEVC)# addEVC evcConfiguration policer_id 1
UCS(ProvisionEVC)# addEVC review
UCS(ProvisionEVC)# addEVC commit

  AddEVC Commit Success!!!
```

Creating a Policer

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionEVC** **Example:** `UCS# ProvisionEVC`	Enters the ProvisionEVC mode.

Step 3	**addPolicerEVC evc_policer** {**cbs***cbs_id* | **cir** *committed_information_rate* | **ebs** *excess_burst_size* | **eir** *excess_information_rate* |**policer_id** *policer_id* | **policer mode** {**color_aware** | **coupled**} |**policer_type** {**mef** | **single**} | **rate_type** {**data** | **line**} | **state** {**enabled** | **disabled**}}
	
	Example:
	`UCS(ProvisionEVC)# addPolicerEVC evc_policer cir 20000`
	`UCS(ProvisionEVC)# addPolicerEVC evc_policer ebs 30000`
	`UCS(ProvisionEVC)# addPolicerEVC evc_policer eir 40000`
	`UCS(ProvisionEVC)# addPolicerEVC evc_policer policer_id 1`
	`UCS(ProvisionEVC)# addPolicerEVC evc_policer state enabled`
		Adds the EVC Policer.
		• **cbs—**Specifies the committed burst size in bytes.
		• **cir—**Specifies the committed information rate. Multiply by 1000 to get rate in BPS.
		• **ebs—**Specifies the excess burst size in bytes.
		• **eir—**Specifies the excess information rate.
		• **policer_id—**Specifies the Policer ID. The valid values are from 1 to 1022.
		• **policer_mode—**Specifies the Policer mode—whether **color-aware** or **coupled**.
		• **policer_type—**Specifies the Policer mode—whether **mef** or **single**.
		• **rate_type—**Specifies the rate type policing—whether **data** or **line**.
		• **state—**Specifies the policer state—whether **enabled** or **disabled**.

Step 4	**addPolicerEVC review** **Example:** `UCS(ProvisionEVC)# addPolicerEVC review`	Displays the addPolicerEVC configuration.

Step 5	**addPolicerEVC commit**
	
	Example:
	`UCS(ProvisionEVC)# addPolicerEVC commit`
	`AddPolicerEVC Commit Success!!!`
		Sends the configuration to the NID.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(ProvisionEVC)# exit`	Exits from the ProvisionEVC mode.

Number of policers allowed are 1022. Use the following scale numbers for the ECE or EVC configuration with or without configuring QoS with tag pop 0, 1, or 2:

- Maximum 510 ECEs can be configured with or without configuring QoS (0-7 COS) with one NNI port to one UNI port.
- Maximum of 340 ECEs can be configured with or without configuring QoS (0-7 COS) with two NNI ports to one UNI or one NNI port to two UNI ports.
- Maximum of 255 ECEs can be configured with or without configuring QoS (0-7 COS) with three NNI ports to one UNI port or one NNI port to three UNI ports.
- Maximum of 170 ECEs can be configured with or without configuring QoS (0-7 COS) with four NNI ports to one UNI port or one NNI port to four UNI ports.
- Maximum of 128 ECEs can be configured with or without configuring QoS (0-7 COS) with five NNI ports to one UNI port or one NNI port to five UNI ports.

If OAM, HQoS, or EFP is configured on the Cisco ME 1200 NID, you can configure the following maximum service instances on every UNI interface:

- 64 ECE or EVC with eight COS classes.
- 104 ECE or EVC with four COS classes.
- 104 ECE or EVC with two COS classes.

EVC Control Entry (ECE) Configuration

ECE rules are used to divide the UNI traffic into two service classes.

This division of UNI traffic is achieved through:

- Simple NNI: All EVCs on the NNI port use the same QoS mapping and statistics.

  Note This method requires fewer resources.

- Advanced NNI: Each EVC on the NNI port has separate QoS mapping and statistics.

In the following example, multiple ECE rules are created:

Configuring ECE Sample Rule 1

For rule 1, frames received on the UNI port with PCP 4-7 values are mapped to class 4 and sent with PCP 4 in the outer tag on the NNI port.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionEVC** **Example:** `UCS# ProvisionEVC`	Enters the ProvisionEVC mode.
Step 3	**addECE ece_configuration ece_id** *ece_id* **Example:** `UCS(ProvisionEVC)# addECE ece_configuration ece_id 2`	Adds ECE configuration.

Step 4	**addECE ece_configuration control actions** {**class** {**disabled** | **specific** *specific_id*} | **direction** {**bothnni_to_uni** |**uni_to_nni**} | **drop_precedence** {**disabled** | **one** | **zero**} | **evc_id** {**none** | **specific** *specific_eve_id*} | **policer_id** {**discard** | **evc** | **none** | **specific** *specific_id*} | **policy_id** *acl_policy_id* | **tag_pop_count** *tag_pop_count*}
	
	Example:
	
	`UCS(ProvisionEVC)# addECE ece_configuration control actions evc_id specific 7`
	`UCS(ProvisionEVC)# addECE ece_configuration control actions tag_pop_count 1`
	`UCS(ProvisionEVC)# addECE ece_configuration control actions policer_id specific 1`
	`UCS(ProvisionEVC)# addECE ece_configuration control actions class specific 4`
		Adds the ECE control action configuration.
		• **class—**Specifies the ECE class.
		• **direction—**Specifies the direction of flow of traffic.
		• **drop_precedence—**Specifies the drop precedence (higher value means more dropping).
		• **evc_id—**Specifies the EVC ID. The valid **specific** values are from 1 to 1024.
		• **policer_id—**Specifies the policer ID. The valid **specific** values are from 1 to 1022.
		• **policy_id—**Specifies the ACL policy ID. The valid values are from 0 to 63.
		• **tag_pop_count—**Specifies the tagged VLAN count to be removed (either one or two outermost tags).

Step 5	**addECE ece_configuration control egress-inner-tagaddECE ece_configuration control egress_inner_tag** {**dei-modedei_mode** {**classified** | **drop_prec** | **fixed**} | **dei_value** *dei* | **pcp_mode** {**classified** | **fixed** | **mapped**} | **pcp_value** *pcp_value* | **type** *type* | **vlan_id** *vlan_id*} **Example:** `UCS(ProvisionEVC)# addECE ece_configuration control`	Adds the ECE control egress inner tag rewrite configuration. • **dei_mode—**Specifies the DEI mode—whether **classified**, **drop precedence**, or **fixed**. • **dei_value—**Specifies the DEI value. The valid values are 0 and 1.

	Command or Action	Purpose

	`egress_inner_tag dei_mode classified`
	`UCS(ProvisionEVC)# addECE ece_configuration control egress_inner_tag type none`
	`UCS(ProvisionEVC)# addECE ece_configuration control egress_inner_tag vlan_id 3`
		• **pcp_mode**—Specifies the PCP mode—whether **classified**, **fixed**, or **mapped**.
		• **pcp_value**—Specifies the PCP value. The valid values are from 1 to 7.
		• **type**—Specifies the type—whether **c-tagged**, **none**, **s-custom**, or **s-tagged**.
		• **vlan_id**—Specifies the VLAN ID. The valid values are from 1 to 4095.

Step 6	**addECE ece_configuration control egress_outer_tag** {**dei_mode** {**classified** | **drop_prec** | **fixed**} | **dei_value** *dei_value* | **mode** {**enabled** | **disabled**} | **pcp_mode** {**classified** | **fixed** | **mapped**} | **pcp_value** *pcp_value* | **vlan_id** *vlan_id*
	
	Example:
	
	`UCS(ProvisionEVC)# addECE ece_configuration control egress_outer_tag pcp_mode fixed`
	`UCS(ProvisionEVC)# addECE ece_configuration control egress_outer_tag pcp_value 4`
		Adds the ECE control egress outer tag rewrite configuration.
		• **dei_mode**—Specifies the DEI mode—whether **classified**, **drop precedence**, or **fixed**.
		• **dei_value**—Specifies the DEI value. The valid values are 0 and 1.
		• **mode**—Specifies the mode—whether **enabled** or **disabled**.
		• **pcp_mode**—Specifies the PCP mode—whether **classified**, **fixed**, or **mapped**.
		• **pcp_value**—Specifies the PCP value. The valid values are from 1 to 7.
		• **vlan_id**—Specifies the VLAN ID. The valid values are from 1 to 4095.

Step 7	**addECE ece_configuration control ingress_match** {**frame_type** {**any** | **ipv4** {**dest_ip_filter** | **source_ip_filter**} | **ipv6** {**dest_ip_filter** | **source_ip_filter**}} | **inner_tag_match** {**match_fields** | **match_type**} | **mac_params** {**dmac_filer** | **smac_filter**} | **outer_tag_match** {**match_fields** | **match_type**} | **uni_ports** {**GigabitEthernet_1_UNI** | **GigabitEthernet_2_UNI** | **GigabitEthernet_3_UNI** | **GigabitEthernet_4_UNI** | **GigabitEthernet_5_UNI** | **GigabitEthernet_6_UNI**}}
	
	Example:
	
	`UCS(ProvisionEVC)# addECE ece_configuration control ingress_match uni_ports GigabitEthernet_2_UNI enable`
	`UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_type c_tagged`
	`UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields vlan_id_filter specific 100`
	`UCS(ProvisionEVC)# addECE ece_configuration control`
		Adds the ECE control ingress inner tag rewrite configuration.
		• **frame_type**—Specifies the type of frame relay.
		• **inner_tag_match**—Specifies the inner tag match value.
		• **mac_params**—Specifies the DMAC and SMAC default values.
		• **outer_tag_match**—Specifies the outer tag match value.
		• **uni_ports**—Specifies the GigabitEthernet UNI ports.

	Command or Action	Purpose
	`ingress_match outer_tag_match match_fields inner_pcp val_4-7`	
Step 8	**addECE review** **Example:** `UCS(ProvisionEVC)# addECE review`	Reviews the addECE configuration.
Step 9	**addECE commit** **Example:** `UCS(ProvisionEVC)# addECE commit`	Sends the configuration to the NID.
Step 10	**exit** **Example:** `UCS(ProvisionEVC)# exit`	Exits the ProvisionEVC mode.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addECE ece_configuration ece_id 1
UCS(ProvisionEVC)# addECE ece_configuration ece_id 1
UCS(ProvisionEVC)# addECE ece_configuration control actions evc_id specific 777
UCS(ProvisionEVC)# addECE ece_configuration control actions tag_pop_count 1
UCS(ProvisionEVC)# addECE ece_configuration control actions policer_id none
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match uni_ports
GigabitEthernet_2_UNI enable
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_type
 c_tagged
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields
 vlan_id_filter specific 100
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields
 inner_dei any
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields
 inner_pcp val_any
UCS(ProvisionEVC)# addECE review
Commands in queue:
 addECE ece_configuration ece_id 1
 addECE ece_configuration ece_id 1
 addECE ece_configuration control actions evc_id specific 777
 addECE ece_configuration control actions tag_pop_count 1
 addECE ece_configuration control actions policer_id none
 addECE ece_configuration control ingress_match uni_ports GigabitEthernet_2_UNI enable
 addECE ece_configuration control ingress_match outer_tag_match match_type c_tagged
 addECE ece_configuration control ingress_match outer_tag_match match_fields vlan_id_filter
 specific 100
 addECE ece_configuration control ingress_match outer_tag_match match_fields inner_dei any

 addECE ece_configuration control ingress_match outer_tag_match match_fields inner_pcp
val_any

UCS(ProvisionEVC)# addECE commit

Clearing Socket 4 Clearing Socket 4
 AddECE Commit Success!!!
```

Configuring the ECE Sample Rule 2

For rule 2, other frames received on the UNI port are mapped to class 0 and sent with PCP 0 in the outer tag on the NNI port.

Note The configuration steps are similar to the ones mentioned in the Configuring ECE Rule 1 section.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addECE ece_configuration control actions evc_id specific 7

UCS(ProvisionEVC)# addECE ece_configuration control actions tag_pop_count 1

UCS(ProvisionEVC)# addECE ece_configuration control actions policer_id specific 1
UCS(ProvisionEVC)# addECE ece_configuration control actions class specific 0
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match uni_ports
GigabitEthernet_2_UNI enable
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_type
 c_tagged
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields
 vlan_id_filter specific 99
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields
 inner_pcp val_0-3
UCS(ProvisionEVC)# addECE ece_configuration control egress_outer_tag pcp_mode fixed
UCS(ProvisionEVC)# addECE ece_configuration control egress_outer_tag pcp_value 0
UCS(ProvisionEVC)# addECE commit
```

Configuring ECE Sample Rule 3

For rule 3, frames received on the NNI port 6 with S-tag 101 and C-tag 100 with any PCP values can be mapped to class 4 and sent with PCP 4 on the UNI port.

Note The configuration steps are similar to the ones mentioned in the Configuring ECE Rule 1 section.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addECE ece_configuration ece_id 3
UCS(ProvisionEVC)# addECE ece_configuration control actions evc_id specific 7
UCS(ProvisionEVC)# addECE ece_configuration control actions class specific 4
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match uni_ports
GigabitEthernet_2_UNI enable
UCS(ProvisionEVC)# addECE ece_configuration control egress_inner_tag pcp_mode fixed
UCS(ProvisionEVC)# addECE ece_configuration control egress_inner_tag pcp_value 4
UCS(ProvisionEVC)# addECE commit
```

Configuring ECE Sample Rule 4

For rule 4, insert a new c-tag in frames forwarding to the NNI port.

Note The configuration steps are similar to the ones mentioned in the Configuring ECE Rule 1 section.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addECE ece_configuration ece_id 4
UCS(ProvisionEVC)# addECE ece_configuration control actions evc_id specific 7
UCS(ProvisionEVC)# addECE ece_configuration control actions tag_pop_count 1
UCS(ProvisionEVC)# addECE ece_configuration control actions policer_id specific 1
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match uni_ports
GigabitEthernet_2_UNI enable
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_type
 c_tagged
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields
 vlan_id_filter specific 99
UCS(ProvisionEVC)# addECE ece_configuration control egress_inner_tag type c_tagged
UCS(ProvisionEVC)# addECE ece_configuration control egress_inner_tag vlan_id 77
UCS(ProvisionEVC)# addECE commit
```

Configuring ECE Sample Rule 5

For rule 5, insert a new tag in frames forwarding to the UNI port, the outer tag for NNI - UNI direction for the ECE.

Note The configuration steps are similar to the ones mentioned in the Configuring ECE Rule 1 section.

Example

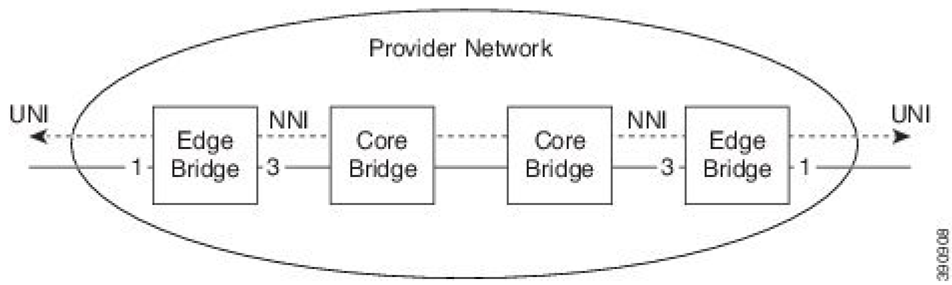
```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addECE ece_configuration ece_id 5
UCS(ProvisionEVC)# addECE ece_configuration control actions direction nni_to_uni
                                            ---> This field is mandatory to pass
UCS(ProvisionEVC)# addECE ece_configuration control actions evc_id specific 7
                                             ---> Map this ECE rule to an EVC configured
above.
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match uni_ports
GigabitEthernet_2_UNI enable
UCS(ProvisionEVC)# addECE ece_configuration control egress_outer_tag enabled
UCS(ProvisionEVC)# addECE ece_configuration control egress_outer_tag vlan_id 78
UCS(ProvisionEVC)# addECE commit
```

Ethernet Private Line or E-LAN

Ethernet Private Line (EPL) or E-LAN and Ethernet Virtual Private Line (EVPL) are Carrier Ethernet data services defined by the Metro Ethernet Forum. E-LAN provides a point-to-point Ethernet virtual connection (EVC) between a pair of dedicated user–network interfaces (UNIs), with a high degree of transparency.

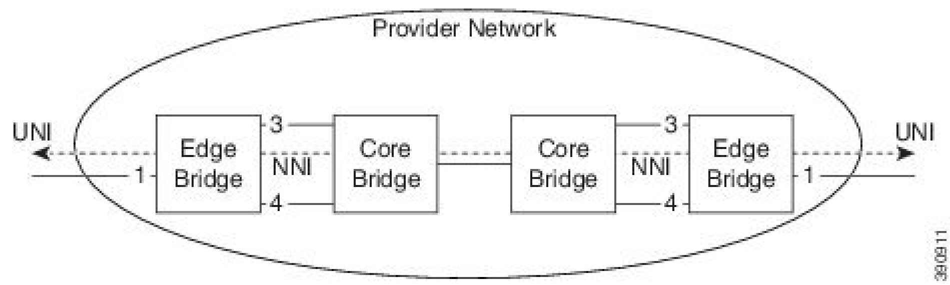
The following diagrams show a Provider Network offering various types of E-LAN between two UNIs.

Figure 5: An Unprotected Ethernet-LAN



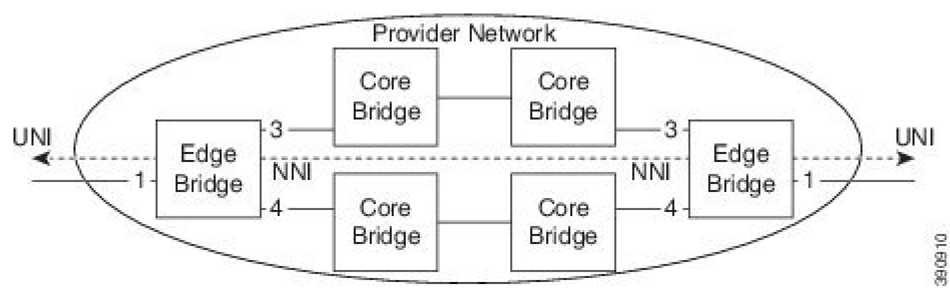
The following diagram shows an ethernet private (EP) line with 1-to-1 port protection on the network-network interface (NNI) side. This setup requires more resources compared to the unprotected EP-Line, because rules must be added for both NNI ports.

Figure 6: Port Protected E-LAN



The following diagram shows an ethernet LAN with ring protection on the network-network interface (NNI) side. The resource consumption is similar to the port protection scenario, because rules are added for each NNI port.

Figure 7: Ring-Protected E-LAN



The following sections describe how to configure the Edge Bridges.

Configuring EVC for E-LAN Between Two UNI and NNI Ports With Double Tag on the NNI Port

For more information on configuring EVC, see the Configuring Ethernet Virtual Circuit section.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addEVC evcConfiguration instance 9
UCS(ProvisionEVC)# addEVC evcConfiguration internal_vid 400
UCS(ProvisionEVC)# addEVC evcConfiguration nni_vid 400
UCS(ProvisionEVC)# addEVC evcConfiguration learning enable
UCS(ProvisionEVC)# addEVC evcConfiguration nni_ports GigabitEthernet_6_NNI enable
UCS(ProvisionEVC)# addEVC evcConfiguration policer_id 1
UCS(ProvisionEVC)# addEVC review
UCS(ProvisionEVC)# addEVC commit
```

Configuring EVC for E-LAN

For more information on configuring EVC, see the Configuring Ethernet Virtual Circuit section.

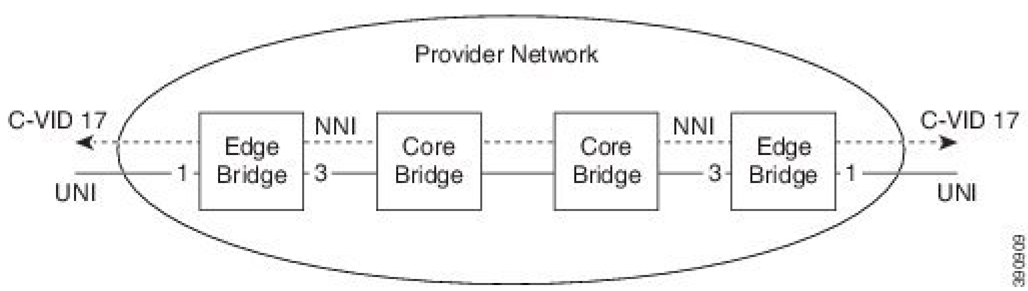
Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addEVC evcConfiguration instance 9
UCS(ProvisionEVC)# addEVC evcConfiguration internal_vid 400
UCS(ProvisionEVC)# addEVC evcConfiguration nni_vid 400
UCS(ProvisionEVC)# addEVC evcConfiguration learning enable
UCS(ProvisionEVC)# addEVC evcConfiguration nni_ports GigabitEthernet_6_NNI enable
UCS(ProvisionEVC)# addEVC evcConfiguration policer_id 1
UCS(ProvisionEVC)# addEVC review
UCS(ProvisionEVC)# addEVC commit
```

Ethernet Virtual Private Line

The following diagram shows an unprotected ethernet virtual private line (EVP-Line) forwarding frames with C-VID = 17 between the user-network interface (UNI) ports.

Figure 8: Unprotected EVP-Line



Thie following section describes the configuration of the EVPL service between the UNI and NNI ports.

Configuring ECE For EVPL Service

For more information on configuring ECE, see the Configuring EVC Control Entry section.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addECE ece_configuration ece_id 6
UCS(ProvisionEVC)# addECE ece_configuration control actions evc_id specific 8

UCS(ProvisionEVC)# addECE ece_configuration control actions policer_id specific 1
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match uni_ports
GigabitEthernet_3_UNI enable
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_type
 c_tagged
UCS(ProvisionEVC)# addECE ece_configuration control ingress_match outer_tag_match match_fields
 vlan_id_filter range 300-350
UCS(ProvisionEVC)# addECE review
UCS(ProvisionEVC)# addECE commit
```

Note The above ECE rule allows all VLANs ranging from 300 to 350. However, if you need to filter specific VLANs then you must create individual ECE rules. For more information, see Configuring ECE Rule 1.

Configuring EVC For EVPL Service

For more information on configuring EVC, see the Configuring Ethernet Virtual Circuit section.

Example

```
UCS# ProvisionEVC
UCS(ProvisionEVC)# addEVC evcConfiguration instance 8
UCS(ProvisionEVC)# addEVC evcConfiguration internal_vid 200
UCS(ProvisionEVC)# addEVC evcConfiguration nni_vid 200
UCS(ProvisionEVC)# addEVC evcConfiguration learning enable
UCS(ProvisionEVC)# addEVC evcConfiguration nni_ports GigabitEthernet_5_NNI enable
UCS(ProvisionEVC)# addEVC evcConfiguration policer_id 1
UCS(ProvisionEVC)# addEVC review
UCS(ProvisionEVC)# addEVC commit
```

Other Commands For EVC Configuration

Clearing EVC Statistics

{**all** | **ece** | | }

Using the Default Configuration

default

```
Switch(ProvisionEVC)# default
```

Note This command resets all configuration to default values.

Deleting Configuration

Use this command to delete the ECE configuration.

Use this command to delete the EVC configuration.

deleteEVC deleteEVCrequest

Use this command to delete the EVC Policer request.

Editing Configuration

Use this command to edit the ECE configuration.

{ | **update** {**class** | **direction** | | | | | | }}

Use this command to edit the EVC configuration.

editEVCConfiguration evcupdateConfiguration {**instance** *instance_id* | **update** { | **learning** | | | }}

Enabling/Disabling/Modifying EVC Policer

Use this command to enable the EVC Policer.

Use this command to disable the EVC Policer.

Use this command to modify the EVC Policer.

Fetching EVC and ECE Configuration

Use the following commands to fetch the ECE configuration:

- **getECEBlankForm getECEForm**
-
- **getECEConfiguration getECEconfig**

Use the following commands to fetch the EVC configuration:

- **getEVCBlankForm getEVCForm**
-
- **getEVCConfiguration getEVCconfig**

CHAPTER 9

Configuring Switch Ports

This document describes various virtual LAN (VLAN) configuration you can perform on the switch ports, such as creating layer 2 and layer 3 VLANs, creating VLAN mapping, VLAN translation groups, and modifying software ports.

How To Configure Switch Ports

Creating Layer 2 VLANs

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**createVlanCommand createVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 100-4095`	Creates the VLAN list. The valid values are from 1 to 4095.

	Command or Action	Purpose
Step 4	**createVlanCommand review** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand review`	Displays the createVlanCommand configuration.
Step 5	**createVlanCommand commit** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand commit`	Sends the createVlanCommand configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 100-4095
UCS(ProvisionPortVlanPortType)# createVlanCommand review
Commands in queue:
 createVlanCommand createVlanReq vlan_list 100-4095
UCS(ProvisionPortVlanPortType)# createVlanCommand commit
 Vlan Creation Commit Success!!!
```

Verifying Layer 2 VLAN Configuration

The following is a sample output of the command that displays in brief the configured layer 2 VLAN list:

```
UCS(ProvisionPortVlanPortType)# showVlans showVlanRequest brief
```

```
Commands in queue:

showVlans showVlanRequest brief
```

```
UCS(ProvisionPortVlanPortType)# showVlans commit
```

```
Configured Vlan List:
    1

 Show Vlans Commit Success!!!
```

Deleting Layer 2 VLANs

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**deleteVlanCommand deleteVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanCommand deleteVlanReq vlan_list 100-4095`	Deletes the VLAN list.
Step 4	**deleteVlanCommand review** **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanCommand review`	Displays the deleteVlanCommand configuration.
Step 5	**deleteVlanCommand commit** **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanCommand commit`	Sends the deleteVlanCommand configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# deleteVlanCommand deleteVlanReq vlan_list 100-4095
UCS(ProvisionPortVlanPortType)# deleteVlanCommand review
Commands in queue:
 deleteVlanCommand deleteVlanReq vlan_list 100-4095
UCS(ProvisionPortVlanPortType)# deleteVlanCommand commit
DeleteVlanCommand_Output.deleteVlanResp = 0

 Vlan Deletion Commit Success!!!
```

Creating Layer 3 VLANs

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**createIntVlan createIntVlanReq***vlan_id* **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan_id 22`	Creates the interface VLAN list.

Step 4	**createIntVlan createIntVlanReq** {**address** {**ipv4** {**dhcp** | **ipv4_address**} | **ipv6ipv6_address** } | *vlan_id*
	
	Example:
	`UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address address 22.22.22.3`
	`UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address mask 255.255.255.0`
	`UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv6 ipv6_address 2001:4::1/64`
		Creates the interface VLAN on the specified IPv4 or IPv6 address, or VLAN ID.

Step 5	**createIntVlan review** **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan review`	Displays the createIntVlan configuration.
Step 6	**createIntVlan commit** **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan commit`	Sends the createIntVlan configuration to the NID.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan_Id 22
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address
```

```
address 22.22.22.3
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 ipv4_address
mask 255.255.255.0
UCS(ProvisionPortVlanPortType)# createIntVlan review

Commands in queue:
 createIntVlan createIntVlanReq vlan_id 22
 createIntVlan createIntVlanReq address ipv4 ipv4_address address 22.22.22.3
 createIntVlan createIntVlanReq address ipv4 ipv4_address mask 255.255.255.0

UCS(ProvisionPortVlanPortType)# createIntVlan commit

CreateIntVlan_Output.createIntVlanResp = 0

 Create Interface Vlan Commit Success!!!

UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan_Id 22
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv6 ipv6_address
2001:4::1/64
UCS(ProvisionPortVlanPortType)# createIntVlan review

Commands in queue:
 createIntVlan createIntVlanReq vlan_id 22
 createIntVlan createIntVlanReq address ipv6 ipv6_address 2001:4::1/64

UCS(ProvisionPortVlanPortType)# createIntVlan commit

CreateIntVlan_Output.createIntVlanResp = 0

 Create Interface Vlan Commit Success!!!
```

Creating Layer 3 VLANs With Dynamic IP Address

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**createIntVlan createIntVlanReq deleteVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan_id 23`	Creates the interface VLAN on the specified VLAN.
Step 4	**createIntVlan createIntVlanReq address ipv4 dhcp** **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 dhcp`	Creates the interface VLAN on the specified address.

	Command or Action	Purpose
Step 5	**createIntVlan review** **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan review`	Displays the createIntVlan configuration.
Step 6	**createIntVlan commit** **Example:** `UCS(ProvisionPortVlanPortType)# createIntVlan commit`	Sends the createIntVlan configuration to the NID.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq vlan_Id 23
UCS(ProvisionPortVlanPortType)# createIntVlan createIntVlanReq address ipv4 dhcp
UCS(ProvisionPortVlanPortType)# createIntVlan review

Commands in queue:
 createIntVlan createIntVlanReq vlan_id 23
 createIntVlan createIntVlanReq address ipv4 dhcp

UCS(ProvisionPortVlanPortType)# createIntVlan commit

CreateIntVlan_Output.createIntVlanResp = 0

 Create Interface Vlan Commit Success!!!
```

Verifying Layer 3 VLANs With Dynamic IP Address

The following is a sample output to display the layer 3 VLANs configured with a dynamic IP address:

```
UCS(ProvisionPortVlanPortType)# showIntVlan showIntVlanReq vlan_list 23
```

```
Commands in queue:

showIntVlan showIntVlanReq vlan_list 23
```

```
UCS(ProvisionPortVlanPortType)# showIntVlan commit
```

```
ShowIntVlan_Output.showIntVlanResp.vlan_list[0].vlan_id = 23
ShowIntVlan_Output.showIntVlanResp.vlan_list[0].Link = 'LINK:
00-3a-99-fd-4d-05 Mtu:1500'
ShowIntVlan_Output.showIntVlanResp.vlan_list[0].dhcp = 'DHCP'
ShowIntVlan_Output.showIntVlanResp.vlan_list[0].ipv6_address = 'IPv6
Address not configured'

 Show Interface Vlan Commit Success!!!
```

Deleting Layer 3 VLANs

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**deleteIntVlan deleteIntVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# deleteIntVlan deleteIntVlanReq vlan_list 23`	Deletes the VLAN list on the interface.
Step 4	**deleteIntVlan review** **Example:** `UCS(ProvisionPortVlanPortType)# deleteIntVlan review`	Displays the deleteIntVlan configuration.
Step 5	**deleteIntVlan commit** **Example:** `UCS(ProvisionPortVlanPortType)# deleteIntVlan commit`	Sends the deleteIntVlan configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits from the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# deleteIntVlan deleteIntVlanReq vlan_list 23
UCS(ProvisionPortVlanPortType)# deleteIntVlan review

Commands in queue:
 deleteIntVlan deleteIntVlanReq vlan_list 23

UCS(ProvisionPortVlanPortType)# deleteIntVlan commit
DeleteIntVlan_Output.deleteIntVlanResp = 0

 Delete Interface Vlan Commit Success!!!
```

Creating a VLAN Translation Group

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**createVlanTranslationGroup createVlanTranslationGroupReq group_id** *group_id* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup createVlanTranslationGroupReq group_Id 3`	Creates the VLAN Translation group ID.
Step 4	**createVlanTranslationGroup createVlanTranslationGroupReq** {**vlan_id***vlan_id* | **vlan_list** *vlan_list*} **Example:** `UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup createVlanTranslationGroupReq vlan_id 22`	Creates the VLAN translation • **vlan_id**—Sets the VLAN ID on which translation occurs. • **vlan_list**—Sets the VLAN list that needs to be translated.
Step 5	**createVlanTranslationGroup review** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup review`	Displays the createVlanTranslationGroup configuration.
Step 6	**createVlanTranslationGroup commit** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup commit`	Sends the createVlanTranslationGroup configuration to the NID.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup createVlanTranslationGroupReq
```

```
group_Id 3
UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup createVlanTranslationGroupReq
vlan_id 22
UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup createVlanTranslationGroupReq
vlan_list 100,101,102
UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup review

Commands in queue:
 createVlanTranslationGroup createVlanTranslationGroupReq group_Id 3
 createVlanTranslationGroup createVlanTranslationGroupReq vlan_id 22
 createVlanTranslationGroup createVlanTranslationGroupReq vlan_list 100,101,102

UCS(ProvisionPortVlanPortType)# createVlanTranslationGroup commit

CreateVlanTranslationGroup_Output.createVlanTranslationGroupResp = 0

 Create VlanTranslation Commit Success!!!
```

Deleting VLAN Translation Groups

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**deleteVlanTranslation deleteVlanTranslationGroupReqgroup_id***group_id* **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanTranslation deleteVlanTranslationGroupReq group_id 3`	Deletes the specified VLAN Translation group id.
Step 4	**deleteVlanTranslation deleteVlanTranslationGroupReqdeleteVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanTranslation deleteVlanTranslationGroupReq vlan_list 2,3`	Deletes the specified VLAN Translation VLAN list.
Step 5	**deleteVlanTranslation review** **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanTranslation review`	Displays the deleteVlanTranslation configuration.

	Command or Action	Purpose
Step 6	**deleteVlanTranslation commit** **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanTranslation commit`	Sends the deleteVlanTranslation configuration to the NID.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# deleteVlanTranslation deleteVlanTranslationGroupReq group_id
 3
UCS(ProvisionPortVlanPortType)# deleteVlanTranslation deleteVlanTranslationGroupReq vlan_list
 100,101,102
UCS(ProvisionPortVlanPortType)# deleteVlanTranslation review

Commands in queue:
 deleteVlanTranslation deleteVlanTranslationGroupReq group_id 3
 deleteVlanTranslation deleteVlanTranslationGroupReq vlan_list 100,101,102

UCS(ProvisionPortVlanPortType)# deleteVlanTranslation commit

DeleteVlanTranslation_Output.deleteVlanTranslationGroupResp = 0

 Delete VlanTranslation Commit Success!!!
```

Verifying VLAN Translation Group

The following is a sample output of the command to verify the VLAN translation group configuration:

```
UCS(ProvisionPortVlanPortType)# showVlanTranslation showVlanTranslationGroupReq
      all
```

```
Commands in queue:

showVlanTranslation showVlanTranslationGroupReq all
```

```
UCS(ProvisionPortVlanPortType)# showVlanTranslation commit
```

```
ShowVlanTranslation_Output.showVlanTranslationGroupResp[0].group_id = 3
ShowVlanTranslation_Output.showVlanTranslationGroupResp[0].vlan_list =
100
ShowVlanTranslation_Output.showVlanTranslationGroupResp[0].transvlan_id
= 22
ShowVlanTranslation_Output.showVlanTranslationGroupResp[1].group_id = 3
ShowVlanTranslation_Output.showVlanTranslationGroupResp[1].vlan_list =
101
ShowVlanTranslation_Output.showVlanTranslationGroupResp[1].transvlan_id
= 22
```

```
ShowVlanTranslation_Output.showVlanTranslationGroupResp[2].group_id = 3
ShowVlanTranslation_Output.showVlanTranslationGroupResp[2].vlan_list =
102
ShowVlanTranslation_Output.showVlanTranslationGroupResp[2].transvlan_id
= 22

 Show VlanTranslation Commit Success!!!
```

Creating VLAN Mapping

Before You Begin

VLAN Mapping should be created for the VLAN translation group, and the mapping interface should be bound to that translation group.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**createVlanMapping createVlanMappingReq group_id** *group_id* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanMapping createVlanMappingReq group_id 3`	Creates the VLAN mapping group ID.
Step 4	**createVlanMapping createVlanMappingReq interface** *interface_id* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanMapping createVlanMappingReq interface 5`	Creates the VLAN mapping on the specified interface.
Step 5	**createVlanMapping review** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanMapping review`	Displays the createVlanMapping configuration.
Step 6	**createVlanMapping commit** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanMapping commit`	Sends the createVlanMapping configuration to the NID.

	Command or Action	Purpose
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createVlanMapping createVlanMappingReq group_id 3
UCS(ProvisionPortVlanPortType)# createVlanMapping createVlanMappingReq interface 5
UCS(ProvisionPortVlanPortType)# createVlanMapping review

Commands in queue:
 createVlanMapping createVlanMappingReq group_id 3
 createVlanMapping createVlanMappingReq interface 5

UCS(ProvisionPortVlanPortType)# createVlanMapping commit
CreateVlanMapping_Output.createVlanMappingResp = 0

 Create VlanMapping Commit Success!!!
```

Deleting VLAN Mapping

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**deleteVlanMapping deleteVlanMappingReq interface** *interface_id* **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanMapping deleteVlanMappingReq interface 5`	Deleted VLAN mapping for the specified interface.
Step 4	**deleteVlanMapping review** **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanMapping review`	Displays the deleteVlanMapping configuration.

	Command or Action	Purpose
Step 5	**deleteVlanMapping commit** **Example:** `UCS(ProvisionPortVlanPortType)# deleteVlanMapping commit`	Sends the deleteVlanMapping configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# deleteVlanMapping deleteVlanMappingReq interface 5
UCS(ProvisionPortVlanPortType)# deleteVlanMapping review

Commands in queue:
 deleteVlanMapping deleteVlanMappingReq interface 5

UCS(ProvisionPortVlanPortType)# deleteVlanMapping commit
DeleteVlanMapping_Output.deleteVlanMappingResp = 0

 Delete Vlan Mapping Commit Success!!!
```

Modifying Switch Ports

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**modifySwPort modifySWPortConfig interface***interface_id* **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interaface 4`	Modifies the switchport configuration on the defined interface.
Step 4	**modifySwPort-v2 modifySWConfig** { **interface** | **intf-description** | **mode**}	• **interface—**Selects the interface to be configured.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 4`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig intf-description`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode`
		• **intf-description—**Specifies the description of the interface.
		• **mode—**Displays the mode of operation.

Step 5	**modifySwPort modifySWPortConfig mode access vlan** *vlan_id* **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 2`	Sets the mode to ACCESS, and assigns a VLAN.
Step 6	**modifySwPort modifySWPortConfig mode trunk** {**allowed vlan** {**add** {**all** | **vlan_list** *vlan_list* } | **remove** {**all** | **vlan_list** *vlan_list* }} | {**native vlan** *vlan_list* } **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 1-5`	Sets the mode to TRUNK. • **allowed—**Sets the allowed VLAN characteristics when interface is in trunk mode. • **add—**Adds either all VLANs or specified VLANs to the current list. • **remove—**Removes either all VLANs or specified VLANs from the current list. • *vlan_id*—The VLAN ID. The valid values are from 0 to 4095.
Step 7	**modifySwPort review** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort review`	Displays the modifySwPort configuration.
Step 8	**modifySwPort commit** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort commit`	Sends the modifySwPort configuration to the NID.
Step 9	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interaface 4
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 2
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 200-225
UCS(ProvisionPortVlanPortType)# modifySwPort review

Commands in queue:
```

```
 modifySwPort modifySWPortConfig interaface 4
 modifySwPort modifySWPortConfig mode trunk native vlan 2
 modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 200-225

UCS(ProvisionPortVlanPortType)# modifySwPort commit

ModifySwPort_Output.modifySwPortConfigResp = 0
 Modify SwitchPort Commit Success!!!
```

Note To configure the Switch Port mode as hybrid and the Port description, use **modifySwPort_v2**. In addition to the available parameters for **modifySwPort**, the following are the new parameters available:

- *hybrid* - Sets mode to HYBRID unconditionally.
- *intf_description description* - Configures interface description.

Example

The following example shows how to configure Switch Port mode as hybrid using **modifySwPort_v2**.

```
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig interaface 1
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig mode hybrid allowed vlan
remove vlan_list 1-100
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig mode hybrid port_type c_port
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig mode hybrid ingress_filtering
 enable
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig mode hybrid ingress_acceptance
 tagged
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig mode hybrid egress_tag all
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig mode hybrid native vlan 10

UCS(ProvisionPortVlanPortType)# modifySwPort_v2 review

Commands in queue:
        modifySwPort_v2 modifySWConfig interaface 1
        modifySwPort_v2 modifySWConfig mode hybrid allowed vlan remove vlan_list 1-100
        modifySwPort_v2 modifySWConfig mode hybrid port_type c_port
        modifySwPort_v2 modifySWConfig mode hybrid ingress_filtering enable
        modifySwPort_v2 modifySWConfig mode hybrid ingress_acceptance tagged
        modifySwPort_v2 modifySWConfig mode hybrid egress_tag all
        modifySwPort_v2 modifySWConfig mode hybrid native vlan 10

UCS(ProvisionPortVlanPortType)# modifySwPort_v2 commit
```

Example

The following example shows how to configure interface description using **modifySwPort_v2**.

```
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig interaface 1
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 modifySWConfig intf_description description
 connected_to_r1
UCS(ProvisionPortVlanPortType)# modifySwPort_v2 review

Commands in queue:
modifySwPort_v2 modifySWConfig interaface 1
modifySwPort_v2 modifySWConfig intf_description description connected_to_r1

UCS(ProvisionPortVlanPortType)# modifySwPort_v2 commit
```

Deleting Switch Ports

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**deleteSwPort deleteSwPortReq interface***interface_id* **Example:** `UCS(ProvisionPortVlanPortType)# deleteSwPort deleteSwPortReq interaface 5`	Deletes the switchport on the specified interface.
Step 4	**deleteSwPort deleteSwPortReq mode** {**access** | **trunk**} **Example:** `UCS(ProvisionPortVlanPortType)# deleteSwPort deleteSwPortReq mode access`	Deletes the switchport on the specified mode.
Step 5	**deleteSwPort review** **Example:** `UCS(ProvisionPortVlanPortType)# deleteSwPort review`	Displays the deleteSwPort configuration.
Step 6	**deleteSwPort commit** **Example:** `UCS(ProvisionPortVlanPortType)# deleteSwPort commit`	Sends the deleteSwPort configuration to the NID.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Example

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# deleteSwPort deleteSwPortReq interaface 5
UCS(ProvisionPortVlanPortType)# deleteSwPort deleteSwPortReq mode access
UCS(ProvisionPortVlanPortType)# deleteSwPort review

Commands in queue:
 deleteSwPort deleteSwPortReq interaface 5
 deleteSwPort deleteSwPortReq mode access

UCS(ProvisionPortVlanPortType)# deleteSwPort commit
DeleteSwPort_Output.deleteSwPortResp = 0
```

```
 Delete SwitchPort Commit Success!!!
```

Verifying Switch Port Details

The following is a sample output to verify all details of the switch ports:

```
UCS(ProvisionPortVlanPortType)# showSwPort showSwPortReq all all
```

```
Commands in queue:

showSwPort showSwPortReq all all
```

```
UCS(ProvisionPortVlanPortType)# showSwPort commit
```

```
ShowSwPort_Output.showSwPortResp.interface_list[0].name = 'GigabitEthernet
 1/1'
ShowSwPort_Output.showSwPortResp.interface_list[0].admin_mode = 'trunk'
ShowSwPort_Output.showSwPortResp.interface_list[0].access_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[0].trunk_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[0].trunk_members = '1-4095'
ShowSwPort_Output.showSwPortResp.interface_list[1].name = 'GigabitEthernet
 1/2'
ShowSwPort_Output.showSwPortResp.interface_list[1].admin_mode = 'trunk'
ShowSwPort_Output.showSwPortResp.interface_list[1].access_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[1].trunk_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[1].trunk_members = '1-4095'
ShowSwPort_Output.showSwPortResp.interface_list[2].name = 'GigabitEthernet
 1/3'
ShowSwPort_Output.showSwPortResp.interface_list[2].admin_mode = 'trunk'
ShowSwPort_Output.showSwPortResp.interface_list[2].access_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[2].trunk_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[2].trunk_members = '1-4095'
ShowSwPort_Output.showSwPortResp.interface_list[3].name = 'GigabitEthernet
 1/4'
ShowSwPort_Output.showSwPortResp.interface_list[3].admin_mode = 'trunk'
ShowSwPort_Output.showSwPortResp.interface_list[3].access_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[3].trunk_mode = 2
ShowSwPort_Output.showSwPortResp.interface_list[3].trunk_members = '1-4095'
ShowSwPort_Output.showSwPortResp.interface_list[4].name = 'GigabitEthernet
 1/5'
ShowSwPort_Output.showSwPortResp.interface_list[4].admin_mode = 'access'
ShowSwPort_Output.showSwPortResp.interface_list[4].access_mode = 120
ShowSwPort_Output.showSwPortResp.interface_list[4].trunk_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[4].trunk_members = '1-4095'
ShowSwPort_Output.showSwPortResp.interface_list[5].name = 'GigabitEthernet
 1/6'
ShowSwPort_Output.showSwPortResp.interface_list[5].admin_mode = 'access'
ShowSwPort_Output.showSwPortResp.interface_list[5].access_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[5].trunk_mode = 1
ShowSwPort_Output.showSwPortResp.interface_list[5].trunk_members = '1-4095'


 Show SwitchPort Commit Success!!!
```

CHAPTER 10

Configuring Spanning-Tree Protocol

The Cisco ME 1200 Series Carrier Ethernet Access Device supports Spanning-Tree Protocol (STP), and this chapter describes how to configure the STP on port-based VLANs. On the Cisco ME 1200 NID, the STP is enabled by default on physical interfaces.

Prerequisites for Configuring Spanning-Tree Protocol

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.

Information About Spanning-Tree Protocol

STP is a Layer 2 link management protocol that provides path redundancy while preventing loops in the network.

For a Layer 2 Ethernet network to function properly, only one active path can exist between any two stations. Multiple active paths among end stations cause loops in the network. If a loop exists in the network, end stations might receive duplicate messages. Devices might also learn end-station MAC addresses on multiple Layer 2 interfaces. These conditions result in an unstable network. Spanning-tree operation is transparent to end stations, which cannot detect whether they are connected to a single LAN segment or a switched LAN of multiple segments.

The STP uses a spanning-tree algorithm to select one switch of a redundantly connected network as the root of the spanning tree. The algorithm calculates the best loop-free path through a switched Layer 2 network by assigning a role to each port based on the role of the port in the active topology:

- Root—A forwarding port elected for the spanning-tree topology
- Designated—A forwarding port elected for every switched LAN segment
- Alternate—A blocked port providing an alternate path to the root bridge in the spanning tree
- Backup—A blocked port in a loopback configuration

The switch that has *all* of its ports as the designated role or the backup role is the root switch. The switch that has at least *one* of its ports in the designated role is called the designated switch.

Spanning tree forces redundant data paths into a standby (blocked) state. If a network segment in the spanning tree fails and a redundant path exists, the spanning-tree algorithm recalculates the spanning-tree topology and activates the standby path. Switches send and receive spanning-tree frames, called bridge protocol data units (BPDUs), at regular intervals. The switches do not forward these frames but use them to construct a loop-free path. BPDUs contain information about the sending switch and its ports, including switch and MAC addresses, switch priority, port priority, and path cost. Spanning tree uses this information to elect the root switch and root port for the switched network and the root port and designated port for each switched segment.

When two ports on a switch are part of a loop, the spanning-tree port priority and path cost settings control which port is put in the forwarding state and which is put in the blocking state. The spanning-tree port priority value represents the location of a port in the network topology and how well it is located to pass traffic. The path cost value represents the media speed.

Note The switch sends keepalive messages (to ensure the connection is up) only on interfaces that do not have small form-factor pluggable (SFP) modules.

Spanning-Tree Topology and BPDUs

The stable, active spanning-tree topology of a switched network is controlled by these elements:

- The unique bridge ID (switch priority and MAC address) associated with each VLAN on each switch.
- The spanning-tree path cost to the root switch.
- The port identifier (port priority and MAC address) associated with each Layer 2 STP-enabled interface.

When the switches in a network are powered up, each functions as the root switch. Each switch sends a configuration BPDU through all of its ports, or on the Cisco ME device, only through the STP-enabled ports. The BPDUs communicate and compute the spanning-tree topology. Each configuration BPDU contains this information:

- The unique bridge ID of the switch that the sending switch identifies as the root switch
- The spanning-tree path cost to the root
- The bridge ID of the sending switch
- Message age
- The identifier of the sending interface
- Values for the hello, forward delay, and max-age protocol timers

When a switch receives a configuration BPDU that contains superior information (lower bridge ID, lower path cost, and so forth), it stores the information for that port. If this BPDU is received on the root port of the switch, the switch also forwards it with an updated message to all attached LANs for which it is the designated switch.

If a switch receives a configuration BPDU that contains inferior information to that currently stored for that port, it discards the BPDU. If the switch is a designated switch for the LAN from which the inferior BPDU was received, it sends that LAN a BPDU containing the up-to-date information stored for that port. In this way, inferior information is discarded, and superior information is propagated on the network

For more information on BPDUs, see Configuring Optional Spanning-Tree features.

Spanning-Tree Interface States

Propagation delays can occur when protocol information passes through a switched LAN. As a result, topology changes can take place at different times and at different places in a switched network. When an STP port transitions directly from nonparticipation in the spanning-tree topology to the forwarding state, it can create temporary data loops. Interfaces must wait for new topology information to propagate through the switched LAN before starting to forward frames. They must allow the frame lifetime to expire for forwarded frames that have used the old topology.

Each Layer 2 interface on a switch using spanning tree exists in one of these states:

- Blocking—The interface does not participate in frame forwarding.
- Listening—The first transitional state after the blocking state when the spanning tree determines that the interface should participate in frame forwarding.
- Learning—The interface prepares to participate in frame forwarding.
- Forwarding—The interface forwards frames.
- Disabled—The interface is not participating in spanning tree because of a shutdown port, no link on the port, or no spanning-tree instance running on the port.

Configuring Port Priority

If a loop occurs, spanning tree uses the port priority when selecting a spanning-tree port to put into the forwarding state. You can assign higher priority values (lower numerical values) to ports that you want selected first and lower priority values (higher numerical values) to ones that you want selected last. If all spanning-tree ports have the same priority value, spanning tree puts the port with the lowest interface number in the forwarding state and blocks the other interfaces.

Configuring Path Cost

The spanning-tree path cost default value is derived from the media speed of an interface (port running spanning tree or port channel of multiple ports running spanning tree). If a loop occurs, spanning tree uses cost when selecting an interface to put in the forwarding state. You can assign lower cost values to interfaces that you want selected first and higher cost values that you want selected last. If all NNIs (or port channels) have the same cost value, spanning tree puts the interface with the lowest interface number in the forwarding state and blocks the other interfaces.

Configuring the Switch Priority of a VLAN

You can configure the switch priority and make it more likely that the switch is chosen as the root switch.

Admin Edge and Auto Edge

These two values control how a port is declared to be an edge port or not. An edge port, is a port which is not connected to a bridge. If auto edge is enabled, then the port determine whether a port is an edge port by registering for BPDUs, and if BPDUs are received on that port.

The admin edge determines what the port should start as being – edge or not.

Restricted Role and Restricted TCN

If restricted role is enabled, it causes the port not to be selected as Root Port for the Common and Internal Spanning Tree (CIST) or any Multiple Spanning Tree Instance (MSTI), even if it has the best spanning tree priority vector. Such a port is selected as an Alternate Port after the Root Port has been selected. If set, it can cause lack of spanning tree connectivity. It can be set by a network administrator to prevent bridges external to a core region of the network influence the spanning tree active topology, possibly because those bridges are not under the full control of the administrator. This feature is also known as Root Guard.

If restricted TCN is enabled, it causes the port not to propagate received topology change notifications and topology changes to other ports. If set it can cause temporary loss of connectivity after changes in a spanning tree's active topology as a result of persistently incorrect learned station location information. It is set by a network administrator to prevent bridges external to a core region of the network, causing address flushing in that region, possibly because those bridges are not under the full control of the administrator or the physical link state of the attached LANs transits frequently.

Understanding Spanning-Tree Modes and Protocols

The switch ports support the following spanning-tree modes and protocols:

- MSTP—This spanning-tree mode is based on the IEEE 802.1s standard. You can map multiple VLANs to the same spanning-tree instance, which reduces the number of spanning-tree instances required to support a large number of VLANs. The MSTP runs on top of the RSTP (based on IEEE802.1w), which provides for rapid convergence of the spanning tree by eliminating the forward delay and by quickly transitioning root ports and designated ports to the forwarding state. You cannot run MSTP without RSTP.
  The most common initial deployment of MSTP is in the backbone and distribution layers of a Layer 2 switched network. For more information, see Configuring MSTP.

Understanding MSTP Configuration

This section describes how to configure the Cisco implementation of the IEEE 802.1s Multiple STP (MSTP) on the Cisco ME 1200 NID. STP is enabled by default on switch ports.

Note The multiple spanning-tree (MST) implementation is a pre-standard implementation. It is based on the draft version of the IEEE standard.

The MSTP enables multiple VLANs to be mapped to the same spanning-tree instance, thereby reducing the number of spanning-tree instances needed to support a large number of VLANs. The MSTP provides for multiple forwarding paths for data traffic and enables load balancing. It improves the fault tolerance of the network because a failure in one instance (forwarding path) does not affect other instances (forwarding paths). The most common initial deployment of MSTP is in the backbone and distribution layers of a Layer 2 switched network. This deployment provides the highly available network required in a service-provider environment.

Both MSTP and RSTP improve the spanning-tree operation and maintain backward compatibility with equipment that is based on the (original) 802.1D spanning tree, with existing Cisco-proprietary Multiple Instance STP (MISTP).

Understanding MSTP

MSTP, which uses RSTP for rapid convergence, enables VLANs to be grouped into a spanning-tree instance, with each instance having a spanning-tree topology independent of other spanning-tree instances. This architecture provides multiple forwarding paths for data traffic, enables load balancing, and reduces the number of spanning-tree instances required to support a large number of VLANs.

Multiple Spanning-Tree Regions

For the NID to participate in multiple spanning-tree (MST) instances, you must consistently configure the switches with the same MST configuration information. A collection of interconnected NIDs that have the same MST configuration comprises an MST region. The MST configuration controls to which MST region each switch belongs. The configuration includes the name of the region, the revision number, and the MST VLAN-to-instance assignment map. You configure the NID for a region by using the global configuration command, after which the NID enters the MST configuration mode. From this mode, you can map VLANs to an MST instance by using the instance MST configuration command, specify the region name by using the name MST configuration command, and set the revision number by using the revision MST configuration command. A region can have one member or multiple members with the same MST configuration; each member must be capable of processing RSTP bridge protocol data units (BPDUs). There is no limit to the number of MST regions in a network, but each region can support up to 65 spanning-tree instances. You can assign a VLAN to only one spanning-tree instance at a time.

IST, CIST, and CST

The MSTP establishes and maintains two types of spanning trees, IST and CIST:

- An internal spanning tree (IST), which is the spanning tree that runs in an MST region. Within each MST region, the MSTP maintains multiple spanning-tree instances. Instance 0 is a special instance for a region, known as the internal spanning tree (IST). All other MST instances are numbered from 1 to 4094.The IST is the only spanning-tree instance that sends and receives BPDUs; all of the other spanning-tree instance information is contained in M-records, which are encapsulated within MSTP BPDUs. Because the MSTP BPDU carries information for all instances, the number of BPDUs that need to be processed by a switch to support multiple spanning-tree instances is significantly reduced.
  All MST instances within the same region share the same protocol timers, but each MST instance has its own topology parameters, such as root switch ID, root path cost, and so forth. By default, all VLANs are assigned to the IST. An MST instance is local to the region; for example, MST instance 1 in region A is independent of MST instance 1 in region B, even if regions A and B are interconnected.

- A common and internal spanning tree (CIST), which is a collection of the ISTs in each MST region, and the common spanning tree (CST) that interconnects the MST regions and single spanning trees. The spanning tree computed in a region appears as a subtree in the CST that encompasses the entire switched domain. The CIST is formed as a result of the spanning-tree algorithm running between switches that support the IEEE 802.1w, IEEE 802.1s, and IEEE 802.1D protocols. The CIST inside an MST region is the same as the CST outside a region.

For information regarding *Operations Within an MST Region*, *Operations Between MST Regions*, *IEEE 802.1s Terminology*, see Configuring MSTP.

Hop Count

The IST and MST instances do not use the message-age and maximum-age information in the configuration BPDU to compute the spanning-tree topology. Instead, they use the path cost to the root and a hop-count mechanism similar to the IP time-to-live (TTL) mechanism.

By using the global configuration command, you can configure the maximum hops inside the region and apply it to the IST and all MST instances in that region. The hopcount achieves the same result as the message-age information (trigger a reconfiguration). The root switch of the instance always sends a BPDU (or M-record) with a cost of 0 and the hop count set to the maximum value. When a switch receives this BPDU, it decrements the received remaining hop count by one and propagates this value as the remaining hop count in the BPDUs it generates. When the count reaches zero, the switch discards the BPDU and ages the information held for the port. The message-age and maximum-age information in the RSTP portion of the BPDU remain the same throughout the region, and the same values are propagated by the region's designated ports at the boundary.

Understanding RSTP

The RSTP takes advantage of point-to-point wiring and provides rapid convergence of the spanning tree. Reconfiguration of the spanning tree can occur in less than 1 second (in contrast to 50 seconds with the default settings in the IEEE 802.1D spanning tree), which is critical for networks carrying delay-sensitive traffic such as voice and video.

Understanding BPDU Guard and BPDU Filtering

BPDU Guard

The BPDU guard feature can be globally enabled on the switch or can be enabled per interface, but the feature operates with some differences.

At the global level, you enable BPDU guard on Port Fast-enabled STP ports by using the default global configuration command. Spanning tree shuts down STP ports that are in a Port Fast-operational state if any BPDU is received on those ports. In a valid configuration, Port Fast-enabled STP ports do not receive BPDUs. Receiving a BPDU on a Port Fast-enabled port signals an invalid configuration, such as the connection of an unauthorized device, and the BPDU guard feature puts the interface in the error-disabled state.

At the interface level, you enable BPDU guard on any STP port by using the interface configuration command without also enabling the Port Fast feature. When the STP port receives a BPDU, it is put in the error-disabled state. The BPDU guard feature provides a secure response to invalid configurations because you must manually put the interface back in service. Use the BPDU guard feature in a service-provider network to prevent an access port from participating in the spanning tree. You can enable the BPDU guard feature for the entire switch or for an interface.

BPDU Filtering

The BPDU filtering feature can be globally enabled on the switch or can be enabled per interface, but the feature operates with some differences.

At the global level, you can enable BPDU filtering on Port Fast-enabled STP ports by using the default global configuration command. This command prevents interfaces that are in a Port Fast-operational state from sending or receiving BPDUs. The interfaces still send a few BPDUs at link-up before the switch begins to filter outbound BPDUs. You should globally enable BPDU filtering on a switch so that hosts connected to these ports do not receive BPDUs. If a BPDU is received on a Port Fast-enabled STP port, the interface loses its Port Fast-operational status, and BPDU filtering is disabled.

At the interface level, you can enable BPDU filtering on any STP port by using the interface configuration command without also enabling the Port Fast feature. This command prevents the interface from sending or receiving BPDUs.

Note Enabling BPDU filtering on an STP port is the same as disabling spanning tree on it and can result in spanning-tree loops.

You can enable the BPDU filtering feature for the entire NID or for an STP port.

For more information on BPDUs, see Understanding BPDUs.

How to Configure Spanning-Tree Protocol

Configuring Spanning-tree Aggregation Port Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.

Step 3	**setStpaggConfig stpAggrConfig** {**auto-edge** {**enable** | **disable**} | **bpdu-guardbpdu-guard** {**enable** | **disable**} | **edge** {**enable** | **disable**} | **link-type** {**auto** | **point-to** | **shared**} {**enable** | **disable**} | **mst instance** *instance-id* {**active** {**enable** | **disable**} | **cost** {**auto** | **cost-range***cost_range*} | **port-priority***port_priority* } |**restricted-role** {**enable** | **disable**} | **restricted-tcn** {**enable** | **disable**}}
	
	Example:
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig auto-edge enable`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig bpdu-guard disable`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig edge disable`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig link-type auto enable`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig link-type point-to disable`
		Configures the spanning-tree port configuration:
		• **stpPortConfig**—Sets the spanning-tree port configuration.
		• **auto-edge**—Detects the auto-edge status.
		◦ **enable**—Enables the auto-edge
		◦ **disable**—Disables the auto-edge
		• **bpdu-guard**—Configures the BPDU guard.
		◦ **enable**—Enables the bpdu-guard
		◦ **disable**—Disables the bpdu-guard
		• **edge**—Configures the edge port.
		◦ **enable**—Enables the edge.
		◦ **disable**—Disables the edge.
		• **link-type**—Configures the port link-type.
		◦ **auto**—Configures the link-type as auto.

	Command or Action	Purpose

	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig link-type shared disable`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance instance-id 1`
	
	`setStpAggConfig stpAggrConfig mst occur instance active enable`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance port-priority 2`
	
	`setStpAggConfig stpAggrConfig mst occur instance cost auto`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig restricted-role enable`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig restricted-tcn disable`
		◦ **enable**—Enables the link-type as auto.
		◦ **disable**—Disables the link-type as auto.
		◦ **point-to**—Forces the link-type as point-to-point.
		◦ **enable**—Enables the link-type as point-to.
		◦ **disable**—Disables the link-type as point-to.
		◦ **shared**—Forces the link-type as shared.
		◦ **enable**—Enables the link-type as shared.
		◦ **disable**—Disables the link-type as shared.
		• **mst**—Configures the STP bridge instance.
		◦ *instance*—Instance. The range is from 0 to 7 where CIST=0, MST2=1 and so on.
		◦ **active**—Adds or removes an instance.
		◦ **enable**—Enables the mst instance as active.
		◦ **disable**—Disables the mst instance as active.
		◦ **cost**—Configures the STP cost for the port.
		◦ **auto**—Uses auto cost.
		◦ *cost-range*—Cost. The range is from 1-200000000.
		◦ *port-priority*—STP priority of the port. The range is from 0 to 240.
		• **restricted-role**—Configures the port role. It is restricted (and never a root port).
		◦ **enable**—Enables the port as having restricted role.
		◦ **disable**—Disables the port as having restricted role.
		• **restricted-tcn**—Restricts the topology change notifications.
		◦ **enable**—Enables the restricted TCN.
		◦ **disable**—Disables the restricted TCN.

	Command or Action	Purpose
Step 4	**setStpAggConfig review** **Example:** `UCS(ProvisionStpPortType)# setStpAggConfig review`	Displays the configuration.
Step 5	**setstpPortConfig commit** **Example:** `UCS(ProvisionStpPortType)# setStpAggConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig auto-edge enable
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig bpdu-guard disable
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig edge disable
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig link-type auto enable
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig link-type point-to disable
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig link-type shared disable
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance instance-id 1

        setStpAggConfig stpAggrConfig mst occur instance active enable
UCS(ProvisionStpPortType)#         setStpAggConfig stpAggrConfig mst occur instance
port-priority 2

        setStpAggConfig stpAggrConfig mst occur instance cost auto
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst instance 0 cost cost-range 1
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst instance 0 port-priority 1
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig restricted-role enable
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig restricted-tcn disable
UCS(ProvisionStpPortType)# setStpAggConfig review

Commands in queue:

    setStpAggConfig stpAggrConfig auto-edge enable
    setStpAggConfig stpAggrConfig bpdu-guard disable
    setStpAggConfig stpAggrConfig edge disable
    setStpAggConfig stpAggrConfig link-type auto enable
    setStpAggConfig stpAggrConfig link-type point-to disable
    setStpAggConfig stpAggrConfig link-type shared disable
    setStpAggConfig stpAggrConfig mst occur instance instance-id 1
    setStpAggConfig stpAggrConfig mst occur instance active enable
    setStpAggConfig stpAggrConfig mst occur instance port-priority 2
    setStpAggConfig stpAggrConfig mst occur instance cost auto
    setStpAggConfig stpAggrConfig mst instance 0 cost cost-range 1
    setStpAggConfig stpAggrConfig mst instance 0 port-priority 1
    setStpAggConfig stpAggrConfig restricted-role enable
    setStpAggConfig stpAggrConfig restricted-tcn disable

UCS(ProvisionStpPortType)# setStpAggConfig commit

    SetStpAggConfig Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Viewing Spanning-Tree Aggregation Port Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.
Step 3	**getstpaggConfig getStpAggConfigRequest** **Example:** `UCS(ProvisionStpPortType)# getstpaggConfig getStpAggConfigRequest`	Displays the configuration.
Step 4	**getstpaggConfig commit** **Example:** `UCS(ProvisionStpPortType)# setStpAggConfig commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# getstpaggConfig getStpAggConfigRequest

    stpAggrConfig.auto_edge = false
    stpAggrConfig.bpdu_guard = true
    stpAggrConfig.edge = false
    stpAggrConfig.link_type.t = 1
    stpAggrConfig.link_type.u.auto_ = false
    stpAggrConfig.mst.instance[0].active = true
    stpAggrConfig.mst.instance[0].cost.t = 1
    stpAggrConfig.mst.instance[0].cost.u.cost_range = 1
    stpAggrConfig.mst.instance[0].port_priority = 1
    stpAggrConfig.restricted_role = false
    stpAggrConfig.restricted_tcn = true

UCS(ProvisionStpPortType)# getstpaggConfig commit

    GetstpaggConfig Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Configuring Spanning-Tree Global Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.

Step 3	**setStpglobalConfig stpGlobalConfig** {**edge** {**bpdu-filter** | **bpdu-guard**} {**enable** | **disable**} | **mode** {**mstp** | **rstp** | **stp**} {**enable** | **disable**} | **mst** {**forward-time** *Fwdtime* | **instance** *instance* {**active** {**enable** | **disable**} | **priority** *Prio* | **vlan** *WORD*} | **max-age** *Maxage* | **max-hops** *Maxhops* | **name** *Name* | **revision** *Revision* } | **port-number** *Port number* {**enable** | **disable**} | **recovery** *Interval* | **transmit** *hold-count* }
	
	Example:
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig edge bpdu-guard enable`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mode mstp enable`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst forward-time 4`
	
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance instance-id 1`
	`        setStpAggConfig stpAggrConfig mst occur instance active enable`
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance port-priority 2`
	`        setStpAggConfig stpAggrConfig mst occur instance cost auto`
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst instance 0 vlan 1`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst max-age 30`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst max-hops 30`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst name ciscoNID123`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst revision 1111`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig port-number 1 enable`
		Configures the spanning-tree global configuration.
		• **stpGlobalConfig—**Sets the spanning-tree global configuration.
		• **edge—**Configures the edge ports.
		  ◦ **bpdu-filter—**Enables or disables the BPDU filter (stop BPDU tx/rx).
		  ◦ **bpdu-guard—**Enables or disables the BPDU guard.
		• **mode—**Configures the STP protocol mode.
		  ◦ **mstp—**Enables or disables the Multiple Spanning Tree (802.1s).
		  ◦ **rstp—**Enables or disables the Rapid Spanning Tree (802.1w)
		  ◦ **stp—**Enables or disables the Spanning Tree (802.1D).
		• **mst—**Configures the STP bridge instance.
		  ◦ *Fwdtime***—**Forward time. The range is from 4 to 30 seconds.
		  ◦ *instance***—**Instance. The range is from 0 to 7 where CIST=0, MST2=1 and so on.
		    ◦ **active—**Enables or disables the instance.
		    ◦ *Prio* **—**Specifies the priority. The range is from 0 to 61440 seconds. The range should be given in the sets of (0, 4096, 8192...) and so on.

	Command or Action	Purpose

	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig recovery interval 3000`
	
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig transmit hold-count 10`
	
	Note If the spanning-tree mode is STP or RSTP, and if the priority for the software needs to be changed, you can change using mst instance 0 and priority.
		◦ *WORD*—VLAN range.
		
		◦ *Maxage*—Maximum age. The range is from 6 to 40 seconds.
		
		◦ *Maxhops*—Maximum hops. The range is from 6 to 40 hop counts.
		
		◦ *Name*—Name of the bridge. You can use 32 characters to define.
		
		◦ *Revision*—Revision. The range is from 0-65535 revisions.
		
		• **port-number**—Configures the port number in the range from 1 to 6.
		
		◦ *Port number*—Port number. The range is from 1 to 6.
		
		◦ **disable**—Disables the port-number.
		
		◦ **enable**—Enables the port-number.
		
		• **recovery**—Configures the error recovery timeout.
		
		◦ *Interval*—Interval. The range is from 30-86400 seconds.
		
		• **transmit**—Configures the BPDUs to transmit.
		
		◦ *hold-count*—Maximum number of transmit BPDUs per second. The range is from 1 to 10 seconds.

Step 4	**setStpGlobalConfig review** **Example:** `UCS(ProvisionStpPortType)# setStpGlobalConfig review`	Displays the configuration.
Step 5	**setStpGlobalConfig commit** **Example:** `UCS(ProvisionStpPortType)# setStpGlobalConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig edge bpdu-guard enable
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mode mstp enable
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst forward-time 4
UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance instance-id 1

        setStpAggConfig stpAggrConfig mst occur instance active enable

UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance port-priority
2

        setStpAggConfig stpAggrConfig mst occur instance cost auto

UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst instance 0 vlan 1
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst max-age 30
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst max-hops 30
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst name myNID123
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig mst revision 1111
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig port-number 1 enable
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig recovery interval 3000
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig transmit hold-count 10
UCS(ProvisionStpPortType)# setStpGlobalConfig review

Commands in queue:

    setStpGlobalConfig stpGlobalConfig edge bpdu-guard enable
    setStpGlobalConfig stpGlobalConfig mode mstp enable
    setStpGlobalConfig stpGlobalConfig mst forward-time 4
    setStpAggConfig stpAggrConfig mst occur instance instance-id 1
    setStpAggConfig stpAggrConfig mst occur instance active enable
    setStpAggConfig stpAggrConfig mst occur instance port-priority 2
    setStpAggConfig stpAggrConfig mst occur instance cost auto
    setStpGlobalConfig stpGlobalConfig mst instance 0 vlan 1
    setStpGlobalConfig stpGlobalConfig mst max-age 30
    setStpGlobalConfig stpGlobalConfig mst max-hops 30
    setStpGlobalConfig stpGlobalConfig mst name myNID123
    setStpGlobalConfig stpGlobalConfig mst revision 1111
    setStpGlobalConfig stpGlobalConfig port-number 1 enable
    setStpGlobalConfig stpGlobalConfig recovery interval 3000
    setStpGlobalConfig stpGlobalConfig transmit hold-count 10

UCS(config-controller-ProvisionStpPortType)# setStpGlobalConfig commit

    SetStpGlobalConfig Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Viewing Spanning-Tree Global Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.

	Command or Action	Purpose
Step 3	**getStpglobalConfig getStpGlobalConfigRequest** **Example:** `UCS(ProvisionStpPortType)# getStpglobalConfig getStpGlobalConfigRequest`	Displays the configuration.
Step 4	**getStpglobalConfig commit** **Example:** `UCS(ProvisionStpPortType)# getStpglobalConfig commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# getStpglobalConfig getStpGlobalConfigRequest

   stpGlobalConfig.edge.bpdu_filter = false
   stpGlobalConfig.edge.bpdu_guard = true
   stpGlobalConfig.mode.t = 1
   stpGlobalConfig.mode.u.mstp = false
   stpGlobalConfig.mst.instance[0].active = true
   stpGlobalConfig.mst.instance[0].priority = 0
   stpGlobalConfig.mst.instance[0].vlan = '1'
   stpGlobalConfig.mst.forward_time = 4
   stpGlobalConfig.mst.max_age = 30
   stpGlobalConfig.mst.max_hops = 30
   stpGlobalConfig.mst.name = 'sandino123'
   stpGlobalConfig.mst.revision = 1111
   stpGlobalConfig.recovery.interval = 3000
   stpGlobalConfig.transmit.hold_count = 10
   stpGlobalConfig.port_number[0] = true
   stpGlobalConfig.port_number[1] = true
   stpGlobalConfig.port_number[2] = true
   stpGlobalConfig.port_number[3] = true
   stpGlobalConfig.port_number[4] = true
   stpGlobalConfig.port_number[5] = true

UCS(ProvisionStpPortType)# getStpglobalConfig commit

   GetStpglobalConfig Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Configuring Spanning-Tree Port Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.

Step 3	**setStpportConfig stpPortConfig** {**auto-edge** {**enable** | **disable**} |**bpdu-guard** {**enable** | **disable**} | **edge** {**enable** | **disable**} | **link-type** {**auto** |**point-to** | **shared**} {**enable** | **disable**} | **mst instance** *instance-id* {**active** {**enable** | **disable**} | **cost** {**auto** | **cost-range***cost_range*} | **port-priority***port_priority*} | **port-number** *Port -number* |**restricted-role** {**enable** | **disable**} | **restricted-tcn** {**enable** | **disable**}}
	
	Example:
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig auto-edge enable`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig bpdu-guard disable`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig edge disable`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig link-type auto enable`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig link-type point-to disable`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig link-type shared disable`
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance instance-id 1`
	`setStpAggConfig stpAggrConfig mst occur instance active enable`
	`UCS(ProvisionStpPortType)# setStpAggConfig stpAggrConfig mst occur instance port-priority 2`
	
	`setStpAggConfig stpAggrConfig mst occur instance cost auto`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig mst instance 0 cost cost-range 1`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig mst instance 0 port-priority 1`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig restricted-role enable`
	`UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig restricted-tcn disable`
		Configures the spanning-tree port configuration.
		• **stpPortConfig—**Sets the spanning-tree port configuration.
		• **auto-edge—**Detects the auto-edge status.
		  ◦ **enable—**Enables the auto-edge
		  ◦ **disable—**Disables the auto-edge
		• **bpdu-guard—**Configures the BPDU guard.
		  ◦ **enable—**Enables the bpdu-guard
		  ◦ **disable—**Disables the bpdu-guard
		• **edge—**Configures the edge port.
		  ◦ **enable—**Enables the edge.
		  ◦ **disable—**Disables the edge.
		• **link-type—**Configures the port link-type.
		  ◦ **auto—**Configures the link-type as auto.
		    ◦ **enable—**Enables the link-type as auto.
		    ◦ **disable—**Disables the link-type as auto.
		  ◦ **point-to—**Forces the link-type as point-to-point.
		    ◦ **enable—**Enables the link-type as point-to.
		    ◦ **disable—**Disables the link-type as point-to.
		  ◦ **shared—**Forces the link-type as shared.
		    ◦ **enable—**Enables the link-type as shared.

	Command or Action	Purpose
		◦ **disable—**Disables the link-type as shared. • **mst—**Configures the STP bridge instance. ◦ *instance*—Instance. The range is from 0 to 7 where CIST=0, MST2=1 and so on. ◦ **active—**Adds or removes an instance. ◦ **enable—**Enables the mst instance as active. ◦ **disable—**Disables the mst instance as active. ◦ **cost—**Configures the STP cost for the port. ◦ **auto—**Uses auto cost. ◦ *cost-range*—Cost range. The range is from 1-200000000. ◦ *port-priority*—STP priority of the port. The range is from 0 to 240. • **port_number—**Configures the port number. ◦ *Port number*—Port number. The range is from 1 to 6. • **restricted-role—**Configures the port role. It is restricted (and never a root port). ◦ **enable—**Enables the port as having restricted role. ◦ **disable—**Disables the port as having restricted role. • **restricted-tcn—**Restricts the topology change notifications. ◦ **enable—**Enables the restricted TCN. ◦ **disable—**Disables the restricted TCN.
Step 4	**setstpPortConfig review** **Example:** `UCS(ProvisionStpPortType)# setstpPortConfig review`	Displays the configuration.
Step 5	**setstpPortConfig commit** **Example:** `UCS(ProvisionStpPortType)# setstpPortConfig commit`	Sends the configuration to the NID.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig auto-edge enable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig bpdu-guard disable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig edge disable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig link-type auto enable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig link-type point-to disable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig link-type shared disable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig mst instance 0 active enable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig mst instance 0 cost auto
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig mst instance 0 cost cost-range 1
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig mst instance 0 port-priority 1
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig restricted-role enable
UCS(ProvisionStpPortType)# setstpPortConfig stpPortConfig restricted-tcn disable
UCS(ProvisionStpPortType)# setstpPortConfig review

Commands in queue:

    setstpPortConfig stpPortConfig auto-edge enable
    setstpPortConfig stpPortConfig bpdu-guard disable
    setstpPortConfig stpPortConfig edge disable
    setstpPortConfig stpPortConfig link-type auto enable
    setstpPortConfig stpPortConfig link-type point-to disable
    setstpPortConfig stpPortConfig link-type shared disable
    setstpPortConfig stpPortConfig mst instance 0 active enable
    setstpPortConfig stpPortConfig mst instance 0 cost auto
    setstpPortConfig stpPortConfig mst instance 0 cost cost-range 1
    setstpPortConfig stpPortConfig mst instance 0 port-priority 1
    setstpPortConfig stpPortConfig restricted-role enable
    setstpPortConfig stpPortConfig restricted-tcn disable

UCS(ProvisionStpPortType)# setstpPortConfig commit

    SetStpAggConfig Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Viewing Spanning-Tree Protocol Port Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.
Step 3	**getStpportConfig getstpPortConfigRequest** {**port_number***port_number*} **Example:** `UCS(ProvisionStpPortType)# getStpportConfig getstpPortConfigRequest port_number 1`	Displays the configuration. • *port_number*—Port number. The range is from 1 to 6.
Step 4	**getStpportConfig commit** **Example:** `UCS(ProvisionStpPortType)# getStpportConfig commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# getStpportConfig getstpPortConfigRequest port_number 1

   stpPortConfig.port_number = 1
   stpPortConfig.auto_edge = false
   stpPortConfig.bpdu_guard = false
   stpPortConfig.edge = false
   stpPortConfig.link_type.t = 1
   stpPortConfig.link_type.u.auto_ = false
   stpPortConfig.restricted_role = false
   stpPortConfig.restricted_tcn = false

UCS(ProvisionStpPortType)# getStpportConfig commit

   GetStpPortConfig Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Verifying Spanning-Tree Status

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.
Step 3	**showStpdetail showStpDetailRequest** **Example:** `UCS(ProvisionStpPortType)# showStpdetail showStpDetailRequest`	Displays the STP status.
Step 4	**showStpdetail commit** **Example:** `UCS(ProvisionStpPortType)# showStpdetail commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# showStpdetail showStpDetailRequest

   stpinfo.instance[0].instance_id = 0
   stpinfo.instance[0].name = 'CIST'
   stpinfo.instance[0].bridgeId = '32768.00-3A-99-FD-4B-1C'
   stpinfo.instance[0].designatedRoot = '8192.00-14-1B-EC-1A-BF'
   stpinfo.instance[0].rootport = '1'
   stpinfo.instance[0].rootPathCost = 200022
   stpinfo.instance[0].RegionalRoot = '32768.00-3A-99-FD-4B-1C'
   stpinfo.instance[0].InternalPathCost = 0
   stpinfo.instance[0].maxHops = 20
   stpinfo.instance[0].topologyChange = 'Steady'
   stpinfo.instance[0].topologyChangeCount = 31
   stpinfo.instance[0].timeSinceTopologyChange = '  0d 00:04:49'
   stpinfo.instance[0].port_status[0].active = true
   stpinfo.instance[0].port_status[0].name = 'CIST'
   stpinfo.instance[0].port_status[0].port = '1'
   stpinfo.instance[0].port_status[0].port_role = 'RootPort'
   stpinfo.instance[0].port_status[0].state = 'Forwarding'
   stpinfo.instance[0].port_status[0].priority = 128
   stpinfo.instance[0].port_status[0].pathcost = 3392
   stpinfo.instance[0].port_status[0].edge = false
   stpinfo.instance[0].port_status[0].ptp = true
   stpinfo.instance[0].port_status[0].uptime = '  0d 00:05:10'

UCS(ProvisionStpPortType)# showStpdetail commit

   ShowStpDetail Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Verifying Spanning-Tree Summary

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.
Step 3	**showStpsummary showstpSummaryRequest** **Example:** `UCS(ProvisionStpPortType)# showStpsummary showstpSummaryRequest`	Displays the STP summary.
Step 4	**showStpdetail commit** **Example:** `UCS(ProvisionStpPortType)# showStpsummary commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# showStpdetail showStpDetailRequest

    StpSummaryinfo.Protocol = 'MSTP'
    StpSummaryinfo.MaxAge = 20
    StpSummaryinfo.ForwardDelay = 15
    StpSummaryinfo.txHoldCount = 6
    StpSummaryinfo.MaxHops = 20
    StpSummaryinfo.bpduFiltering = false
    StpSummaryinfo.bpduGuard = false
    StpSummaryinfo.errRecoveryDelay = 0
    StpSummaryinfo.mstp_bridge[0].instance_id = 0
    StpSummaryinfo.mstp_bridge[0].name = 'CIST'
    StpSummaryinfo.mstp_bridge[0].status = true
    StpSummaryinfo.mstp_bridge[1].instance_id = 1
    StpSummaryinfo.mstp_bridge[1].name = 'MSTI1'
    StpSummaryinfo.mstp_bridge[1].status = false
    StpSummaryinfo.mstp_bridge[2].instance_id = 2
    StpSummaryinfo.mstp_bridge[2].name = 'MSTI2'
    StpSummaryinfo.mstp_bridge[2].status = false
    StpSummaryinfo.mstp_bridge[3].instance_id = 3
    StpSummaryinfo.mstp_bridge[3].name = 'MSTI3'
    StpSummaryinfo.mstp_bridge[3].status = false
```

```
    StpSummaryinfo.mstp_bridge[4].instance_id = 4
    StpSummaryinfo.mstp_bridge[4].name = 'MSTI4'
    StpSummaryinfo.mstp_bridge[4].status = false
    StpSummaryinfo.mstp_bridge[5].instance_id = 5
    StpSummaryinfo.mstp_bridge[5].name = 'MSTI5'
    StpSummaryinfo.mstp_bridge[5].status = false
    StpSummaryinfo.mstp_bridge[6].instance_id = 6
    StpSummaryinfo.mstp_bridge[6].name = 'MSTI6'
    StpSummaryinfo.mstp_bridge[6].status = false
    StpSummaryinfo.mstp_bridge[7].instance_id = 7
    StpSummaryinfo.mstp_bridge[7].name = 'MSTI7'
    StpSummaryinfo.mstp_bridge[7].status = false
    StpSummaryinfo.portcounters[0].port_number = 0
    StpSummaryinfo.portcounters[0].rxMstp = 0
    StpSummaryinfo.portcounters[0].txMstp = 4
    StpSummaryinfo.portcounters[0].rxRstp = 0
    StpSummaryinfo.portcounters[0].txRstp = 0
    StpSummaryinfo.portcounters[0].rxstp = 144
    StpSummaryinfo.portcounters[0].txstp = 122790
    StpSummaryinfo.portcounters[0].rxtcn = 29
    StpSummaryinfo.portcounters[0].txtcn = 2
    StpSummaryinfo.portcounters[0].rxIllegalFrames = 0
    StpSummaryinfo.portcounters[0].unknownFrames = 0

UCS(ProvisionStpPortType)# showStpsummary commit

    ShowStpSummary Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Clearing Spanning-Tree Statistics

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.
Step 3	**clearStpstatistics stpPortSelect** {**all** | **port** {*port-number*}} **Example:** `UCS(ProvisionStpPortType)# clearStpstatistics stpPortSelect port-number 1`	Clears the spanning-tree statistics. • **all—**Clears the statistics from all the ports. • **port—**Clears the statistics from a specified port number. ◦ *port-number***—**Port number. The range is from 1 to 6.

	Command or Action	Purpose
Step 4	**ClearStpStatistics review** **Example:** `UCS(ProvisionStpPortType)# ClearStpStatistics review`	Displays the configuration.
Step 5	**ClearStpStatistics Commit** **Example:** `UCS(ProvisionStpPortType)# ClearStpStatistics Commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# clearStpstatistics stpPortSelect port-number 1
UCS(ProvisionStpPortType)# ClearStpStatistics Review

Commands in queue:
      clearStpstatistics stpPortSelect port-number 1

UCS(ProvisionStpPortType)# ClearStpStatistics Commit

      ClearStpStatistics Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

Clearing Spanning-Tree Detected Protocols

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStpPortType** **Example:** `UCS# ProvisionStpPortType`	Enters the ProvisionStpPortType mode.

	Command or Action	Purpose
Step 3	**clearStpdetected stpPortSelect** {**all** | **port** {*port-number*}} **Example:** `UCS(ProvisionStpPortType)# clearStpdetected stpPortSelect port-number 1`	Clear spanning-tree detected-protocols. • **all**—Clears from all the ports. • **port**—Clears from a specified port number. ◦ *port-number*—Port number. The range is from 1 to 6.
Step 4	**clearStpdetected review** **Example:** `UCS(ProvisionStpPortType)# clearStpdetected review`	Displays the configuration.
Step 5	**clearStpdetected commit** **Example:** `UCS(ProvisionStpPortType)# clearStpdetected commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# clearStpdetected stpPortSelect port-number 1
UCS(ProvisionStpPortType)# clearStpdetected review
Commands in queue:
    clearStpdetected stpPortSelect port-number 1

UCS(ProvisionStpPortType)# clearStpdetected commit

    clearStpdetected Commit Success!!!

UCS(ProvisionStpPortType)# exit
```

CHAPTER 11

Configuring Link Aggregation Control Protocol (LACP)

LACP is defined in IEEE 802.3ad standard and enables Cisco switches to manage Ethernet channels between switches that conform to the standard. LACP facilitates the automatic creation of EtherChannels by exchanging LACP packets between Ethernet ports.

By using LACP, the switch learns the identity of partners capable of supporting LACP and the capabilities of each port. It then dynamically groups similarly configured port s into a single logical link (channel or aggregate port). Similarly configured ports are grouped based on key value. For example, LACP groups the ports with the same speed, duplex mode, native VLAN, VLAN range, and trunking status and type.

Information About LACP

IEEE 802.3ad Link Bundling

The IEEE 802.3ad Link Bundling feature provides a method for aggregating multiple Ethernet links into a single logical channel based on the IEEE 802.3ad standard. This feature helps improve the cost effectiveness of a device by increasing cumulative bandwidth without necessarily requiring hardware upgrades. In addition, IEEE 802.3ad link bundling provides a capability to dynamically provision, manage, and monitor various aggregated links and enables interoperability between various Cisco devices and devices of third-party vendors.

LACP uses the following parameters:

- LACP port priority—You must configure an LACP port priority on each port configured to use LACP. The port priority can be configured automatically or through the CLI. LACP uses the port priority to decide which ports should be put in standby mode when there is a hardware limitation that prevents all compatible ports from aggregating. LACP also uses the port priority with the port number to form the port identifier.
-

How to Configure LACP

Provisioning the UCS Controller to Configure LACP

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionLacpPortType** **Example:** `UCS#` **`ProvisionLacpPortType`**	Enters LACP provisioning mode.

Step 3	**ProvisionLacpPortType** {**clearLacpStats** | **default** | **exit** | **getLacpConfig** | **getLacpDefaults** | **getLacpPortConfig** | **getLacpPortDefaults** | **no** | **setLacpConfig** | **setLacpDefaults** | **setLacpPortConfig** | **setLacpPortDefaults** | **showLacpAggLB** | **showLacpInternal** | **showLacpNeighbors** | **showLacpStats** | **showLacpSysId**}
	
	Example:
	`UCS(ProvisionLacpPortType)# ?`
	`  ProvisionLacpPortType sub-mode commands:`
	`  clearLacpStats       Clear LACP statistics request`
	`  default              Set a command to its defaults`
	`  exit                 Exit from ProvisionLacpPortType sub configuration`
	`mode`
	`  getLacpConfig        Get LACP configuration request`
	`  getLacpDefaults      Get LACP default configuration request`
	`  getLacpPortConfig    Get LACP port configuration request`
	`  getLacpPortDefaults  Get LACP port default configuration request`
	`  no                   Negate a command or set its defaults`
	`  setLacpConfig        Set LACP configuration request`
	`  setLacpDefaults      Set LACP default configuration request`
	`  setLacpPortConfig    Set LACP port configuration request`
	`  setLacpPortDefaults  Set LACP port default configuration request`
	`  showLacpAggLB        Show LACP load balance request`
	`  showLacpInternal     Show LACP internal request`
	`  showLacpNeighbors    Show LACP neighbor status request`
	`  showLacpStats        Show LACP statistics request`
	`  showLacpSysId        Show LACP system-id request`
		Displays the supported configurations for LACP.

Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)#` **`exit`**	Exits the LACP provisioning mode.

Configuration Example

The following example shows the supported LACP configuration:

```
UCS(ProvisionLacpPortType)# ?
ProvisionLacpPortType sub-mode commands:
  clearLacpStats       Clear LACP statistics request
  default              Set a command to its defaults
  exit                 Exit from ProvisionLacpPortType sub configuration mode
  getLacpConfig        Get LACP configuration request
  getLacpDefaults      Get LACP default configuration request
  getLacpPortConfig    Get LACP port configuration request
  getLacpPortDefaults  Get LACP port default configuration request
  no                   Negate a command or set its defaults
  setLacpConfig        Set LACP configuration request
  setLacpDefaults      Set LACP default configuration request
  setLacpPortConfig    Set LACP port configuration request
  setLacpPortDefaults  Set LACP port default configuration request
  showLacpAggLB        Show LACP load balance request
  showLacpInternal     Show LACP internal request
  showLacpNeighbors    Show LACP neighbor status request
  showLacpStats        Show LACP statistics request
  showLacpSysId        Show LACP system-id request
```

Configuring LACP Globally on the UCS Controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setLacpConfig** {**commit** | **flush** | **lacpGlobalConfiguration** | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# setLacpConfig ?`
	` commit                    commit setLacpConfig`
	` flush                     flush all setLacpConfig`
	`commands from queue`
	` lacpGlobalConfiguration  Set LACP configuration`
	`request`
	` review                    review setLacpConfig`
	`commands`
		Configures global LACP.
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **lacpGlobalConfiguration—**Sets LACP configuration globally on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**setLacpConfig lacpGlobalConfiguration** {**lacpGlobalState** {**enable**} | **systemPrio***priority_value*}
	
	Example:
	`UCS(ProvisionLacpPortType)# setLacpConfig lacpGlobalConfiguration lacpGlobalState enable`
	`UCS(ProvisionLacpPortType)# setLacpConfig lacpGlobalConfiguration systemPrio 2`
		Sets global LACP configuration.
		• **lacpGlobalState—**Enables the LACP configuration globally on the UCS controller.
		Note LACP is always enabled globally. Disable is not supported.
		• **enable—**Enables global LACP configuration.
		• **systemPrio** *priority_value***—**Sets priority value. The valid range is from 1 to 65535.

	Command or Action	Purpose

Step 3	**setLacpConfig review**
	
	Example:
	`UCS(ProvisionLacpPortType)# setLacpConfig review`
	`Commands in queue:`
	`        setLacpConfig lacpGlobalConfiguration`
	`lacpGlobalState enable`
	`        setLacpConfig lacpGlobalConfiguration`
	`systemPrio 2`
		Displays the LACP configuration on the UCS controller.

Step 4	**setLacpConfig commit** **Example:** `UCS(ProvisionLacpPortType)# setLacpConfig commit`	Sends the LACP configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionLacpPortType)#exit`	Exits the LACP provisioning mode.

Configuration Example

The example shows global LACP configuration on the UCS controller:

```
UCS(ProvisionLacpPortType)# setLacpConfig lacpGlobalConfiguration lacpGlobalState enable
UCS(ProvisionLacpPortType)# setLacpConfig lacpGlobalConfiguration systemPrio 2
UCS(ProvisionLacpPortType)# setLacpConfig review
Commands in queue:
        setLacpConfig lacpGlobalConfiguration lacpGlobalState enable
        setLacpConfig lacpGlobalConfiguration systemPrio 2
UCS(ProvisionLacpPortType)# setLacpConfig commit
 SetLacpConfig Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Configuring LACP Defaults Globally on the UCS Controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setLacpDefaults** {**commit** | **flush** | **setLacpDefaultsRequest** | **review**}	Configures default LACP globally. • **commit**—Sends the LACP configuration to NID.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionLacpPortType)# setLacpDefaults ?`
	` commit                    commit setLacpDefaults`
	` flush                     flush all setLacpDefaults`
	`commands from queue`
	` review                    review setLacpDefaults`
	`commands`
	` setLacpDefaultsRequest  Set LACP default`
	`configuration request`
	`UCS(ProvisionLacpPortType)# setLacpDefaults`
	`setLacpDefaultsRequest`
		• **flush—**Flushes all LACP configuration from the queue.
		• **setLacpDefaultsRequest—**Sets LACP default configuration globally on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**setLacpDefaults review**
	Example:
	`UCS(ProvisionLacpPortType)# setLacpDefaults review`
	`Commands in queue:`
	`        setLacpDefaults setLacpDefaultsRequest`
	`        setLacpDefaults setLacpDefaultsRequest`
		Displays the default LACP configuration on the UCS controller.
		Note The default system priority value is set to 32768.

Step 3	**setLacpDefaults commit** **Example:** `UCS(ProvisionLacpPortType)# setLacpDefaults commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example how to configure default LACP configuration on the UCS controller:

```
UCS(ProvisionLacpPortType)# setLacpDefaults setLacpDefaultsRequest
UCS(ProvisionLacpPortType)# setLacpDefaults review
Commands in queue:
        setLacpDefaults setLacpDefaultsRequest
UCS(ProvisionLacpPortType)# setLacpDefaults commit
 SetLacpConfig Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Configuring LACP at Port level on the UCS Controller

Before You Begin

- Perform the steps to provision LACP on the UCS Controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setLacpPortConfig** {**commit** | **flush** | **lacpPortConfiguration** | **review**}
	
	Example:
	
	`UCS(ProvisionLacpPortType)# `**`setLacpPortConfig ?`**
	` commit                 commit setLacpPortConfig`
	` flush                  flush all setLacpPortConfig`
	`commands from queue`
	` lacpPortConfiguration  Set LACP port configuration`
	`request`
	` review                review setLacpPortConfig commands`
		Configures LACP at port level.
		
		• **commit—**Sends the LACP configuration to the NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **lacpPortConfiguration—**Sets LACP configuration at port level on the UCS Controller.
		• **review—**Displays the configuration on the controller.

Step 2	**setLacpPortConfig lacpPortConfiguration** {**key***key_group* | **lacpEnable** {**enable** | **disable**} | **portNumber** *port-num* | **portPriority** *priority-value* | **role** {**active** | **passive**} {**enable** | **disable**} | **timeout** {**fast** | **slow**} {**enable** | **disable**}} **Example:** `UCS(ProvisionLacpPortType)# `**`setLacpPortConfig lacpPortConfiguration key 1`** `UCS(ProvisionLacpPortType)# `**`setLacpPortConfig lacpPortConfiguration lacpEnable enable`** `UCS(ProvisionLacpPortType)# `**`setLacpPortConfig lacpPortConfiguration portNumber 2`** `UCS(ProvisionLacpPortType)# `**`setLacpPortConfig lacpPortConfiguration role active enable`** `UCS(ProvisionLacpPortType)# `**`setLacpPortConfig lacpPortConfiguration portPriority 23`** `UCS(ProvisionLacpPortType)# `**`setLacpPortConfig lacpPortConfiguration timeout fast enable`**	Configures LACP port configuration. • **key** *key_group***—**Specifies the key or channel group for LACP aggregation. The valid range is 0 to 65535. • **lacpEnable —**Enables LACP on the interface. • **enable—**Enables LACP configuration. • **disable—**Disables LACP configuration. • **portNumber** *port-num***—**Specifies the targeted port. The valid range is from 1 to 6. • **portPriority** *priority-value***—**Specifies the LACP priority. The valid range is from 1 to 65535. • **role—**Sets the activity mode. • **active—**Transmits the LACP BPDUs actively. • **passive—**Waits for the neighbor before transmitting. • **timeout—**Sets period between BPDU transmissions. • **fast—**Transmits BPDUs every second. • **slow—**Transmits BPDUs every 30th second.

Step 3	**setLacpPortConfig review**
	
	Example:
	
	`UCS(ProvisionLacpPortType)# `**`setLacpPortConfig review`**
	`Commands in queue:`
	`        setLacpPortConfig lacpPortConfiguration key 3`
	`       setLacpPortConfig lacpPortConfiguration lacpEnable`
		Displays the LACP configuration on the UCS Controller.

	Command or Action	Purpose

	`enable`
	`setLacpPortConfig lacpPortConfiguration portNumber 2`
	`setLacpPortConfig lacpPortConfiguration portPriority 2`
	`setLacpPortConfig lacpPortConfiguration role active enable`
	`setLacpPortConfig lacpPortConfiguration timeout fast enable`
	`setLacpPortConfig lacpPortConfiguration key 2`
	`setLacpPortConfig lacpPortConfiguration lacpEnable enable`
	`setLacpPortConfig lacpPortConfiguration portNumber 2`
	`setLacpPortConfig lacpPortConfiguration role active enable`
	`setLacpPortConfig lacpPortConfiguration timeout fast enable`	

Step 4	**setLacpPortConfigcommit** **Example:** `UCS(ProvisionLacpPortType)#` **`setLacpPortConfig commit`**	Sends the LACP configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionLacpPortType)#` **`exit`**	Exits the LACP provisioning mode.

Configuration Example

The example shows LACP port configuration on the UCS Controller:

```
UCS(ProvisionLacpPortType)# setLacpPortConfig lacpPortConfiguration key 1
UCS(ProvisionLacpPortType)# setLacpPortConfig lacpPortConfiguration lacpEnable enable
UCS(ProvisionLacpPortType)# setLacpPortConfig lacpPortConfiguration portNumber 2
UCS(ProvisionLacpPortType)# setLacpPortConfig lacpPortConfiguration role active enable
UCS(ProvisionLacpPortType)# setLacpPortConfig lacpPortConfiguration portPriority 23
UCS(ProvisionLacpPortType)# setLacpPortConfig lacpPortConfiguration timeout fast enable
UCS(ProvisionLacpPortType)# setLacpPortConfig review
Commands in queue:
        setLacpPortConfig lacpPortConfiguration key 3
        setLacpPortConfig lacpPortConfiguration lacpEnable enable
        setLacpPortConfig lacpPortConfiguration portNumber 2
        setLacpPortConfig lacpPortConfiguration portPriority 2
        setLacpPortConfig lacpPortConfiguration role active enable
        setLacpPortConfig lacpPortConfiguration timeout fast enable
        setLacpPortConfig lacpPortConfiguration key 2
        setLacpPortConfig lacpPortConfiguration lacpEnable enable
        setLacpPortConfig lacpPortConfiguration portNumber 2
        setLacpPortConfig lacpPortConfiguration role active enable
        setLacpPortConfig lacpPortConfiguration timeout fast enable
UCS(ProvisionLacpPortType)# setLacpPortConfig commit
 SetLacpPortConfig Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Configuring Default LACP Configuration at Port level on the UCS Controller

The default values for LACP port parameters are:

- lacpEnable: false
- portPriority 32768
- role: active
- timeout: fast

There is no default value for key. Configure a valid value to identify the LACP channel aggregation group. If no value is set, key value is displayed as 0.

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setLacpPortDefaults** {**commit** | **flush** | **lacpPhysicalPort***port_num*| **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# setLacpPortDefaults ?`
	` commit            commit setLacpPortDefaults`
	` flush             flush all setLacpPortDefaults`
	`commands from queue`
	` lacpPhysicalPort  Set LACP port default configuration`
	` request`
	` review           review setLacpPortDefaults commands`
	`UCS(ProvisionLacpPortType)# setLacpPortDefaults`
	`lacpPhysicalPort 2`
		Configures default LACP at port level.
		• **commit**—Sends the LACP configuration to NID.
		• **flush**—Flushes all LACP configuration from the queue.
		• **lacpPhysicalPort** *port_num*—Sets LACP default configuration at port level on the controller. The valid ports are 1 to 6.
		• **review**—Displays the configuration on the UCS controller.

Step 2	**setLacpPortDefaults review**
	
	Example:
	`UCS(ProvisionLacpPortType)# setLacpPortDefaults review`
	`Commands in queue:`
	`        setLacpPortDefaults lacpPhysicalPort 2`
		Displays the LACP configuration on the UCS controller.

Step 3	**setLacpPortDefaults commit** **Example:** `UCS(ProvisionLacpPortType)# setLacpPortDefaults commit`	Sends the LACP configuration to the NID.

	Command or Action	Purpose
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example shows default LACP port configuration on the UCS controller:

```
UCS(ProvisionLacpPortType)# setLacpPortDefaults lacpPhysicalPort 2
UCS(ProvisionLacpPortType)# setLacpPortDefaults review
Commands in queue:
        setLacpPortDefaults lacpPhysicalPort 2
UCS(ProvisionLacpPortType)# setLacpPortDefaults commit
SetLacpPortDefaults Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Clearing LACP Statistics on the UCS Controller

Before You Begin

- Perform the steps to provision LACP on the UCS Controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**clearLacpStats** {**commit** | **flush** | **lacpPhysicalPort***port_num*| **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# clearLacpStats ?`
	`  commit             commit clearLacpStats`
	`  flush            flush all clearLacpStats commands`
	` from queue`
	`  lacpPhysicalPort  Clear LACP statistics request`
	`  review            review clearLacpStats commands`
	`UCS(ProvisionLacpPortType)# clearLacpStats lacpPhysicalPort 3`
		Clears LACP statistics.
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **lacpPhysicalPort** *port_num***—**Clears the LACP statistics on a specified port on the UCS Controller. The valid values are 1 to 6.
		• **review—**Displays the configuration on the UCS Controller.

Step 2	**clearLacpStats review**
	
	Example:
	`UCS(ProvisionLacpPortType)# clearLacpStats review`
	`Commands in queue:`
	`        clearLacpStats lacpPhysicalPort 3`
		Displays the LACP configuration on the UCS Controller.

	Command or Action	Purpose
Step 3	**clearLacpStats commit** **Example:** `UCS(ProvisionLacpPortType)# clearLacpStats commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example clears the LACP statistics on port 3 on the UCS Controller:

```
UCS(ProvisionLacpPortType)# clearLacpStats lacpPhysicalPort 3
UCS(ProvisionLacpPortType)# clearLacpStats review
Commands in queue:
        clearLacpStats lacpPhysicalPort 3
UCS(ProvisionLacpPortType)# clearLacpStats commit
ClearLacpStats_Output.clearLacpStatsResponse = 0

 ClearLacpStats Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Negating LACP Configuration and Restoring Defaults

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**no ?**
	
	Example:
	`UCS(ProvisionLacpPortType)# no ?`
	
	`clearLacpStats       Clear LACP statistics request`
	`exit                 Exit from ProvisionLacpPortType sub configuration`
	`mode`
	`getLacpConfig        Get LACP configuration request`
	`getLacpDefaults      Get LACP default configuration request`
	`getLacpPortConfig    Get LACP port configuration request`
	`getLacpPortDefaults  Get LACP port default configuration request`
	`setLacpConfig        Set LACP configuration request`
	`setLacpDefaults      Set LACP default configuration request`
	`setLacpPortConfig    Set LACP port configuration request`
	`setLacpPortDefaults  Set LACP port default configuration request`
		Negates the commands and sets the default configuration.

	Command or Action	Purpose

	`showLacpAggLB        Show LACP load balance request`
	`showLacpInternal     Show LACP internal request`
	`showLacpNeighbors    Show LACP neighbor status request`
	`showLacpStats        Show LACP statistics request`
	`showLacpSysId        Show LACP system-id request`	

Step 2	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Viewing the Global LACP Configuration on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getLacpConfig** {**commit** | **flush** | **getLacpConfigRequest** | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# getLacpConfig ?`
	` commit                commit getLacpConfig`
	` flush                flush all getLacpConfig commands`
	` from queue`
	` getLacpConfigRequest  Get LACP configuration request`
	` review                review getLacpConfig commands`
	` commit                commit getLacpConfig`
	`UCS(ProvisionLacpPortType)# getLacpConfig`
	`getLacpConfigRequest`
		Retrieve the global LACP configuration.
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **getLacpConfigRequest—**Retrieves the configured global LACP configuration on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**getLacpConfig review**
	
	Example:
	`UCS(ProvisionLacpPortType)# getLacpConfig review`
	`Commands in queue:`
	`        getLacpConfig getLacpConfigRequest`
	`        getLacpConfig getLacpConfigRequest`
		Displays the LACP configuration on the UCS controller.

Step 3	**getLacpConfig commit** **Example:** `UCS(ProvisionLacpPortType)# getLacpConfig commit`	Sends the LACP configuration to the NID.

	Command or Action	Purpose
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example retrieves the global LACP configuration on the UCS controller:

```
UCS(ProvisionLacpPortType)# getLacpConfig getLacpConfigRequest
UCS(ProvisionLacpPortType)# getLacpConfig review
Commands in queue:
        getLacpConfig getLacpConfigRequest
        getLacpConfig getLacpConfigRequest
UCS(ProvisionLacpPortType)# getLacpConfig commit
GetLacpConfig_Output.lacpGlobalConfiguration.systemPrio = 32768
GetLacpConfig_Output.lacpGlobalConfiguration.lacpGlobalState = true
 GetLacpConfig Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Viewing the Default LACP Configuration on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getLacpDefaults** {**commit** | **flush** | **getLacpDefaultsRequest** | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# getLacpDefaults ?`
	` commit                  commit getLacpDefaults`
	` flush                   flush all getLacpDefaults`
	`commands from queue`
	` getLacpDefaultsRequest  Get LACP default configuration`
	` request`
	` review                  review getLacpDefaults`
	`commands`
	`UCS(ProvisionLacpPortType)# getLacpDefaults getLacpDefaultsRequest`
		Retrieves the default LACP configuration.
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **getLacpDefaultsRequest—**Retrieves the default LACP configuration on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**getLacpDefaults review** **Example:** `UCS(ProvisionLacpPortType)# getLacpDefaults review`	Displays the LACP configuration on the UCS controller.

	Command or Action	Purpose
	`Commands in queue:` `        getLacpDefaults getLacpDefaultsRequest`	
Step 3	**getLacpDefaults commit** **Example:** `UCS(ProvisionLacpPortType)# ` **`getLacpDefaults commit`**	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# ` **`exit`**	Exits the LACP provisioning mode.

Configuration Example

The example retrieves the defaukt LACP configuration on the UCS controller:

```
UCS(ProvisionLacpPortType)# getLacpDefaults getLacpDefaultsRequest
UCS(ProvisionLacpPortType)# getLacpDefaults review
Commands in queue:
        getLacpDefaults getLacpDefaultsRequest
UCS(ProvisionLacpPortType)# getLacpDefaults commit
GetLacpDefaults_Output.lacpGlobalConfiguration.systemPrio = 32768
GetLacpDefaults_Output.lacpGlobalConfiguration.lacpGlobalState = true

 GetLacpDefaults Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Viewing the LACP Configuration at Port Level on the UCS Controller

Before You Begin

- Perform the steps to provision LACP on the UCS Controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getLacpPortConfig** {**commit** | **flush** | **lacpPhysicalPort***port_num* | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# ` **`getLacpPortConfig ?`**
	` commit            commit getLacpPortConfig`
	` flush             flush all getLacpPortConfig`
	`commands from queue`
	` lacpPhysicalPort  Get LACP port configuration request`
	
	` review            review getLacpPortConfig commands`
		Retrieves the LACP configuration at port.
		• **commit**—Sends the LACP configuration to NID.
		• **flush**—Flushes all LACP configuration from the queue.
		• **lacpPhysicalPort** *port_num*—Retrieves the LACP configuration for specified port on the controller. The valid values are 1 to 6.

	Command or Action	Purpose
	`UCS(ProvisionLacpPortType)# getLacpPortConfig lacpPhysicalPort 1`	• **review—**Displays the configuration on the UCS Controller.

Step 2	**getLacpPortConfig review**
	Example:
	`UCS(ProvisionLacpPortType)# getLacpPortConfig review`
	`Commands in queue:`
	`        getLacpPortConfig lacpPhysicalPort 1`
		Displays the LACP configuration on the UCS Controller.

Step 3	**getLacpPortConfig commit** **Example:** `UCS(ProvisionLacpPortType)# getLacpPortConfig commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example retrieves the LACP configuration for port 1 on the UCS Controller:

```
UCS(ProvisionLacpPortType)# getLacpPortConfig lacpPhysicalPort 1
UCS(ProvisionLacpPortType)# getLacpPortConfig review
Commands in queue:
        getLacpPortConfig lacpPhysicalPort 1
UCS(ProvisionLacpPortType)# getLacpPortConfig commit
GetLacpPortConfig_Output.lacpPortConfiguration.portNumber = 1
GetLacpPortConfig_Output.lacpPortConfiguration.lacpEnable = false
GetLacpPortConfig_Output.lacpPortConfiguration.key = 1
GetLacpPortConfig_Output.lacpPortConfiguration.role.t = 1
GetLacpPortConfig_Output.lacpPortConfiguration.role.u.active = true
GetLacpPortConfig_Output.lacpPortConfiguration.portPriority = 32768
GetLacpPortConfig_Output.lacpPortConfiguration.timeout.t = 1
GetLacpPortConfig_Output.lacpPortConfiguration.timeout.u.fast = true

 GetLacpPortConfig Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Viewing the Default LACP Configuration at Port Level on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getLacpPortDefaults** {**commit** | **flush** | **lacpPhysicalPort***port_num* | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# getLacpPortDefaults ?`
	` commit            commit getLacpPortDefaults`
	` flush             flush all getLacpPortDefaults`
	`commands from queue`
	` lacpPhysicalPort  Get LACP port default configuration`
	` request`
	` review           review getLacpPortDefaults commands`
	`UCS(ProvisionLacpPortType)# getLacpPortDefaults`
	`lacpPhysicalPort 1`
		Retrieve the LACP configuration at port.
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **lacpPhysicalPort** *port_num***—**Retrieves the default LACP configuration for specified port on the UCS controller. The valid values are 1 to 6.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**getLacpPortDefaults review**
	
	Example:
	`UCS(ProvisionLacpPortType)# getLacpPortDefaults review`
	`Commands in queue:`
	`        getLacpPortDefaults lacpPhysicalPort 1`
		Displays the LACP configuration on the UCS controller.

Step 3	**getLacpPortDefaults commit** **Example:** `UCS(ProvisionLacpPortType)# getLacpPortDefaults commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example retrieves the default LACP configuration for port 1 on the UCS controller:

```
UCS(ProvisionLacpPortType)# getLacpPortDefaults lacpPhysicalPort 1
UCS(ProvisionLacpPortType)# getLacpPortDefaults review
Commands in queue:
        getLacpPortDefaults lacpPhysicalPort 1
UCS(ProvisionLacpPortType)# getLacpPortDefaults commit
GetLacpPortDefaults_Output.lacpPortConfiguration.portNumber = 1
GetLacpPortDefaults_Output.lacpPortConfiguration.lacpEnable = false
GetLacpPortDefaults_Output.lacpPortConfiguration.key = 0
GetLacpPortDefaults_Output.lacpPortConfiguration.role.t = 1
GetLacpPortDefaults_Output.lacpPortConfiguration.role.u.active = true
GetLacpPortDefaults_Output.lacpPortConfiguration.portPriority = 32768
GetLacpPortDefaults_Output.lacpPortConfiguration.timeout.t = 1
GetLacpPortDefaults_Output.lacpPortConfiguration.timeout.u.fast = true
```

```
 GetLacpPortDefaults Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Note You must explicitly configure a key value. The default value for key retrieved is 0 until it is set to a value using the setLacpPortConfig operation.

Verifying LACP

Viewing the LACP System ID Information on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLacpSysId** {**commit** | **flush** | **showLacpSysIdRequest** | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# showLacpSysId ?`
	` commit                 commit showLacpSysId`
	` flush                flush all showLacpSysId commands`
	` from queue`
	` review                 review showLacpSysId commands`
	` showLacpSysIdRequest  Show LACP system-id request`
	`UCS(ProvisionLacpPortType)# showLacpSysId`
	`showLacpSysIdRequest`
		Displays the LACP system ID information.
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **showLacpSysIdRequest—**Displays the LACP system ID information on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showLacpSysId review**
	
	Example:
	`UCS(ProvisionLacpPortType)# showLacpSysId review`
	`Commands in queue:`
	`        showLacpSysId showLacpSysIdRequest`
	`        showLacpSysId showLacpSysIdRequest`
		Displays the LACP configuration on the UCS controller.

Step 3	**showLacpSysId commit** **Example:** `UCS(ProvisionLacpPortType)# showLacpAggLB commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example displays the LACP system ID information on the UCS controller:

```
UCS(ProvisionLacpPortType)# showLacpSysId showLacpSysIdRequest
Commands in queue:
        showLacpSysId showLacpSysIdRequest
        showLacpSysId showLacpSysIdRequest
UCS(ProvisionLacpPortType)# showLacpSysId commit
ShowLacpSysId_Output.showLacpSysIdResponse.systemId = 'b8-38-61-68-7b-bc'
ShowLacpSysId_Output.showLacpSysIdResponse.systemPriority = 32768

 ShowLacpSysId Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Viewing the LACP Load Balance Information on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLacpAggLB** {**commit** | **flush** | **showLacpAggLBRequest** | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# showLacpAggLB ?`
	` commit                commit showLacpAggLB`
	` flush                 flush all showLacpAggLB`
	`commands from queue`
	` review                review showLacpAggLB commands`
	` showLacpAggLBRequest  Show LACP load balance request`
	`UCS(ProvisionLacpPortType)# showLacpAggLB`
	`showLacpAggLBRequest`
		Displays LACP load balance information.
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **showLacpAggLBRequest—**Displays the LACP load balance information on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showLacpAggLB review**
	
	Example:
	`UCS(ProvisionLacpPortType)# showLacpAggLB review`
	`Commands in queue:`
	`        showLacpAggLB showLacpAggLBRequest`
		Displays the LACP configuration on the UCS controller.

Step 3	**showLacpAggLB commit** **Example:** `UCS(ProvisionLacpPortType)# showLacpAggLB commit`	Sends the LACP configuration to the NID.

	Command or Action	Purpose
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example displays the LACP load balance information on the UCS controller:

```
UCS(ProvisionLacpPortType)# showLacpAggLB showLacpAggLBRequest
UCS(ProvisionLacpPortType)# showLacpAggLB review
Commands in queue:
        showLacpAggLB showLacpAggLBRequest
UCS(ProvisionLacpPortType)# showLacpAggLB commit
ShowLacpAggLB_Output.lacpAggLBMode.smac_enable = true
ShowLacpAggLB_Output.lacpAggLBMode.dmac_enable = false
ShowLacpAggLB_Output.lacpAggLBMode.ip_enable = true
ShowLacpAggLB_Output.lacpAggLBMode.port_enable = true

 ShowLacpAggLB Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Viewing the LACP Internal State Information on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLacpInternal** {**commit** | **flush** | **lacpPhysicalPort***port_num* | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# showLacpInternal ?`
	` commit            commit showLacpInternal`
	` flush            flush all showLacpInternal commands from queue`
	` lacpPhysicalPort  Show LACP internal request`
	` review            review showLacpInternal commands`
	`UCS(ProvisionLacpPortType)# showLacpInternal lacpPhysicalPort 2`
		Displays LACP internal state information.
		• **commit**—Sends the LACP configuration to NID.
		• **flush**—Flushes all LACP configuration from the queue.
		• **lacpPhysicalPort***port_num*—Displays the LACP internal state information for specified port on the UCS controller.
		• **review**—Displays the configuration on the UCS controller.

	Command or Action	Purpose

Step 2	**showLacpInternal review**
	
	Example:
	
	`UCS(ProvisionLacpPortType)# showLacpInternal review`
	`Commands in queue:`
	`showLacpNeighbors lacpPhysicalPort 1`
	`showLacpInternal lacpPhysicalPort 1`
		Displays the LACP configuration on the UCS controller.

Step 3	**showLacpInternal commit** **Example:** `UCS(ProvisionLacpPortType)# showLacpInternal commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example displays the LACP internal state information on the UCS controller:

```
UCS(ProvisionLacpPortType)# showLacpInternal lacpPhysicalPort 2
UCS(ProvisionLacpPortType)# showLacpInternal review
Commands in queue:
        showLacpNeighbors lacpPhysicalPort 1
        showLacpInternal lacpPhysicalPort 1
UCS(ProvisionLacpPortType)# showLacpInternal commit
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].portNumber =1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].mode = false
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].key = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].role = true
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].timeout = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].portPriority= 32768
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].adminKey = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].operKey = 3
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[0].collectorMaxDelay = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].portNumber =2
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].mode = false
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].key = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].role = true
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].timeout = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].portPriority= 26733
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].adminKey = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].operKey = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[1].collectorMaxDelay = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].portNumber =3
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].mode = false
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].key = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].role = true
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].timeout = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].portPriority= 32768
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].adminKey = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].operKey = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[2].collectorMaxDelay = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].portNumber =4
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].mode = false
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].key = 0
```

```
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].role = true
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].timeout = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].portPriority= 32768
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].adminKey = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].operKey = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[3].collectorMaxDelay = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].portNumber =5
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].mode = false
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].key = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].role = true
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].timeout = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].portPriority= 32768
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].adminKey = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].operKey = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[4].collectorMaxDelay = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].portNumber =6
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].mode = false
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].key = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].role = true
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].timeout = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].portPriority= 32768
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].adminKey = 0
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].operKey = 1
ShowLacpInternal_Output.lacpPortInternals.lacpPortInternalslist[5].collectorMaxDelay = 0

 ShowLacpInternal Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Viewing the LACP Neighbors Status Information on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLacpNeighbors** {**commit** | **flush** | **lacpPhysicalPort***port_num* | **review**}
	
	Example:
	
	`UCS(ProvisionLacpPortType)# showLacpNeighbors ?`
	` commit             commit showLacpNeighbors`
	` flush              flush all showLacpNeighbors`
	`commands from queue`
	` lacpPhysicalPort   Show LACP neighbor status request`
	` review             review showLacpNeighbors commands`
	`UCS(ProvisionLacpPortType)# showLacpNeighbors`
	`lacpPhysicalPort 2`
		Displays LACP neighbor state information.
		
		• **commit—**Sends the LACP configuration to NID.
		• **flush—**Flushes all LACP configuration from the queue.
		• **lacpPhysicalPort***port_num***—**Displays the LACP neighbors information for specified port on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showLacpNeighbors review**
	
	Example:
	
	`UCS(ProvisionLacpPortType)# showLacpNeighbors review`
	`Commands in queue:`
	`        showLacpNeighbors lacpPhysicalPort 2`
		Displays the LACP configuration on the UCS controller.

	Command or Action	Purpose
Step 3	**showLacpNeighbors commit** **Example:** `UCS(ProvisionLacpPortType)# showLacpNeighbors commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)# exit`	Exits the LACP provisioning mode.

Configuration Example

The example displays the LACP neighbors status information on the UCS controller:

```
UCS(ProvisionLacpPortType)# showLacpNeighbors lacpPhysicalPort 2
UCS(ProvisionLacpPortType)# showLacpNeighbors review
Commands in queue:
        showLacpNeighbors lacpPhysicalPort 2
UCS(ProvisionLacpPortType)# showLacpNeighbors commit
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].aggrID = 1
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].partnerSysId =
'00-3a-99-fd-4a-44'
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].partnerPort = 3
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].partnerPortPriority
= 32768
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].partnerSysPriority =
 32768
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].partnerOperKey = 3
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].aggrProtocolType =
'LACP'
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].bandwidth = 0
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[0].aggrMacAddr =
'00-3a-99-fd-4a-3b'
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].aggrID = 1
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].partnerSysId =
'00-3a-99-fd-4a-44'
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].partnerPort = 6
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].partnerPortPriority
= 32768
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].partnerSysPriority =
 32768
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].partnerOperKey = 3
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].aggrProtocolType =
'LACP'
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].bandwidth = 0
ShowLacpNeighbors_Output.lacpNeighborStatus.lacpNeighborStatusList[1].aggrMacAddr =
'00-3a-99-fd-4a-3e'

ShowLacpNeighbors Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

Viewing the LACP Statistics on the UCS controller

Before You Begin

- Perform the steps to provision LACP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLacpStats** {**commit** | **flush** | **lacpPhysicalPort***port_num* | **review**}
	
	Example:
	`UCS(ProvisionLacpPortType)# showLacpStats ?`
	` commit            commit showLacpStats`
	` flush             flush all showLacpStats commands from queue`
	` lacpPhysicalPort  Show LACP statistics request`
	` review            review showLacpStats commands`
	`UCS(ProvisionLacpPortType)# showLacpStats lacpPhysicalPort 2`
		Displays the LACP statistics.
		• **commit**—Sends the LACP configuration.
		• **flush**—Flushes all LACP configuration from the queue.
		• **lacpPhysicalPort***port_num*—Displays the LACP statistics for specified port on the UCS controller.
		• **review**—Displays the configuration on the UCS controller.

Step 2	**showLacpStats review**
	
	Example:
	`UCS(ProvisionLacpPortType)# showLacpStats review`
	`Commands in queue:`
	`        showLacpStats lacpPhysicalPort 1`
	`        showLacpStats lacpPhysicalPort 2`
		Displays the LACP configuration on the UCS controller.

Step 3	**showLacpStats commit** **Example:** `UCS(ProvisionLacpPortType)# showLacpStats commit`	Sends the LACP configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionLacpPortType)#exit`	Exits the LACP provisioning mode.

Configuration Example

The example displays the LACP statistics on the UCS controller:

```
UCS(ProvisionLacpPortType)# showLacpStats lacpPhysicalPort 2
UCS(ProvisionLacpPortType)# showLacpStats review
Commands in queue:
        showLacpStats lacpPhysicalPort 1
        showLacpStats lacpPhysicalPort 2
UCS(ProvisionLacpPortType)# showLacpStats commit
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[0].rxUnknown = 0
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[0].port = 3
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[0].rxFrames = 17866
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[0].txFrames = 12527
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[0].rxIllegal = 0
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[1].rxUnknown = 0
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[1].port = 6
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[1].rxFrames = 17244
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[1].txFrames = 12132
ShowLacpStats_Output.lacpPortStatistics.lacpPortStatsList[1].rxIllegal = 0
```

```
 ShowLacpStats Commit Success!!!
UCS(ProvisionLacpPortType)# exit
```

CHAPTER 12

Provisioning Link Layer Discovery Protocol

The Cisco Discovery Protocol (CDP) is a device discovery protocol that runs over Layer 2 (the data link layer) on all Cisco-manufactured devices (routers, bridges, access servers, and switches). CDP allows network management applications to automatically discover and learn about other Cisco devices connected to the network.

To support non-Cisco devices and to allow for interoperability between other devices, the switch supports the IEEE 802.1AB Link Layer Discovery Protocol (LLDP). LLDP is a neighbor discovery protocol that is used for network devices to advertise information about themselves to other devices on the network. This protocol runs over the data link layer, which allows two systems running different network layer protocols to learn about each other.

LLDP supports a set of attributes that it uses to discover neighbor devices. These attributes contain type, length, and value descriptions and are referred to as TLVs. LLDP supported devices can use TLVs to receive and send information to their neighbors. Details such as configuration information, device capabilities, and device identity can be advertised using this protocol.

By default, LLDP is disabled globally and on interfaces.

The switch supports these basic management TLVs. These are mandatory LLDP TLVs.

- Port description TLV
- System name TLV
- System description
- System capabilities TLV
- Management address TLV

These organizationally-specific LLDP TLVs are also advertised to support LLDP-MED.

- Port VLAN ID TLV (IEEE 802.1 organizationally specific TLVs)
- MAC/PHY configuration/status TLV (IEEE 802.3 organizationally specific TLVs)

How To Configure LLDP

Setting LLDP Global Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.

Step 3	**setLldpConfig lldpGlobalConfiguration** {**global_state** {**enable** | **disable**} |**hold_time***lldp_hold_time* |**lldp_transmission_delay** *value* |**reinit_delay** *tx_reinit_value* **timer***tx_value* |**tlv_select** {**mgmt_address** | **port_description** | **system_capabilities** | **system_description** | **system_name**}
	
	Example:
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration global_state enable`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration hold_time 5`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration lldp_transmission_delay 10`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration timer 10`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration reinit_delay 10`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration tlv_select system_description enable`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration tlv_select port_description enable`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration tlv_select management_address enable`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration tlv_select system_capabilities enable`
	`UCS(ProvisionLldpPortType)# setLldpConfig lldpGlobalConfiguration tlv_select system_name enable`
		Sets the LLDP global configuration.
		• **global_state—**LLDP global state. This state is either *enabled* or *disabled*.
		• **hold_time—**LLDP hold time before discarding the configuration. The valid values are from 2 to 10 seconds. The default value is 4 seconds.
		• **lldp_transmission_delay—**LLD Transmission delay value. The valid values are from 1 to 8192. The default value is 2 seconds.
		• **reinit_delay—**LLDP transmission re-initialization delay. The valid values are from 1 to 10 seconds. The default value is 2 seconds.
		• **timer—**Time between each LLDP frame transmitted in seconds. The valid values are from 5 to 32768. The default value is 30 seconds.
		• **tlv_select—**Transmission TLV.

Step 4	**setLldpConfig review** **Example:** `UCS(ProvisionLldpPortType)# setLldpConfig review`	Reviews the setLldpConfig.

	Command or Action	Purpose
Step 5	**setLldpConfig commit** **Example:** `UCS(ProvisionLldpPortType)# setLldpConfig commit`	Sends the setLldpConfig configuration to the Cisco ME 1200 NID.

Step 6	**exit**
	
	Example:
	`UCS(ProvisionLldpPortType)# exit`
	`UCS(ProvisionLldpPortType)#`
		Exits from the provisionLldpPortType mode.

What to Do Next

After the configuration is sent to the Cisco ME 1200 NID, use the following **get** command to view the setLldpConfig configuration.

```
UCS(ProvisionLldpPortType)# getLldpConfig getLldpConfigRequest
UCS(ProvisionLldpPortType)# getLldpConfig review
```

```
Commands in queue:
                 getLldpConfig getLldpConfigRequest
```

```
UCS(ProvisionLldpPortType)# getLldpConfig commit
```

```
GetLldpConfig_Output.lldpGlobalConfiguration.global_state = true
GetLldpConfig_Output.lldpGlobalConfiguration.hold_time = 5
GetLldpConfig_Output.lldpGlobalConfiguration.timer = 10
GetLldpConfig_Output.lldpGlobalConfiguration.tlv_select.system_name =
true
GetLldpConfig_Output.lldpGlobalConfiguration.tlv_select.system_description
 = true
GetLldpConfig_Output.lldpGlobalConfiguration.tlv_select.port_description
 = true
GetLldpConfig_Output.lldpGlobalConfiguration.tlv_select.management_address
 = true
GetLldpConfig_Output.lldpGlobalConfiguration.tlv_select.system_capabilities
 = true
GetLldpConfig_Output.lldpGlobalConfiguration.reinit_delay = 10
GetLldpConfig_Output.lldpGlobalConfiguration.lldp_transmission_delay =
10

GetLldpConfig Commit Success!!!
```

Setting LLDP Configuration to Default

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**setLldpDefaults setLldpDefaultsRequest** **Example:** `UCS(ProvisionLldpPortType)# setLldpDefaults setLldpDefaultsRequest`	Sets the LLDP configuration to default values.
Step 4	**setLldpDefaults commit** **Example:** `UCS(ProvisionLldpPortType)# setLldpDefaults commit`	Sends the setLldpDefaults configuration to the Cisco ME 1200 NID.

Step 5	**exit**
	
	Example:
	`UCS(ProvisionLldpPortType)# exit`
	`UCS(ProvisionLldpPortType)#`
		Exits from the ProvisionLldpPortType mode.

What to Do Next

After the configuration is sent to the Cisco ME 1200 NID, use the following **get** command to view the setLldpDefaults configuration.

```
UCS(ProvisionLldpPortType)# getLldpDefaults getLldpDefaultsRequest
UCS(ProvisionLldpPortType)# getLldpDefaults review
```

```
Commands in queue:
                getLldpDefaults getLldpDefaultsRequest
```

```
UCS(ProvisionLldpPortType)# getLldpDefaults commit
```

```
GetLldpDefaults_Output.lldpGlobalConfiguration.global_state = true
GetLldpDefaults_Output.lldpGlobalConfiguration.hold_time = 5
GetLldpDefaults_Output.lldpGlobalConfiguration.timer = 30
GetLldpDefaults_Output.lldpGlobalConfiguration.tlv_select.system_name =
true
GetLldpDefaults_Output.lldpGlobalConfiguration.tlv_select.system_description
```

```
 = true
GetLldpDefaults_Output.lldpGlobalConfiguration.tlv_select.port_description
 = true
GetLldpDefaults_Output.lldpGlobalConfiguration.tlv_select.management_address
 = true
GetLldpDefaults_Output.lldpGlobalConfiguration.tlv_select.system_capabilities
 = true
GetLldpDefaults_Output.lldpGlobalConfiguration.reinit_delay = 2
GetLldpDefaults_Output.lldpGlobalConfiguration.lldp_transmission_delay =
 10

GetLldpDefaults Commit Success!!!
```

Setting LLDP Port Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.

Step 3	**setLldpportconfig lldpPortConfiguration** {**lldp_receive_enable** {**disable** | **enable**} |**lldp_transmit_enable** {**disable** | **enable**} | **port_number***port_number*}
	
	Example:
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration port_number 3`
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_receive_enable disable`
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_transmit_enable disable`
		Sets the LLDP port configuration.
		• **lldp_receive_enable—**Whether LLDP receive is enabled or disabled.
		• **lldp_transmit_enable—**Whether LLDP transmit is enabled or disabled.
		• **port_number—**The target interface number. The valid values are from 1 to 6.

Step 4	**setLldpPortConfig review** **Example:** `UCS(ProvisionLldpPortType)# setLldpPortConfig review`	Reviews the setLldpPortConfig.
Step 5	**setLldpPortConfig commit** **Example:** `UCS(ProvisionLldpPortType)# setLldpConfig commit`	Sends the setLldpConfig configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(ProvisionLldpPortType)# exit`	Exits from the ProvisionLldpPortType mode.

What to Do Next

After the configuration is sent to the Cisco ME 1200 NID, use the following **get** command to view the setLldpPortConfig configuration.

```
UCS(ProvisionLldpPortType)# getLldpportConfig physicalPortNum 3
UCS(ProvisionLldpPortType)# getLldpportConfig review
```

```
Commands in queue:
                getLldpConfig physicalPortNum 3
```

```
UCS(ProvisionLldpPortType)# getLldpportConfig commit
```

```
GetLldpPortConfig_Output.lldpPortConfiguration.port_number = 3
GetLldpPortConfig_Output.lldpPortConfiguration.lldp_transmit_enable =
false
GetLldpPortConfig_Output.lldpPortConfiguration.lldp_receive_enable = false


GetLldpPortConfig Commit Success!!!
```

Setting LLDP Port Configuration to Default

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**setlldpportdefaults physicalPortNum***port_number* **Example:** `UCS(ProvisionLldpPortType)# setlldpportdefaults physicalPortNum 3`	Sets the LLDP port configuration to default values. • **physicalPortNum—**Port number for which the LLDP configuration is set to default. The valid values are from 1 to 6.
Step 4	**setlldpportdefaults commit** **Example:** `UCS(ProvisionLldpPortType)# setlldpportdefaults commit`	Sends the setlldpportdefaults configuration to the Cisco ME 1200 NID.

	Command or Action	Purpose
Step 5	**exit** **Example:** `UCS(ProvisionLldpPortType)# exit`	Exits from the ProvisionLldpPortType mode.

What to Do Next

After the configuration is sent to the Cisco ME 1200 NID, use the following **get** command to view the setlldpportdefaults configuration.

```
UCS(ProvisionLldpPortType)# getLldpportdefaults physicalPortNum 3
UCS(ProvisionLldpPortType)# getLldpportdefaults review
```

```
Commands in queue:
                getLldpportdefaults physicalPortNum 3
```

```
UCS(ProvisionLldpPortType)# getLldpportdefaults commit
```

```
GetLldpPortDefaults_Output.lldpPortConfiguration.port_number = 3
GetLldpPortDefaults_Output.lldpPortConfiguration.lldp_transmit_enable =
true
GetLldpPortDefaults_Output.lldpPortConfiguration.lldp_receive_enable =
true

GetLldpPortDefaults Commit Success!!!
```

Other Commands For LLDP Configuration

Clearing LLDP Counters

clearLldpCounters

```
UCS(ProvisionLldpPortType)# clearLldpCounters physicalPortNum 3
```

Displaying LLDP Neighbors

showlldpneighbors physicalPortNum*physical_port_number*

```
UCS(ProvisionLldpPortType)# showlldpneighbors physicalPortNum 3
UCS(ProvisionLldpPortType)# showlldpneighbors commit
```

```
ShowLldpNeighbors_Output.lldpNeighborInformation.local_port_id = 3
ShowLldpNeighbors_Output.lldpNeighborInformation.chassis_id =
'18-9C-5D-A7-F4-1C'
ShowLldpNeighbors_Output.lldpNeighborInformation.remote_port_id = 'Gi0/3'
ShowLldpNeighbors_Output.lldpNeighborInformation.remote_port_description
 = 'GigabitEthernet0/3'
ShowLldpNeighbors_Output.lldpNeighborInformation.remote_system_name =
'IRF-Whales-1'
ShowLldpNeighbors_Output.lldpNeighborInformation.remote_system_capabilities
 = 'Bridge(+), Router(+)'
ShowLldpNeighbors_Output.lldpNeighborInformation.remote_system_description
```

```
 = 'Cisco IOS Software, ME360x Software (ME360x-UNIVERSAL-M), Version
15.4(2)SN, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2014 by Cisco Systems, Inc.
Compiled Fri 21-Mar-14 09:12 by prod_rel_team'
ShowLldpNeighbors_Output.lldpNeighborInformation.remote_management_IP =
'7.3.9.13 (IPv4)'
ShowLldpNeighbors_Output.lldpNeighborInformation.remote_management_IPv6
= ''

ShowLldpNeighbors Commit Success!!!
```

Displaying LLDP Statistics

showlldpstatistics physicalPortNum*physical_port_number*

```
UCS(ProvisionLldpPortType)# showlldpstatistics physicalPortNum 3
UCS(ProvisionLldpPortType)# showlldpstatistics commit
```

```
ShowLldpStatistics_Output.lldpPortStatistics.global_counters.total_neighbor_entries_added
 = 1
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.Tx_Frames =
17
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.Rx_Frames =
0
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.Rx_Errors =
0
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.Rx_Frames_Discarded
 = 0
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.TLVs_Discarded
 = 0
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.TLVs_Unrecognized
 = 0
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.Org_Discarded
 = 0
ShowLldpStatistics_Output.lldpPortStatistics.local_counters.Age_Outs = 0

ShowLldpStatistics Commit Success!!!
```

CHAPTER 13

Configuring SNMP

This document describes the Simple Network Management Protocol (SNMP) feature and configuration steps to implement network management using SNMP.

Prerequisites for Configuring SNMP

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.

Information About SNMP

SNMP is an application layer protocol that facilitates the exchange of management information among network devices, such as nodes and routers. It comprises part of the TCP/IP suite. System administrators can remotely manage network performance, find and solve network problems, and plan for network growth by using SNMP.

How to Provision SNMP

Configuring and Retrieving Default SNMP Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionSnmpConf** **Example:** `UCS# ProvisionSnmpConf`	Enters the ProvisionSnmpConf mode.
Step 3	**setSnmpDefaultConf setSnmpDefaultConfigRequest***request_id* **Example:** `UCS(ProvisionSnmpConf)# setSnmpDefaultConf setSnmpDefaultConfigRequest 1`	Configures SNMP default configuration status. • **setSnmpDefaultConfigRequest—**Specifies SNMP default configuration. • *request_id*—Request ID. The default value is 1.
Step 4	**getSnmpDefaultConf getSnmpDefaultConfRequest** **Example:** `UCS(ProvisionSnmpConf)# getSnmpDefaultConf getSnmpDefaultConfRequest`	Retrieves SNMP default configuration status. • **getSnmpDefaultConfRequest—**Retrieves SNMP default configuration.
Step 5	**getSnmpDefaultConf review** **Example:** `UCS(ProvisionSnmpConf)# getSnmpDefaultConf review`	Displays the configuration.
Step 6	**getSnmpDefaultConf commit** **Example:** `UCS(ProvisionSnmpConf)# getSnmpDefaultConf commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(ProvisionSnmpConf)# exit`	Exits from the ProvisionSnmpConf mode.

Configuration Example

The example shows how to configure and retrieve Default SNMP Configurations:

```
UCS(ProvisionSnmpConf)# setSnmpDefaultConf  setSnmpDefaultConfigRequest 1
UCS(ProvisionSnmpConf)# getSnmpDefaultConf  getSnmpDefaultConfRequest

UCS(ProvisionSnmpConf)# getSnmpDefaultConf review
UCS(ProvisionSnmpConf)# getSnmpDefaultConf commit
UCS(ProvisionSnmpConf)# exit
```

The following is a sample output on the NID.

```
UCS(ProvisionSnmpConf)#getSnmpDefaultConf  getSnmpDefaultConfRequest
UCS(ProvisionSnmpConf)#
UCS(ProvisionSnmpConf)#getSnmpDefaultConf  review
Commands in queue:
getSnmpDefaultConf getSnmpDefaultConfRequest
UCS(ProvisionSnmpConf)#getSnmpDefaultConf  commit
GetSnmpDefaultConf_Output.getSnmpDefaultResponse.trap = false
GetSnmpDefaultConf_Output.getSnmpDefaultResponse.version = 'v2c'
GetSnmpDefaultConf_Output.getSnmpDefaultResponse.snmp_server = 'disable'

GetSnmpDefaultConf Commit Success!!!
```

Configuring SNMPv2c Community Parameters

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionSnmpConf** **Example:** `UCS# ProvisionSnmpConf`	Enters the ProvisionSnmpConf mode.

Step 3	**setSnmpServerConfsnmsnp_server_conf** { **trap** {**true** | **false**} | **version** {**v1** | **v2c** | **v3**} | **snmp_server** {**enable** | **disable**} | **community_v2c** {**comm_name** *comm_name* | **mode** {**ro** | **rw**}}}
	
	Example:
	`UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf  snmp_server  enable`
	`UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf version  v2c`
	`UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf trap true`
	`UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf community_v2c comm_name Public`
	`UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf community_v2c mode ro`
		Configures SNMP server.
		• **trap—**Specifies SNMP traps.
		• **true—**Enables SNMP trap.
		• **false—**Disables SNMP trap.
		• **version—**Specifies SNMP host version.
		• **v1—**Specifies SNMP version v1.
		• **v2c—**Specifies SNMP version v2c.
		• **v3—**Specifies SNMP version v3.
		• **snmp_server—**Specifies the SNMP server.

	Command or Action	Purpose
		• **Enable**—Enables the SNMP server. • **Disable**—Disables the SNMP server. • **community_v2c**—Specifies the v2c community. • **comm_name**—Specifies the v2c community name. • *comm_name*—v2c community name. • **mode**—Specifies read or write mode. • **ro**—Read mode. • **rw**—Write mode.
Step 4	**setSnmpServerConf review** **Example:** `UCS(ProvisionSnmpConf)# setSnmpServerConf review`	Displays the configuration.
Step 5	**setSnmpServerConf commit** **Example:** `UCS(ProvisionSnmpConf)# setSnmpServerConf commit`	Sends the configuration to NID.
Step 6	**getSnmpServerConf getSnmpServerConfigRequest** **Example:** `UCS(ProvisionSnmpConf)# getSnmpServerConf getSnmpServerConfigRequest`	Retrieves SNMP server configuration.
Step 7	**getSnmpServerConf review** **Example:** `UCS(ProvisionSnmpConf)# getSnmpServerConf review`	Displays the configuration.
Step 8	**getSnmpServerConf commit** **Example:** `UCS(ProvisionSnmpConf)# getSnmpServerConf commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(ProvisionSnmpConf)# exit`	Exits from the ProvisionSnmpConf mode.

Configuration Example

The example shows how to configure and retrieve SNMPv2c community parameters:

```
UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf snmp_server  enable
UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf version  v2c
```

```
UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf trap true
UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf community_v2c comm_name Public
UCS(ProvisionSnmpConf)# setSnmpServerConf snmp_server_conf community_v2c mode ro
UCS(ProvisionSnmpConf)# setSnmpServerConf review
UCS(ProvisionSnmpConf)# setSnmpServerConf commit

UCS(ProvisionSnmpConf)# getSnmpServerConf getSnmpServerConfigRequest
UCS(ProvisionSnmpConf)# getSnmpServerConf review
UCS(ProvisionSnmpConf)# getSnmpServerConf commit
UCS(ProvisionSnmpConf)# exit
```

The following is a sample output on the NID.

```
UCS(ProvisionSnmpConf)#getSnmpServerConf commit
GetSnmpServerConf_Output.snmp_server_conf.trap = true
GetSnmpServerConf_Output.snmp_server_conf.version.t = 2
GetSnmpServerConf_Output.snmp_server_conf.version.u.v2c = 'v2c'
GetSnmpServerConf_Output.snmp_server_conf.snmp_server.t = 1
GetSnmpServerConf_Output.snmp_server_conf.snmp_server.u.enable = 'enabled
GetSnmpServerConf_Output.snmp_server_conf.community_v2c.comm_name =
'Public'
GetSnmpServerConf_Output.snmp_server_conf.community_v2c.mode.t = 1
GetSnmpServerConf_Output.snmp_server_conf.community_v2c.mode.u.ro = 'RO'

GetSnmpServerConf Commit Success!!!
```

Configuring SNMPv3 Community

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionSnmpConf** **Example:** `UCS# ProvisionSnmpConf`	Enters the ProvisionSnmpConf mode.

Step 3	**setSnmpCommunitysnmp_community_conf** {**community_name** *name* | **community_ip***ip_address* |**community_mask** *mask*}
	
	Example:
	`UCS(ProvisionSnmpConf)# setSnmpCommunity snmp_community_conf  community_ip 10.10.10.1`
	`UCS(ProvisionSnmpConf)ProvisionSnmpConf)# setSnmpCommunity  snmp_community_conf  community_mask  255.255.255.0`
	`UCS(ProvisionSnmpConf)ProvisionSnmpConf)# setSnmpCommunity  snmp_community_conf  community_name  user12`
		Configures SNMP default configuration status.
		• **community_name—**Configures SNMP community string.
		• *name*—Name of the community.
		• **community_ip—**Specifies community IP.
		• *ip_address*—IP address.
		• **community_mask—**Specifies community mask.
		• *mask*—Mask address.

	Command or Action	Purpose
Step 4	**getSnmpCommunity getSnmpCommunityConfRequest** **Example:** `UCS(ProvisionSnmpConf)# getSnmpCommunity getSnmpCommunityConfRequest`	Configures SNMP default configuration status. • **getSnmpCommunityConfRequest—**Retrieves SNMP configuration information.
Step 5	**getSnmpCommunity review** **Example:** `UCS(ProvisionSnmpConf)# getSnmpCommunity review`	Displays the configuration.
Step 6	**getSnmpCommunity commit** **Example:** `UCS(ProvisionSnmpConf)# getSnmpCommunity commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(ProvisionSnmpConf)# exit`	Exits from the ProvisionSnmpConf mode.

Configuration Example

The example shows how to configure and retrieve SNMPv3 Community:

```
UCS(ProvisionSnmpConf)# setSnmpCommunity  snmp_community_conf  community_ip 10.10.10.1
UCS(ProvisionSnmpConf)# setSnmpCommunity  snmp_community_conf  community_mask 255.255.255.0
UCS(ProvisionSnmpConf)# setSnmpCommunity  snmp_community_conf  community_name user12
UCS(ProvisionSnmpConf)# getSnmpCommunity  getSnmpCommunityConfReques

UCS(ProvisionSnmpConf)# getSnmpCommunity review
UCS(ProvisionSnmpConf)# getSnmpCommunity commit
UCS(ProvisionSnmpConf)# exit
```

The following is a sample output on the NID.

Configuring Trap Destination

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionSnmpConf** **Example:** `UCS# ProvisionSnmpConf`	Enters the ProvisionSnmpConf mode.

	Command or Action	Purpose

Step 3	**setSnmpHostsnmp_host_config** {**host_id***host_id* |**set_trap** {**enable** | **disable**} | **version** {**v1** | **v2c** | **v3**}**udp_port_no***udp_port_no* | **address** {**ipv4***ipv4_address* | **ipv6***ipv6_address*} | **inform_mode** {**enable** | **disable**} | **inf_retries***retry_instances* | **timeout_inform***timeout_inform_value*}
	
	Example:
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config address ipv4 10.106.212.248`
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config host_id trap-config`
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config inf_retries 255`
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config inform_mode enable`
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config set_trap enable`
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config timeout_inform 2147`
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config udp_port_no 162`
	`UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config version v2c`
		Configures SNMP host.
		• **host_id**—Specifies the host name.
		• *host_id*—Host name.
		• **set_trap**—Specifies the trap.
		• **enable**—Enables the trap.
		• **disable**—Disables the trap.
		• **version**—Specifies SNMP host version.
		• **v1**—Specifies SNMP version v1.
		• **v2c**—Specifies SNMP version v2c.
		• **v3**—Specifies SNMP version v3.
		• **udp_port_no**—Specifies the host port number.
		• *udp_port_no*—Host port number.
		• **address**—Specifies the IP address.
		• **ipv4**—Specifies IPv4 address.
		• *ipv4_address*— IPv4 address.
		• **ipv6**—Specifies IPv6 address.
		• *ipv6_address*—IPv6 address.
		• **inform_mode**—Specifies inform mode.
		• **enable**—Enables inform mode.
		• **disable**—Disables inform mode.
		• **inf_retries**—Specifies inform retries.
		• *retry_instances*—Inform retry number.
		• **timeout_inform**—Specifies timeout inform.
		• *timeout_inform_value*—Timeout inform value.

Step 4	**setSnmpHost review** **Example:** `UCS(ProvisionSnmpConf)# setSnmpHost review`	Displays the configuration.
Step 5	**setSnmpHost commit** **Example:** `UCS(ProvisionSnmpConf)# setSnmpHost commit`	Sends the configuration to NID.

	Command or Action	Purpose
Step 6	**getSnmpHost getSnmpHostRequest** **Example:** `UCS(ProvisionSnmpConf)# getSnmpHost getSnmpHostRequest`	Retrieves SNMP host configuration information.
Step 7	**getSnmpHost review** **Example:** `UCS(ProvisionSnmpConf)# getSnmpHost review`	Displays the configuration.
Step 8	**getSnmpHost commit** **Example:** `UCS(ProvisionSnmpConf)# getSnmpHost commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(ProvisionSnmpConf)# exit`	Exits from the ProvisionSnmpConf mode.

Configuration Example

The example shows how to configure and retrieve trap destination:

```
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config address ipv4 10.106.212.248
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config host_id trap-config
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config inf_retries 255
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config inform_mode enable
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config set_trap enable
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config timeout_inform 2147
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config udp_port_no 162
UCS(ProvisionSnmpConf)# setSnmpHost snmp_host_config version v2c
UCS(ProvisionSnmpConf)# setSnmpHost review
UCS(ProvisionSnmpConf)# setSnmpHost commit

UCS(ProvisionSnmpConf)# getSnmpHost getSnmpHostRequest
UCS(ProvisionSnmpConf)# getSnmpHost review
UCS(ProvisionSnmpConf)# getSnmpHost commit
UCS(ProvisionSnmpConf)# exit
```

Configuring an Entry in SNMP View List

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionSnmpConf** **Example:** `UCS# ProvisionSnmpConf`	Enters the ProvisionSnmpConf mode.

Step 3	**setSnmpView setSnmpViewRequest** {**view-nameview_name** |**view_oid***oid_value* |**view_type** {**included** | **excluded**}}
	
	Example:
	`UCS(ProvisionSnmpConf)# setSnmpView setSnmpViewRequest  view_type  included`
	`UCS(ProvisionSnmpConf)# setSnmpView setSnmpViewRequest  view_oid .1.3.6.1.2.1.31`
	`UCS(ProvisionSnmpConf)# setSnmpView setSnmpViewRequest  view_name ifMIB`
		Configures SNMP view.
		• **view_name—**Specifies view name.
		• **view_oid—**Specifies object identifier.
		• *oid_value*—Object identifier value.
		• **view_type—**Specifies view type.
		• **included—**Includes view type.
		• **excluded—**Excludes view type.

Step 4	**setSnmpView review** **Example:** `UCS(ProvisionSnmpConf)# setSnmpView review`	Displays the configuration.
Step 5	**setSnmpView commit** **Example:** `UCS(ProvisionSnmpConf)# setSnmpView commit`	Sends the configuration to NID.
Step 6	**getSnmpView getSnmpViewRequest** **Example:** `UCS(ProvisionSnmpConf)# getSnmpView getSnmpViewRequest`	Retrieves SNMP default configuration status. • **getSnmpViewRequest—**Retrieves SNMP default configuration.
Step 7	**getSnmpView review** **Example:** `UCS(ProvisionSnmpConf)# getSnmpView review`	Displays the configuration.
Step 8	**getSnmpView commit** **Example:** `UCS(ProvisionSnmpConf)# getSnmpView commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(ProvisionSnmpConf)# exit`	Exits from the ProvisionSnmpConf mode.

Configuration Example

The example shows how to configure and retrieve an entry in SNMP view list:

```
UCS(ProvisionSnmpConf)# setSnmpView  setSnmpViewRequest  view_type  included
UCS(ProvisionSnmpConf)# setSnmpView  setSnmpViewRequest  view_oid .1.3.6.1.2.1.31
UCS(ProvisionSnmpConf)# setSnmpView  setSnmpViewRequest  view_name ifMIB
UCS(ProvisionSnmpConf)# setSnmpView  review
UCS(ProvisionSnmpConf)# setSnmpView  commit

UCS(ProvisionSnmpConf)# getSnmpView  getSnmpViewRequest
UCS(ProvisionSnmpConf)# getSnmpView  review
UCS(ProvisionSnmpConf)# getSnmpView  commit
UCS(ProvisionSnmpConf)# exit
```

The following is a sample output on the NID.

```
UCS(ProvisionSnmpConf)#
UCS(ProvisionSnmpConf)#getSnmpView  getSnmpViewRequest
UCS(ProvisionSnmpConf)#getSnmpView  review
Commands in queue:
getSnmpView getSnmpViewRequest
UCS(ProvisionSnmpConf)#getSnmpView  commit
GetSnmpView_Output.getSnmpViewResponse[0].view_name = 'ifMIB'
GetSnmpView_Output.getSnmpViewResponse[0].view_oid = '.1.3.6.1.2.1.31'
GetSnmpView_Output.getSnmpViewResponse[0].view_type = 'included'
GetSnmpView_Output.getSnmpViewResponse[1].view_name = 'default_view'
GetSnmpView_Output.getSnmpViewResponse[1].view_oid = '.1'
GetSnmpView_Output.getSnmpViewResponse[1].view_type = 'included'
```

Creating an SNMPv3 User

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionSnmpConf** **Example:** `UCS# ProvisionSnmpConf`	Enters the ProvisionSnmpConf mode.

Step 3	**setSnmpUserConf setSnmpUsersRequest** {**user_name***user_name* | **user_engine_id** *user_engine_id* | **protocolauth** {**md5** | **sha**} | **protocol_passwd***protocol_passwd* | **priviledgeauth** {**aes** | **des**} | priv_passwd *priv_passwd*}
	
	Example:
	`UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest  priv_passwd 12345678`
	`UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest priviledgeauth aes`
	`UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest protocol_passwd 12345678`
	`UCS(ProvisionSnmpConf)# setSnmpUserConf`
		Configures an SNMP user.
		• **user_name—**Specifies user name.
		• *user_name*—User name.
		• **user_engine_id—**Specifies user engine ID.
		• *user_engine_id*—User engine ID.
		• **protocolauth—**Specifies authentication protocol.
		• **md5—**Specifies MD5 authentication protocol.

	Command or Action	Purpose

	`setSnmpUsersRequest protocolauth  md5`
	`UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest user_engine_id 800007e5017f000001`
	`UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest user_name user1`
		• **sha—**Specifies SHA authentication protocol.
		• **protocol_passwd—**Specifies protocol password.
		• *protocol_passwd*—Protocol password.
		• **priviledgeauth—**Specifies privilege authentication type.
		• **aes—**Specifies AES authentication.
		• **des—**Specifies DES authentication.
		• **priv_passwd—**Specifies privacy password.
		• *priv_passwd*—Privacy password.

Step 4	**setSnmpUserConf review** **Example:** `UCS(ProvisionSnmpConf)# setSnmpUserConf review`	Displays the configuration.
Step 5	**setSnmpUserConf commit** **Example:** `UCS(ProvisionSnmpConf)# setSnmpUserConf commit`	Sends the configuration to NID.
Step 6	**getSnmpUserConf getSnmpUserConfRequest** **Example:** `UCS(ProvisionSnmpConf)#  getSnmpUserConf getSnmpUserConfRequest`	Retrieves SNMP user configuration.
Step 7	**getSnmpUserConf review** **Example:** `UCS(ProvisionSnmpConf)# getSnmpUserConf review`	Displays the configuration.
Step 8	**getSnmpUserConf commit** **Example:** `UCS(ProvisionSnmpConf)# getSnmpUserConf commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(ProvisionSnmpConf)# exit`	Exits the ProvisionSnmpConf mode.

Configuration Example

The example shows how to create and retrieve an SNMPv3 user:

```
UCS(ProvisionSnmpConf)# setSnmpUserConf  setSnmpUsersRequest  priv_passwd 12345678
UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest priviledgeauth aes
```

```
UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest protocol_passwd 12345678
UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest protocolauth  md5
UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest user_engine_id 800007e5017f000001
UCS(ProvisionSnmpConf)# setSnmpUserConf setSnmpUsersRequest user_name user1
UCS(ProvisionSnmpConf)# setSnmpUserConf review
UCS(ProvisionSnmpConf)# setSnmpUserConf commit

UCS(ProvisionSnmpConf)# getSnmpUserConf getSnmpUserConfRequest
UCS(ProvisionSnmpConf)# getSnmpUserConf review
UCS(ProvisionSnmpConf)# getSnmpUserConf commit
UCS(ProvisionSnmpConf)# exit
```

The following is a sample output on the NID.

```
UCS(ProvisionSnmpConf)#$  getSnmpUserConfRequest
UCS(ProvisionSnmpConf)#getSnmpUserConf  review
Commands in queue:
getSnmpUserConf getSnmpUserConfRequest
UCS(ProvisionSnmpConf)#getSnmpUserConf  commit
GetSnmpUserConf_Output.getSnmpUserConfResponse[0].user_name = 'user1'
GetSnmpUserConf_Output.getSnmpUserConfResponse[0].engine_id =
'800007e5017f000001'
GetSnmpUserConf_Output.getSnmpUserConfResponse[0].protocol = 'MD5'
GetSnmpUserConf_Output.getSnmpUserConfResponse[0].priv = 'DES'
GetSnmpUserConf_Output.getSnmpUserConfResponse[1].user_name =
'default_user'
GetSnmpUserConf_Output.getSnmpUserConfResponse[1].engine_id =
'800007e5017f000001'
GetSnmpUserConf_Output.getSnmpUserConfResponse[1].protocol = 'None'
GetSnmpUserConf_Output.getSnmpUserConfResponse[1].priv = 'None'

GetSnmpUserConf Commit Success!!!
```

Creating an SNMP User

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionSnmpConf** **Example:** `UCS# ProvisionSnmpConf`	Enters the ProvisionSnmpConf mode.

Step 3	**setSnmpConfigsnmp_config** {**location***location_name* | **contact***contact_name* | **engine_id** *engine_id* }
	
	Example:
	`UCS(ProvisionSnmpConf)# setSnmpConfig  snmp_config engine_id 800007e5017f000001`
	`UCS(ProvisionSnmpConf)# setSnmpConfig  snmp_config contact user2`
		Configures an SNMP user.
		• **location**—Specifies SNMP location.
		• *location_name*—SNMP location name.
		• **contact**—Specifies SNMP contact.
		• *contact_name*—SNMP contact name.

	Command or Action	Purpose
	`UCS(ProvisionSnmpConf)# setSnmpConfig  snmp_config location Bangalore`	• **engine_id—**Specifies engine ID. • *engine_id*—Engine ID.
Step 4	**setSnmpConfig review** **Example:** `UCS(ProvisionSnmpConf)# setSnmpConfig review`	Displays the configuration.
Step 5	**setSnmpConfig commit** **Example:** `UCS(ProvisionSnmpConf)# setSnmpConfig commit`	Sends the configuration to NID.
Step 6	**getSnmpConfig getSnmpConfigRequest** **Example:** `UCS(ProvisionSnmpConf)# getSnmpConfig getSnmpConfigRequest`	Retrieves SNMP user configuration.
Step 7	**getSnmpConfig review** **Example:** `UCS(ProvisionSnmpConf)# getSnmpConfig review`	Displays the configuration.
Step 8	**getSnmpConfig commit** **Example:** `UCS(ProvisionSnmpConf)# getSnmpConfig commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(ProvisionSnmpConf)# exit`	Exits the ProvisionSnmpConf mode.

Configuration Example

The example shows how to create and retrieve an SNMP user:

```
UCS(ProvisionSnmpConf)# setSnmpUserConf  setSnmpUsersRequest  priv_passwd 12345678
UCS(ProvisionSnmpConf)# setSnmpConfig  snmp_config  engine_id 800007e5017f000001
UCS(ProvisionSnmpConf)# setSnmpConfig  snmp_config  contact user2
UCS(ProvisionSnmpConf)# setSnmpConfig  snmp_config  location Bangalore
UCS(ProvisionSnmpConf)# setSnmpConfig  review
UCS(ProvisionSnmpConf)# setSnmpConfig  commit

UCS(ProvisionSnmpConf)# getSnmpConfig  getSnmpConfigRequest
UCS(ProvisionSnmpConf)# getSnmpConfig review
UCS(ProvisionSnmpConf)# getSnmpConfig commit
UCS(ProvisionSnmpConf)# exit
```

The following is a sample output on the NID.

```
UCS(ProvisionSnmpConf)#$getSnmpConfigRequest
UCS(ProvisionSnmpConf)# getSnmpConfig  commit
```

```
GetSnmpConfig_Output.getSnmpConfigResponse.location = 'Bangalore'
GetSnmpConfig_Output.getSnmpConfigResponse.contact = 'user2'
GetSnmpConfig_Output.getSnmpConfigResponse.engine_id = '800007e5017f000001'
```

CHAPTER 14

Configuring PTP

This document describes the Precision Time Protocol (PTP) feature and configuration steps to implement PTP.

Prerequisites for Configuring PTP

- NID must be added to the controller.
- NID must be accessible from the controller.

Information About PTP

PTP, as defined in the IEEE 1588 standard, synchronizes with nanosecond accuracy the real-time clocks of the devices in a network. The clocks are organized into a master-member hierarchy. PTP identifies the switch port that is connected to a device with the most precise clock. This clock is referred to as the master clock. All the other devices on the network synchronize their clocks with the master and are referred to as members. Constantly exchanged timing messages ensure continued synchronization.

How to Provision PTP

Configuring Slave IPv4

To configure slave IPv4, unicast, one step on VLAN 7 with domain number 0, perform the following steps:

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**PTPPortType** **Example:** `UCS# PTPPortType`	Enters the PTPPortType mode.

Step 3	**setPTPclockInstance_v3 ptpClkConfig** {**clk_inst_domain** | **clk_slave** | **clock_enable** | **clock_instance** *clock_instance_number* |**dscp** | **mode** {**boundary** | **e2etransparent** | **master** | **p2ptransparent** | **slave**} | **two_step_flag** | **one_way** | **protocol** {**disable ethernet** | **unicast_ipv4** | **multicast_ipv4**} |**servo** | **slave-cfg** | **vlan** {**disable** | **vlan_id** *vlan_id_number*} | **pcp** {**disable** | **pcp_value**} | **enabled_ports** {**port1** | **port2** | **port3** | **port4** | **port5** | **port6**} | **filter** | **ho** | **identifier** | **localpriority** | **priority1** | **priority2** { **profile** | **disable** | **g8265dot1** | **g8275dot1** | **ieee1588**} | **clock_domain** *clock_domain_number*}
	
	Example:
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config clock_instance 1`
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config clock_domain 0`
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config clock_enable enable`
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config mode slave`
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config one_way disable`
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config protocol unicast_ipv4`
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config two_step_flag disable`
	`UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config vlan vlan_id 7`
		Configures slave IPV4, unicast, one step on VLAN 7 with domain number 0.
		
		• **clk_inst_domain—**HW based or SW based Clock domain.
		• **clk_slave—**Set PTP slave clock options.
		• **clock_enable—**Enables or disables clock.
		• **clock_instance—** Specifies PTP clock instance.
		• *clock_instance_number*— Clock instance number.
		• **dscp—** Dscp value.
		• **mode—**Specifies clock mode.
		• **boundary—**Specifies ordinary boundary clock.
		• **e2etransparent—** Specifies end to end transparent clock.
		• **master—**Specifies master only clock.
		• **p2ptransparent—**Specifies peer to peer transparent clock.
		• **slave—** Specifies slave only clock.
		• **two_step_flag—**Specifies two step flag.
		• **one_way—**Specifies one way.
		• **protocol—** Specifies the protocol.
		• **servo—** Set servo parameters.
		• **slave_cfg—** Specifies Unicast Slave configuration Entry.
		• **disable—**Disables protocol.
		• **ethernet—**Specifies EPS Ethernet protocol.
		• **unicast_ipv4—**Specifies unicast protocol.
		• **multicast_ipv4—** Specifies multicast protocol.
		• **vlan—**Specifies the VLAN ID.

	Command or Action	Purpose
		• **disable—**Disables VLAN tag. • **vlan_id—** Specifies the VLAN tag. • *vlan_id_number*— VLAN tag number. • **pcp—**Specifies VLAN PCP. • **disable—**Disables VLAN PCP. • **pcp_value—**Specifies the PCP value. • **enabled_ports—**Specifies UNI ports. • **port1—** Specifies physical port 1. • **port2—**Specifies physical port 2. • **port3—**Specifies physical port 3. • **port4—** Specifies physical port 4. • **port5—**Specifies physical port 5. • **port6—**Specifies physical port 6. • **filter—**Specifies filter parameters. • **ho—** Set PTP Servo holdover parameters. • **identifier—** Defines PTP clock instance identifier. • **localpriority—** Set Local priority for the port. • **priority1—**Specifies clock priority 1 for PTP BMC algorithm, 0 is highest priority. • **priority2—**Specifies clock priority 2 for PTP BMC algorithm. • **profile—** Specifies Clock's associated profile. • **clock_domain—**Specifies PTP domain. • *clock_domain_number*—PTP domain number.
Step 4	**setPTPclockInstance_v3 review** **Example:** `UCS(PTPPortType)# setPTPclockInstance_v3 review`	Displays the configuration.
Step 5	**setPTPclockInstance_v3 commit** **Example:** `UCS(PTPPortType)# setPTPclockInstance_v3 commit`	Sends the configuration to NID.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(PTPPortType)# exit`	Exits from the PTPPortType mode.

Configuration Example

The example shows how to configure slave IPv4, unicast, one step on VLAN 7 with domain number 0:

```
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config clock_instance 1
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config clock_domain 0
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config clock_enable enable
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config mode slave
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config one_way disable
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config protocol unicast_ipv4
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config two_step_flag disable
UCS(PTPPortType)# setPTPclockInstance_v3 ptp_clock_config vlan vlan_id 7

UCS(PTPPortType)# setPTPclockInstance_v3 review
UCS(PTPPortType)# setPTPclockInstance_v3 commit
UCS(PTPPortType)# exit
```

Enabling PTP on a Port

To enable PTP on port 1/4 with PTP instance 1, perform the following steps:

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**PTPPortType** **Example:** `UCS# PTPPortType`	Enters the PTPPortType mode.
Step 3	**setPTPportProperties ptp_port_conf** {**ptp_run** | **clock_inst** *clock_inst_number* | **port_id** *port_id_number* | **announce_interval** {**disable** | **value**} | **announce_timeout** {**disable** | **value**} | **delay_asymmetry** {**disable** | **value**} | **delay_mechanism** {**disable** | **e2e** | **p2p**} | **delay_req** {**disable** | **interval** *interval_value* } | **egress_latency** {**disable** | **value**} | **ingress_latency** {**disable** | **value**} | **internal** | **sync_interval** {**disable** | **value** *value*}}	Enables PTP on port 1/4, instance: 1. • **ptp_run—**Enables PTP on the specified port. • **clock_inst—** Specifies the PTP clock number. • *clock_inst_number*— PTP clock number. • **port_id—**Specifies the physical port number. • *port_id_number*—Physical port number.

	Command or Action	Purpose

	Example:
	`UCS(PTPPortType)# setPTPportProperties ptp_port_conf port_id 4`
	`UCS(PTPPortType)# setPTPportProperties ptp_port_conf clock_inst 1`
	`UCS(PTPPortType)# setPTPportProperties ptp_port_conf sync_interval value -4`
	`UCS(PTPPortType)# setPTPportProperties ptp_port_conf delay_req interval -6`
	`UCS(PTPPortType)# setPTPportProperties ptp_port_conf ptp_run enable`
		• **announce_interval—**Specifies the time interval for sending announce messages.
		• **disable—** Disables announce interval.
		• **value—**Specifies announce interval value.
		• **announce_timeout—**Sets announce timeout.
		• **disable—** Disables announce timeout.
		• **value—**Specifies announce timeout value.
		• **delay_asymmetry—**Sets path delay asymmetry.
		• **disable—** Disables delay asymmetry.
		• **value—**Specifies delay asymmetry in nano seconds.
		• **delay_mechanism—**Sets delay mechanism.
		• **disable—** Disables delay mechanism.
		• **e2e—**Specifies end to end delay mechanism.
		• **p2p—**Specifies peer to peer delay mechanism.
		• **delay_req—**Sets delay request interval.
		• **disable—** Disables delay request mechanism.
		• **interval—**Specifies peer to peer delay mechanism.
		• *interval_value*—Peer to peer delay value.
		• **egress_latency—**Sets port egress latency.
		• **disable—** Disables delay asymmetry.
		• **value—** Specifies egress latency in nano seconds.
		• **ingress_latency—**Sets port ingress latency.
		• **disable—** Disables delay asymmetry.
		• **value—** Specifies ingress latency in nano seconds.
		• **internal—** Enables as an internal interface.
		• **sync_interval—**Sets sync interval.
		• **disable—** Disables sync interval.
		• **value—** Specifies sync interval value.
		• *value*—Sync interval value.

	Command or Action	Purpose
Step 4	**setPTPportProperties review** **Example:** `UCS(PTPPortType)# setPTPportProperties review`	Displays the configuration.
Step 5	**setPTPportProperties commit** **Example:** `UCS(PTPPortType)# setPTPportProperties commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(PTPPortType)# exit`	Exits from the PTPPortType mode.

Configuration Example

The following example shows how to enable PTP on a port.

```
UCS(PTPPortType)# setPTPportProperties ptp_port_conf port_id 4
UCS(PTPPortType)# setPTPportProperties ptp_port_conf clock_inst 1
UCS(PTPPortType)# setPTPportProperties ptp_port_conf sync_interval value -4
UCS(PTPPortType)# setPTPportProperties ptp_port_conf delay_req interval -6
UCS(PTPPortType)# setPTPportProperties ptp_port_conf ptp_run enable

UCS(PTPPortType)# setPTPportProperties review
UCS(PTPPortType)# setPTPportProperties commit
UCS(PTPPortType)# exit
```

Note For G.8275.1 profile, there are a few additional parameters that have been added under **setPTPportProperties_v2**. In addition to the available parameters for **setPTPportProperties**, the following are the new parameters available:

- *localpriority* - Sets Local priority for the port.
- *mcast_dest* - Sets multicast destination address type for the port for G.8275.1 profile.
- *not_slave* - Sets not_slave attribute for the port for G8275.1 BMC algorithm.

Configuration Example

The following example shows how to enable PTP on a port using **setPTPportProperties_v2**.

```
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config ptp_run enable
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config clock_inst 1
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config port_id 2
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config localpriority value 12
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config not_slave enable
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config mcast_dest default
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config sync_interval value -4
UCS(PTPPortType)# setPTPportProperties_v2 ptp_port_config delay_req interval -6

UCS(PTPPortType)# setPTPportProperties_v2 review
```

```
Commands in queue:
        setPTPportProperties_v2 ptp_port_config ptp_run enable
        setPTPportProperties_v2 ptp_port_config clock_inst 1
        setPTPportProperties_v2 ptp_port_config port_id 2
        setPTPportProperties_v2 ptp_port_config localpriority value 12
        setPTPportProperties_v2 ptp_port_config not_slave enable
        setPTPportProperties_v2 ptp_port_config mcast_dest default
setPTPportProperties_v2 ptp_port_config sync_interval value -4
setPTPportProperties_v2 ptp_port_config delay_req interval -6

UCS(PTPPortType)# setPTPportProperties_v2 commit
UCS(PTPPortType)# exit
```

Enabling or Disabling Microsemi APR

To enable or disable Microsemi APR, perform the following steps:

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**PTPPortType** **Example:** `UCS# PTPPortType`	Enters the PTPPortType mode.

Step 3	**setPTPexternalProperties ptp_external_conf** {**clock_output_freq** | **one_pps_output** | **one_pps_input** | **vcxo_freq_control** | **algorithm** {**enable** | **one_hertz** | **min_phase**}}
	
	Example:
	`UCS(PTPPortType)# setPTPexternalProperties ptp_external_conf algorithm enable enable`
	`UCS(PTPPortType)# setPTPexternalProperties ptp_external_conf algorithm one_hertz enable`
		Enables Microsemi APR.
		• **clock_output_freq—**Specifies external clock output frequency in Hz.
		• **one_pps_output—** Enables 1PPS output.
		• **one_pps_input—** Enables 1PPS input.
		• **vcxo_freq_control—**Specifies APR settings.
		• **enable—**Enables or disables the APR.
		• **one_hertz—** Enables or disables 1 Hz.
		• **min_phase—**Specifies phase correction begin threshold.

Step 4	**setPTPexternalProperties ptp_external_conf** {**clock_output_freq** | **one_pps_output** | **one_pps_input** | **vcxo_freq_control** | **algorithm** {**enable** | **one_hertz** | **min_phase**}} **Example:** `UCS(PTPPortType)# setPTPexternalProperties ptp_external_conf algorithm enable disable`	Disables Microsemi APR and returns to default VTSS APR. • **clock_output_freq—**Specifies external clock output frequency in Hz. • **one_pps_output—** Enables 1PPS output. • **one_pps_input—** Enables 1PPS input. • **vcxo_freq_control—**Specifies APR settings.

	Command or Action	Purpose
		• **enable**—Enables or disables the APR. • **one_hertz**— Enables or disables 1 Hz. • **min_phase**—Specifies phase correction begin threshold.
Step 5	**setPTPexternalProperties review** **Example:** `UCS(PTPPortType)# setPTPexternalProperties review`	Displays the configuration.
Step 6	**setPTPclockInstance commit** **Example:** `UCS(PTPPortType)# setPTPexternalProperties commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(PTPPortType)# exit`	Exits from the PTPPortType mode.

Configuration Example

The following example shows how to enable Microsemi APR.

```
UCS(PTPPortType)# setPTPexternalProperties ptp_external_conf algorithm enable enable
UCS(PTPPortType)# setPTPexternalProperties ptp_external_conf algorithm one_hertz enable

UCS(PTPPortType)# setPTPexternalProperties review
UCS(PTPPortType)# setPTPexternalProperties commit
UCS(PTPPortType)# exit
```

The following example shows how to disable Microsemi APR and return to default VTSS APR.

```
UCS(PTPPortType)# setPTPexternalProperties ptp_external_conf algorithm enable disable

UCS(PTPPortType)# setPTPexternalProperties review
UCS(PTPPortType)# setPTPexternalProperties commit
UCS(PTPPortType)# exit
```

Note For G.8275.1 profile, there are a few additional parameters that have been added under **setPTPexternalProperties_v2**. In addition to the available parameters for **setPTPexternalProperties**, the following are the new parameters available:

- *ho_spec* - Holdover specification for G.8275 PTP clocks.
- *adjustment_method* - Adjustment method.

Configuration Example

The following example shows how to set holdover specification and adjustment method for G.8275.1 profile.

```
UCS(PTPPortType)# setPTPexternalProperties_v2 ptp_external_config ho_spec enable enable
UCS(PTPPortType)# setPTPexternalProperties_v2 ptp_external_config ho_spec cat1_value 11
UCS(PTPPortType)# setPTPexternalProperties_v2 ptp_external_config ho_spec cat2_value 12
UCS(PTPPortType)# setPTPexternalProperties_v2 ptp_external_config ho_spec cat3_value 13
UCS(PTPPortType)# setPTPexternalProperties_v2 ptp_external_config one_pps_mode one_pps_input
 enable
UCS(PTPPortType)# setPTPexternalProperties_v2 ptp_external_config adjustment_method ltc_phase
 enable

UCS(PTPPortType)# setPTPexternalProperties_v2 review

Commands in queue:
        setPTPexternalProperties_v2 ptp_external_config ho_spec enable enable
        setPTPexternalProperties_v2 ptp_external_config ho_spec cat1_value 11
        setPTPexternalProperties_v2 ptp_external_config ho_spec cat2_value 12
        setPTPexternalProperties_v2 ptp_external_config ho_spec cat3_value 13
        setPTPexternalProperties_v2 ptp_external_config one_pps_mode one_pps_input enable
        setPTPexternalProperties_v2 ptp_external_config adjustment_method ltc_phase enable

UCS(PTPPortType)# setPTPexternalProperties_v2 commit
UCS(PTPPortType)# exit
```

Verifying PTP

Use the following commands to verify the PTP status on the controller:

```
Switch(config-controller-PTPPortType)# showPTPall ptp_show_req 1
Switch(config-controller-PTPPortType)# showPTPall commit
```

This command displays the PTP configuration status on the NID.

Note To view the values of the parameters that have been added as part of G.8275.1 profile, use the **showPTPall_v2** command.

The following is a sample output from the **showPTPall** command:

```
ShowPTPall_Output.ptp_show_response[0].local_current_time.ptp_time =
'local time not implemented for clk_inst = 1 '
ShowPTPall_Output.ptp_show_response[0].local_current_time.clock_adjustment_method.t
 = 4
ShowPTPall_Output.ptp_show_response[0].local_current_time.clock_adjustment_method.u.software
 = ''
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.clock_id =
1
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.mode.t = 5
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.mode.u.slave
 = ''
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.two_step_flag
 = false
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.ports = '255'
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.clock_identity
 = '4348018d07000075'
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.clock_quality
 = 'Cl:255 Ac:128 Va:00004'
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.properties.priority1
 = 0
```

```
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.properties.priority2
 = 0
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.properties.clock_domain
 = 127
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.protocol.t
= 1
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.protocol.u.ethernet
 = ''
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.one_way =
true
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.vlan.t = 1
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.vlan.u.disable
 = ''
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.pcp.t = 1
ShowPTPall_Output.ptp_show_response[0].clock_default_dataset.pcp.u.disable
 = ''
ShowPTPall_Output.ptp_show_response[0].time_property.UTC_offset.t = 2
ShowPTPall_Output.ptp_show_response[0].time_property.UTC_offset.u.value
= 34
ShowPTPall_Output.ptp_show_response[0].time_property.frequency_traceable
 = true
ShowPTPall_Output.ptp_show_response[0].time_property.leap59 = true
ShowPTPall_Output.ptp_show_response[0].time_property.leap61 = true
ShowPTPall_Output.ptp_show_response[0].time_property.timing_traceable =
true
ShowPTPall_Output.ptp_show_response[0].time_property.ptp_timescale = true
ShowPTPall_Output.ptp_show_response[0].time_property.ptp_timesource = 0
ShowPTPall_Output.ptp_show_response[0].clock_parent_dataset =
'ParentPortIdentity port Pstat Var ChangeRate d0c282fffe17dfbf 1 0
65170-8207664

GrandmasterIdentity GrandmasterClockQuality Pri1 Pri2fe17dfbf68fe70cd
Cl:128 Ac:128 Va:33914 92 145
'
ShowPTPall_Output.ptp_show_response[0].clock_current_dataset.stpRm = 1
ShowPTPall_Output.ptp_show_response[0].clock_current_dataset.offset_from_master
 = ' 0.000.000.003'
ShowPTPall_Output.ptp_show_response[0].clock_current_dataset.mean_path_delay
 = ' 0.000.000.067'
ShowPTPall_Output.ptp_show_response[0].clock_current_dataset.slave_port
= 4
ShowPTPall_Output.ptp_show_response[0].clock_current_dataset.slave_state.t
 = 4
ShowPTPall_Output.ptp_show_response[0].clock_current_dataset.slave_state.u.LOCKED
 = ''
ShowPTPall_Output.ptp_show_response[0].clock_current_dataset.holdover =
'TRUE -368.3
'
ShowPTPall_Output.ptp_show_response[0].slave_cfg[0].index_ = 0
ShowPTPall_Output.ptp_show_response[0].slave_cfg[0].duration = 100
ShowPTPall_Output.ptp_show_response[0].slave_cfg[0].peer_ip_addr =
'7.7.7.7'
ShowPTPall_Output.ptp_show_response[0].slave_cfg[1].index_ = 1
ShowPTPall_Output.ptp_show_response[0].slave_cfg[1].duration = 100
ShowPTPall_Output.ptp_show_response[0].slave_cfg[1].peer_ip_addr =
'0.0.0.0'
ShowPTPall_Output.ptp_show_response[0].slave_cfg[2].index_ = 2
ShowPTPall_Output.ptp_show_response[0].slave_cfg[2].duration = 100
```

```
ShowPTPall_Output.ptp_show_response[0].slave_cfg[2].peer_ip_addr =
'0.0.0.0'
ShowPTPall_Output.ptp_show_response[0].slave_cfg[3].index_ = 3
ShowPTPall_Output.ptp_show_response[0].slave_cfg[3].duration = 100
ShowPTPall_Output.ptp_show_response[0].slave_cfg[3].peer_ip_addr =
'0.0.0.0'
ShowPTPall_Output.ptp_show_response[0].slave_cfg4.index_ = 4
ShowPTPall_Output.ptp_show_response[0].slave_cfg4.duration = 100
ShowPTPall_Output.ptp_show_response[0].slave_cfg4.peer_ip_addr = '0.0.0.0'

ShowPTPall Commit Success!!!
```

Additional References

Related Documents

Related Topic	Document Title
Cisco ME 3800x and ME 3600x Switches Software Configuration Guide, Cisco IOS Release 15.4(1)S	http://www.cisco.com/c/en/us/td/docs/switches/metro/me3600x_3800x/software/release/15-4_1_S/configuration/guide/3800x3600xscg.html

MIBs

MIB	MIBs Link
MIBs Supporting Cisco IOS	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

CHAPTER 15

Configuring ACLs

This chapter describes how to configure network security on the Cisco ME 1200 NID using access control lists (ACLs), which are also referred as access lists. Information in this chapter about ACLs is specific to IP Version 4 (IPv4). The other supported ACLs from Cisco IOS 15.4(2)SN release onwards are UDP, Ethernet, TCP, and ICMP.

For more information on ACLs, see Configuring Network Security with ACLs.

Prerequisites for Configuring ACLs

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.

Restrictions for Configuring ACLs

- The Policy 0 is applied on all interfaces, but is not displayed on any interface configuration. This means, you cannot view the policy 0 details when you use the **show running-config** command.
- Logging of the packet frames are not supported.
- The ACLs that are common to both IOS CLI and that of Cisco ME 1200 NID CLIs are only supported.
- Modifying ACL parameter that is applied on Cisco ME 1200 NID is not supported. To modify, remove the parameter using the **removeAclConfig** and **removeAclFromPort** commands, and apply it to a port afresh.
- A maximum of 512 ACL entries can be programmed.
- ACLs can be configured with policy IDs ranging from 0 to 63.

Tip To add more number of ACLs when the maximum entries are achieved, remove unwanted ACLs to configure more ACLs.

How to Configure ACLs

Configuring ACL Rules on the NID using the UCS Controller

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionACL** **Example:** `UCS# ProvisionACL`	Enters the ProvisionACL mode.
Step 3	**mac access-list** {**extended** *WORD*} **Example:** `UCS# mac access-list extended nid_acl_50000`	Configures the extended ACL rules. • **access-list—**Specifies the named access-list. ◦ **extended—**Specifies that the access-list is of the type **extended**. ◦ *WORD*—access-list name.

Step 4	**deny** {**H.H.H** |**any** |**host** *H.H.H*} {**H.H.H** |**any** |**host** *H.H.H*} OR **permit** {**H.H.H** | **any** |**host** *H.H.H* } {**H.H.H** |**any** |**host** *H.H.H* }
	
	Example:
	`UCS(ext-macl)# deny host 0000.0000.0001 host 5555.6666.7777`
	
	`UCS(ext-macl)# permit host 4444.aaaa.cccc host 4444.cccc.aaaa`
		Configures the ACL rule.
		• **deny—**Specifies the packets to be reject.
		◦ **icmp—**Internet Control Message Protocol.
		◦ **ip—**Any Internet Protocol.
		◦ **tcp—**Transmission Control Protocol.
		◦ **udp—**User Datagram Protocol.
		• **permit—**Specifies the packets to be forward.
		◦ **icmp—**Internet Control Message Protocol.
		◦ **ip—**Any Internet Protocol.
		◦ **tcp—**Transmission Control Protocol.

	Command or Action	Purpose
		◦ **udp**—User Datagram Protocol. • **H.H.H**—Specifies the 48-bit source or destination MAC address. • **any**—Specifies any source or destination MAC address. • **host**—Specifies a single source or destination host. ◦ *H.H.H*—48-bit source or destination MAC address.
Step 5	**exit** **Example:** `UCS(ext-macl)# exit`	Exits from the ext-macl mode.

Configuration Example

Note The following two configuration examples use IOS ACL commands.

```
Example 1: MAC
UCS# mac access-list extended nid_acl_50000
UCS(ext-macl)# deny host 0000.0000.0001 host 5555.6666.7777
UCS# mac access-list extended nid_acl_1002
UCS(ext-macl)# permit host 4444.aaaa.cccc host 4444.cccc.aaaa
UCS(ext-macl)# permit any any etype-6000
UCS(ext-macl)# exit

Example 2: IP

UCS# ip access-list extended nid_acl_1001
UCS(ext-nacl)# deny ip host 15.15.1.2  host 15.15.1.3
UCS(ext-nacl)# deny tcp host 10.10.1.5 eq 101 host 10.10.1.6 eq 100
UCS(ext-macl)# exit
```

Creating ACL Global Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionACL** **Example:** `UCS# ProvisionACL`	Enters the ProvisionACL mode.
Step 3	**createAclConfigcreateAclConfigRequest** {**acl_name** *WORD*} **Example:** `UCS(ProvisionACL)# createAclConfig createAclConfigRequest acl_name nid_acl_50000`	Configures the ACL global configuration. • **acl_name—**Specifies the extended ACL. ◦ *WORD*—ACL name.
Step 4	**createAclConfig review** **Example:** `UCS(ProvisionACL)# createAclConfig review`	Displays the configuration.
Step 5	**createAclConfig commit** **Example:** `UCS(ProvisionACL)# createAclConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionACL)# exit`	Exits from the ProvisionACL mode.

Configuration Example

```
UCS# ProvisionACL
UCS(ProvisionACL)# createAclConfig createAclConfigRequest acl_name acl_nid_50000
UCS(ProvisionACL)# createAclConfig review

Commands in queue:
   createAclConfig createAclConfigRequest acl_name acl_nid_50000

UCS(ProvisionACL)# createAclConfig commit

   CreateAclConfig Commit Success!!!

UCS(ProvisionACL)# exit
```

Applying ACL Configuration to the Ports

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionACL** **Example:** `UCS# ProvisionACL`	Enters the ProvisionACL mode.

Step 3	**applyAclToPortapplyAclToPortRequest** {**acl_name** *WORD* |**port_number***Port_Number*}
	
	Example:
	`UCS(ProvisionACL)# applyAclToPort applyAclToPortRequest acl_name nid_acl_50000`
	`UCS(ProvisionACL)# applyAclToPort applyAclToPortRequest port_number 3`
		Applies the ACL global configuration.
		• **acl_name—**specifies the extended ACL.
		  ◦ *WORD*—ACL name.
		• **port_number—**Specifies the port number.
		  ◦ *Port_Number*—port number. The range is from 1 to 6.

Step 4	**applyAclToPort review** **Example:** `UCS(ProvisionACL)# applyAclToPort review`	Displays the configuration.
Step 5	**applyAclToPort commit** **Example:** `UCS(ProvisionACL)# applyAclToPort commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionACL)# exit`	Exits from the ProvisionACL mode.

Configuration Example

```
UCS# ProvisionACL
UCS(ProvisionACL)# applyAclToPort applyAclToPortRequest acl_name nid_acl_50000
UCS(ProvisionACL)# applyAclToPort applyAclToPortRequest port_number 3
UCS(ProvisionACL)# applyAclToPort review

Commands in queue:
    applyAclToPort applyAclToPortRequest acl_name nid_acl_50000
    applyAclToPort applyAclToPortRequest port_number 3
```

```
UCS(ProvisionACL)# applyAclToPort commit

    ApplyAclToPort  Commit Success!!!

UCS(ProvisionACL)# exit
```

For applying the policy ID using EVC configuration, see Configuring ECE Rule 1 and Other Commands For EVC Configuration.

Viewing ACL Global Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionACL** **Example:** `UCS# ProvisionACL`	Enters the ProvisionACL mode.
Step 3	**getAclConfiggetAclConfigRequest** {**acl_name** *WORD*} **Example:** `UCS(ProvisionACL)# getAclConfig getAclConfigRequest acl_name nid_acl_50000`	Displays the ACL global configuration. • **acl_name**—Specifies the extended ACL . ◦ *WORD*—ACL name.
Step 4	**getAclConfig review** **Example:** `UCS(ProvisionACL)# getAclConfig review`	Displays the configuration.
Step 5	**getAclConfig commit** **Example:** `UCS(ProvisionACL)# getAclConfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionACL)# exit`	Exits from the ProvisionACL mode.

Configuration Example

```
Example 1: MAC

UCS# ProvisionACL
UCS(ProvisionACL)# getAclConfig getAclConfigRequest acl_name nid_acl_50000
UCS(ProvisionACL)# getAclConfig review

Commands in queue:
```

```
    getAclConfig getAclConfigRequest acl_name nid_acl_50000

UCS(ProvisionACL)# getAclConfig commit
    GetAclConfig_Output.getAclConfigResponse.acl_definition = '
   Extended MAC access list nid_acl_50000
      permit host 0000.0000.0001 host 5555.5556.5557
      permit any any etype-6000'

    GetAclConfig Commit Success!!!

UCS(ProvisionACL)# exit

Example 2: IP

UCS# ProvisionACL
UCS(ProvisionACL)# getAclConfig getAclConfigRequest acl_name nid_acl_50000
UCS(ProvisionACL)# getAclConfig review

Commands in queue:
     getAclConfig getAclConfigRequest acl_name nid_acl_50000

UCS(ProvisionACL)# getAclConfig commit
     GetAclConfig_Output.getAclConfigResponse.acl_definition = '
    Extended IP access list nid_acl_1001
       ip access-list extended nid_acl_1001
       deny ip host 15.15.1.2 host 15.15.1.3'

     GetAclConfig Commit Success!!!

UCS(ProvisionACL)# exit
```

Removing ACL Global Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionACL** **Example:** `UCS# ProvisionACL`	Enters the ProvisionACL mode.
Step 3	**removeAclconfigremoveAclConfigRequest** {**acl_name** *WORD*} **Example:** `UCS(ProvisionACL)# removeAclconfig removeAclConfigRequest acl_name nid_acl_50000`	Removes the ACL global configuration. • **acl_name—**Specifies the extended ACL . ◦ *WORD*—ACL name.
Step 4	**removeAclconfig review** **Example:** `UCS(ProvisionACL)# removeAclconfig review`	Displays the configuration.

	Command or Action	Purpose
Step 5	**removeAclconfig commit** **Example:** `UCS(ProvisionACL)# removeAclconfig commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionACL)# exit`	Exits the ProvisionACL mode.

Configuration Example

```
UCS# ProvisionACL
UCS(ProvisionACL)# removeAclconfig removeAclConfigRequest acl_name nid_acl_50000
UCS(ProvisionACL)# removeAclconfig review

Commands in queue:
   removeAclconfig removeAclConfigRequest acl_name nid_acl_50000

UCS(ProvisionACL)# removeAclconfig commit
   RemoveAclConfig Commit Success!!!


To verify the remove action:

UCS(ProvisionACL)# getAclConfig getAclConfigRequest acl_name nid_acl_50000
UCS(ProvisionACL)# getAclConfig commit

   GetAclConfig_Output.getAclConfigResponse.acl_definition = '
   nid_acl_50000 does not exist '

   GetAclConfig Commit Success!!!

UCS(ProvisionACL)# exit
```

What to Do Next

Use the **getAclConfig getAclConfigRequest** command to verify the remove action.

```
UCS(ProvisionACL)# getAclConfig getAclConfigRequest acl_name acl_name
UCS(ProvisionACL)# getAclConfig review
UCS(ProvisionACL)# getAclConfig commit
```

Removing ACL Port Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionACL** **Example:** `UCS# ProvisionACL`	Enters the ProvisionACL mode.
Step 3	**removeAclFromPort removeAclFromPortRequest** {**acl_name** *WORD* |**port_number***Port_Number*} **Example:** `UCS(ProvisionACL)# removeAclFromPort removeAclFromPortRequest port_number 3`	Removes the ACL port configuration. • **acl_name—**Specifies the extended ACL. ◦ *WORD*—ACL name. • **port_number—**Specifies the port number. ◦ *Port_Number*—Port number. The range is from 1to 6.
Step 4	**removeAclFromPort review** **Example:** `UCS(ProvisionACL)# removeAclFromPort review`	Displays the configuration.
Step 5	**removeAclFromPort commit** **Example:** `UCS(ProvisionACL)# removeAclFromPort commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionACL)# exit`	Exits from the ProvisionACL mode.

Configuration Example

```
UCS# ProvisionACL
UCS(ProvisionACL)# removeAclFromPort removeAclFromPortRequest port_number 3
UCS(ProvisionACL)# removeAclFromPort review

Commands in queue:
   removeAclFromPort removeAclFromPortRequest port_number 3

UCS(ProvisionACL)# removeAclFromPort commit
   RemoveAclFromPort Commit Success!!!

UCS(ProvisionACL)# exit
```

What to Do Next

Use the **showAclConfigSummary** command to view the interface ACL summary.

```
UCS(ProvisionACL)# showAclConfigSummary showAclConfigSummaryRequest
UCS(ProvisionACL)# showAclConfigSummary review
UCS(ProvisionACL)# showAclConfigSummary commit
```

Verifying ACL Configurations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionACL** **Example:** `UCS# ProvisionACL`	Enters the ProvisionACL mode.
Step 3	**showAclConfigSummaryshowAclConfigSummaryRequest** **Example:** `UCS(ProvisionACL)# showAclConfigSummary showAclConfigSummaryRequest`	Displays the ACL configuration summary.
Step 4	**showAclConfigSummary review** **Example:** `UCS(ProvisionACL)# showAclConfigSummary review`	Displays the configuration.
Step 5	**showAclConfigSummary commit** **Example:** `UCS(ProvisionACL)# showAclConfigSummary commit`	Sends the configuration to the NID.
Step 6	**exit** **Example:** `UCS(ProvisionACL)# exit`	Exits from the ProvisionACL mode.

Configuration Example

```
UCS# ProvisionACL
UCS(ProvisionACL)# showAclConfigSummary showAclConfigSummaryRequest
UCS(ProvisionACL)# showAclConfigSummary review

Commands in queue:
   showAclConfigSummary showAclConfigSummaryRequest

UCS(ProvisionACL)# showAclConfigSummary commit

   ShowAclConfigSummary_Output.showAclConfigSummaryResponse.showOutput = '
   Acl Configuration Summary
    policyid:  0 aclname: nid_acl_50000

   Interface Acl Summary
    Port 3: nid_acl_50000'

   ShowAclConfigSummary Commit Success!!!
```

```
UCS(ProvisionACL)# exit
```

CHAPTER 16

Configuring Quality of Service (QoS)

QoS includes traffic classification, marking, policing, queuing, and scheduling configured with service policies that are attached to ingress and egress targets. With QoS, you can provide preferential treatment to certain types of traffic at the expense of other types. When you do not configure QoS, the switch offers best-effort service to each packet, regardless of the packet contents or size.

Ingress QoS includes classification, marking, and policing. Classification can be based on the class of service (CoS), Differentiated Services Code Point (DSCP) in the inbound packet. You can classify based on Layer 2 MAC, IP-standard, or match based on AMAC, IP parameters using QCE configurations.

For EVC level QoS, see Creating a Policer.

Hierarchical QoS on the Cisco ME 1200 Series Carrier Ethernet Access Devices supports queuing and scheduling per EVC level per port. 8 queues are supported per port on the EVC. The EVC must be configured on the interface before configuring HQoS.

Note Single EVC per single UNI (input port) is supported.

The port scheduler is configured by default for fair round-robin scheduling between each EVC and non-service traffic, but can be weighted by configuring guaranteed bandwidths for the EVC. When guaranteed bandwidth is configured for an EVC, the remaining bandwidth of the port is divided equally between the remaining EVCs and non-service traffic.

Note Configuring of guaranteed bandwidth for non-service traffic is *not* supported.

The CIR bandwidth requirements are configured for each CoS per EVC. Excess traffic is strictly *not* prioritized, but is shared proportionally between the CoS within the EVC.

For more information, see Configuring Quality of Service (QoS).

How to Configure QoS

Provisioning the UCS Controller to Configure QoS

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionQos** **Example:** `UCS# ProvisionQos`	Enters QoS provisioning mode.

Step 3	**ProvisionQos** {**default** | **deleteQCE** | **exit** | **getInputGlobalPolicy** | **getInputGlobalPolicy** | **getOutputGlobalPolicy** | **getOutputPortPolicy** | **getQCE** | **getSystemQosSettings** | **no** | **reorderQCEentries** | **setInputGlobalPolicy** | **setInputPortPolicy** | **setOutputGlobalPolicy** | **setOutputPortPolicy** | **setQCE** | **setSystemQosSettings** | **showQCElist** | **showQueueStatistics**}
	
	Example:
	`UCS(ProvisionQos)# ?`
	`ProvisionQos sub-mode commands:`
	`  default                 Set a command to its defaults`
	`  deleteQCE               Delete a particular QCE`
	`  exit                    Exit from ProvisionQos sub configuration mode`
	`  getInputGlobalPolicy    Show Output QoS global features configured`
	`  getInputPortPolicy      Show Input Policy configured on Physical Port`
	`  getOutputGlobalPolicy   Show Global Output QoS features`
	`  getOutputPortPolicy     Show Output Policy configured on Physical Port`
	`  getQCE                  getQCE (default)`
	`  getSystemQosSettings    getSystemQosSettings (default)`
	`  no                      Negate a command or set its defaults`
	`  reorderQCEentries       reorderQCEentries (default)`
	`  setInputGlobalPolicy    configure Global Input QoS features`
	`  setInputPortPolicy      configure Input policy on Physical Port`
	`  setOutputGlobalPolicy   configure Global Output QoS features`
	`  setOutputPortPolicy     configure Output policy on Physical Port`
	`  setQCE                  setQCE (default)`
	`  setSystemQosSettings    set System-wide QoS settings`
	`  showQCElist             showQCElist (default)`
	`  showQueueStatistics     Display egress queue statistics`
		Displays the supported configurations for QoS.

Step 4	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS provisioning mode.

Configuration Example

The following example shows the supported QoS configuration:

```
UCS(ProvisionQos)# ?
ProvisionQos sub-mode commands:
  default                 Set a command to its defaults
  deleteQCE               Delete a particular QCE
  exit                    Exit from ProvisionQos sub configuration mode
  getInputGlobalPolicy    Show Output QoS global features configured
  getInputPortPolicy      Show Input Policy configured on Physical Port
  getOutputGlobalPolicy   Show Global Output QoS features
  getOutputPortPolicy     Show Output Policy configured on Physical Port
  getQCE                  getQCE (default)
  getSystemQosSettings    getSystemQosSettings (default)
  no                      Negate a command or set its defaults
  reorderQCEentries       reorderQCEentries (default)
  setInputGlobalPolicy    configure Global Input QoS features
  setInputPortPolicy      configure Input policy on Physical Port
  setOutputGlobalPolicy   configure Global Output QoS features
  setOutputPortPolicy     configure Output policy on Physical Port
  setQCE                  setQCE (default)
  setSystemQosSettings    set System-wide QoS settings
  showQCElist             showQCElist (default)
  showQueueStatistics     Display egress queue statistics
```

Configuring QoS Input Policy Features Globally on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setInputGlobalPolicy** {**commit** | **flush** | **inputGlobalPolicyConfig** | **review**}
	
	Example:
	
	`UCS(ProvisionQos)# setInputGlobalPolicy ?`
	`  commit                  commit setInputGlobalPolicy`
	`  flush                   flush all setInputGlobalPolicy commands from queue`
	
	`  inputGlobalPolicyConfig  configure Global Input QoS features`
	`  review                  review setInputGlobalPolicy commands`
		Sets the global input QoS policy.
		
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **inputGlobalPolicyConfig**—Sets the input policy global configuration on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**setInputGlobalPolicy inputGlobalPolicyConfig** {**match_DSCP value_xx** {**mark_DSCP** *dscp_value* | **mark_egress_class** *egress_queue*} | **service_policy** {**attach** | **detach**}}	Configures input global policy. • **match_DSCP**—Matches a particular DSCP value . • **value_xx**—Sets DSCP values on the controller. Use the following values:

	Command or Action	Purpose
	Example: `UCS(ProvisionQos)#` **`setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_00 mark_DSCP 4`** `UCS(ProvisionQos)#` **`setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_00 mark_egress_class 4`** `UCS(ProvisionQos)#` **`setInputGlobalPolicy inputGlobalPolicyConfig service_policy attach`**	• **value_00** DSCP 0. Default PHB for best effort traffic **value_01** DSCP 1 **value_02** DSCP 2 **value_03** DSCP 3 **value_04** DSCP 4 **value_05** DSCP 5 **value_06** DSCP 6 **value_07** DSCP 7 **value_08** CS1. Class Selector PHB precedence 1(DSCP 8) **value_09** DSCP 9 **value_10** AF11. Assured Forwarding PHB (DSCP 10) **value_11** DSCP 11 **value_12** AF12. Assured Forwarding PHB (DSCP 12) **value_13** DSCP 13 **value_14** AF13. Assured Forwarding PHB (DSCP 14) **value_15** DSCP 15 **value_16** CS2. Class Selector PHB precedence 1(DSCP 16) **value_16** CS2. Class Selector PHB precedence 1(DSCP 16) **value_17** DSCP 17 **value_18** AF21. Assured Forwarding PHB (DSCP 18) **value_19** DSCP 19 **value_20** AF22. Assured Forwarding PHB (DSCP 20) **value_21** DSCP 21 **value_22** AF23. Assured Forwarding PHB (DSCP 22) **value_23** DSCP 23 **value_24** CS3. Class Selector PHB precedence 1(DSCP 24) **value_25** DSCP 25 **value_26** AF31. Assured Forwarding PHB (DSCP 26) **value_27** DSCP 27 **value_28** AF32. Assured Forwarding PHB (DSCP 28) **value_29** DSCP 29 **value_30** AF33. Assured Forwarding PHB (DSCP 30) **value_31** DSCP 31 **value_32** CS4. Class Selector PHB precedence 1(DSCP 32)

	Command or Action	Purpose
		value_33DSCP 33 **value_34** AF41. Assured Forwarding PHB (DSCP 34) **value_35** DSCP 35 **value_36** AF42. Assured Forwarding PHB (DSCP 36) **value_37** DSCP 37 **value_38** AF43. Assured Forwarding PHB (DSCP 38) **value_39** DSCP 39 **value_40** CS5. Class Selector PHB precedence 1(DSCP 40) **value_41** DSCP 41 **value_42** DSCP 42 **value_43** DSCP 43 **value_44** VA. Voice Admit PHB(DSCP 44) **value_45** DSCP 45 **value_46** Expedited Forwarding PHB(DSCP 46) **value_47** DSCP 47 **value_48** CS6. Class Selector PHB precedence 1(DSCP 48) **value_49** DSCP 49 **value_50** DSCP 50 **value_51** DSCP 51 **value_52** DSCP 52 **value_53** DSCP 53 **value_54** DSCP 54 **value_55** DSCP 55 **value_56** CS7. Class Selector PHB precedence 1(DSCP 56) **value_57** DSCP 57 **value_58** DSCP 58 **value_59** DSCP 59 **value_60** DSCP 60 **value_61** DSCP 61 **value_62** DSCP 62 **value_63** DSCP 63 • **mark_DSCP—**Marks the DSCP on the controller. The valid range is from 0 to 63. 64 is invalid.

	Command or Action	Purpose
		• **mark_egress_class—**Assigns to egress queue. The valid range is from 0 to 7. 8 is invalid. • **service_policy—**Attaches or detaches the service policy. ◦ **attach—**Attaches the service policy and enables the configuration. ◦ **detach—**Removes the service policy, removes all configuration and restore the default configuration.

Step 3	**setInputGlobalPolicy review**
	
	Example:
	`UCS(ProvisionQos)# setInputGlobalPolicy review`
	
	`Commands in queue:`
	`setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_02 mark_DSCP 4`
	`setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_02 mark_egress_class 4`
	`setInputGlobalPolicy inputGlobalPolicyConfig service_policy attach`
		Displays the QoS configuration on the controller.

Step 4	**setInputGlobalPolicy commit** **Example:** `UCS(ProvisionQos)# setInputGlobalPolicy commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS provisioning mode.

Configuration Example

The example shows how to configure QoS input policy globally on the UCS controller:

```
UCS(ProvisionQos)# setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_00 mark_DSCP
 4
UCS(ProvisionQos)# setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_00
mark_egress_class 4
UCS(ProvisionQos)# setInputGlobalPolicy inputGlobalPolicyConfig service_policy attach
UCS(ProvisionQos)# setInputGlobalPolicy review
Commands in queue:
        setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_02 mark_DSCP 4
        setInputGlobalPolicy inputGlobalPolicyConfig match_DSCP value_02 mark_egress_class
 4
        setInputGlobalPolicy inputGlobalPolicyConfig service_policy attach

UCS(ProvisionQos)# setInputGlobalPolicy commit
```

```
SetInputGlobalPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Configuring QoS Input Policy Features at Port level on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setInputPortPolicy** {**commit** | **flush** | **inputPortPolicyConfig** | **review**}
	
	Example:
	`UCS(ProvisionQos)# setInputPortPolicy ?`
	`commit                  commit setInputPortPolicy`
	`flush                   flush all setInputPortPolicy commands from queue`
	`inputPortPolicyConfig   configure Input policy on Physical Port`
	`review                  review setInputPortPolicy commands`
		Sets the input QoS policy at port level.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **inputPortPolicyConfig**—Sets the input policy configuration at port level on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**inputPortPolicyConfig** {**egress_class_marking** {**enable** | **disable**} | **globalDscpBasedDscpIngressMarking** {**enable** | **disable**} | **globalDscpBasedEgressClassMarking** {**enable** | **disable**} | **match cos value_xx mark_egress_class** *egress_queue* | **port_number***port_number* | **port_policer** {**cir** *kbps* | **state** {**enable** | **disable**}} | **qce** {**address** {**destination** | **source**} | **key** {**double_tag** | **ip_address** | **mac_ip_addr** | **normal**}} | **service_policy** {**attach** | **detach**}}
	
	Example:
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig egress_class_marking enable`
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig globalDscpBasedDscpIngressMarking enable`
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig globalDscpBasedEgressClassMarking enable`
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig match cos value_0`
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig port_number 1`
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig port_policer cir 1000`
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig port_policer state enable`
	`UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig qce address destination`
	`UCS(ProvisionQos)# setInputPortPolicy`
		Configures input policy at port.
		• **egress_class_marking**—Enables egress class marking as configured per each cos matched packet.
		• **globalDscpBasedDscpIngressMarking**—Enables DSCP based DSCP ingress marking on the port as per the configured global policy.
		• **globalDscpBasedEgressClassMarking**—Enable DSCP based Egress Class marking on the port as per the configured global policy.
		• **enable**—Enables the marking.
		• **disable**—Disables the marking.
		• **match**—Matches input packet COS.
		• **cos**—Sets the COS value.
		◦ **value_0**—Sets the COS value 0.
		◦ **value_1**—Sets the COS value 1.
		◦ **value_2**—Sets the COS value 2.
		◦ **value_3**—Sets the COS value 3.
		◦ **value_4**—Sets the COS value 4.

	Command or Action	Purpose
	inputPortPolicyConfig qce key double_tag UCS(ProvisionQos)# **setInputPortPolicy inputPortPolicyConfig service_policy attach**	◦ **value_5**—Sets the COS value 5. ◦ **value_6**—Sets the COS value 6. ◦ **value_7**—Sets the COS value 7. • **mark_egress_class** *egress_queue*—Sets the egress queue value. The valid range is from 0 to 7. Queue 8 in invalid. • **port_number** *port_number*—Sets the port number. The valid range is from 1 to 6. Port 7 is invalid. • **port_policer**—Sets port level policer. • **cir** *kbps*—Sets committed information rate in kbps. The valid range is from 100 to 1000000. • **state**—Enables or disables the policer state. • **qce**—Sets TCAM based QoS control entry settings for the port. • **address**—Matches the source or destination address of incoming packet in QCE. • **destination**—Matches against destination address. • **source**—Matches against source address. • **key**—Matches the key template in QCE. The default is normal. ◦ **double_tag**—Matches against Match outer tag, inner tag, IP protocol, DSCP and DPORT. ◦ **ip_address**—Matches against Match outer tag, SMAC/DMAC, IP protocol, DSCP, SIP and DIP. ◦ **mac_ip_addr**—Match outer tag, inner tag, SMAC, DMAC, IP protocol, DSCP, SIP,DIP, SPORT and DPORT. ◦ **normal**—Match outer tag, SMAC/DMAC, IP protocol, DSCP, SIP/DIP, SPORT and DPORT (default). • **service_policy**—Attaches or detaches the service policy on the port. • **attach**—Attaches the service policy and enables the configuration. • **detach**—Removes the service policy, removes the configuration and restores the default configuration.

	Command or Action	Purpose

Step 3	**setInputPortPolicy review**
	
	Example:
	`UCS(ProvisionQos)# setInputPortPolicy review`
	
	`Commands in queue:`
	`Commands in queue:`
	`setInputPortPolicy inputPortPolicyConfig egress_class_marking enable`
	`setInputPortPolicy inputPortPolicyConfig globalDscpBasedDscpIngressMarking enable`
	`setInputPortPolicy inputPortPolicyConfig globalDscpBasedEgressClassMarking enable`
	`setInputPortPolicy inputPortPolicyConfig match cos value_0 mark_egress_class 5`
	`setInputPortPolicy inputPortPolicyConfig match cos value_0 mark_egress_class 5`
	`setInputPortPolicy inputPortPolicyConfig port_policer state enable`
	`setInputPortPolicy inputPortPolicyConfig port_number 1`
	`setInputPortPolicy inputPortPolicyConfig port_policer cir 1000`
	`setInputPortPolicy inputPortPolicyConfig port_policer state enable`
	`setInputPortPolicy inputPortPolicyConfig qce address source`
	`setInputPortPolicy inputPortPolicyConfig qce key normal`
	`setInputPortPolicy inputPortPolicyConfig service_policy attach`
		Displays the QoS configuration on the controller.

Step 4	**setInputPortPolicycommit** **Example:** `UCS(ProvisionQos)# setInputPortPolicy commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS provisioning mode.

Configuration Example

The example shows how to configure QoS input port policy on the UCS controller:

```
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig egress_class_marking enable
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig globalDscpBasedDscpIngressMarking
 enable
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig globalDscpBasedEgressClassMarking
 enable
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig match cos value_0
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig port_number 1
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig port_policer cir 1000
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig port_policer state enable
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig qce address destination
UCS(ProvisionQos)# setInputPortPolicy inputPortPolicyConfig qce key double_tag
UCS(ProvisionQos))# setInputPortPolicy inputPortPolicyConfig service_policy attach
UCS(ProvisionQos)# setInputPortPolicy review
```

```
Commands in queue:
        Commands in queue:
        setInputPortPolicy inputPortPolicyConfig egress_class_marking enable
        setInputPortPolicy inputPortPolicyConfig globalDscpBasedDscpIngressMarking enable
        setInputPortPolicy inputPortPolicyConfig globalDscpBasedEgressClassMarking enable
        setInputPortPolicy inputPortPolicyConfig match cos value_0 mark_egress_class 5
        setInputPortPolicy inputPortPolicyConfig match cos value_0 mark_egress_class 5
        setInputPortPolicy inputPortPolicyConfig port_policer state enable
        setInputPortPolicy inputPortPolicyConfig port_number 1
        setInputPortPolicy inputPortPolicyConfig port_policer cir 1000
        setInputPortPolicy inputPortPolicyConfig port_policer state enable
        setInputPortPolicy inputPortPolicyConfig qce address source
        setInputPortPolicy inputPortPolicyConfig qce key normal
        setInputPortPolicy inputPortPolicyConfig service_policy attach
UCS(ProvisionQos)# setInputPortPolicy commit
SetInputPortPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Configuring QoS Output Policy Features Globally on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setOutputGlobalPolicy** {**commit** | **flush** | **outputGlobalPolicyConfig** | **review**}
	
	Example:
	
	`UCS(ProvisionQos)#` **`setOutputGlobalPolicy ?`**
	`commit commit setOutputGlobalPolicy`
	`flush flush all setOutputGlobalPolicy commands from queue`
	`outputGlobalPolicyConfig configure Global Output QoS features`
	`review review setOutputGlobalPolicy commands`
		Sets the global output QoS policy.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **outputGlobalPolicyConfig**—Sets the output policy global configuration on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**setOutputGlobalPolicy outputGlobalPolicyConfig** {**match_DSCP value_xx** {**mark_DSCP** *dscp_value*} | **service_policy** {**attach** | **detach**}}
	
	Example:
	
	`UCS(ProvisionQos)#` **`setOutputGlobalPolicy outputGlobalPolicyConfig match_DSCP value_01 mark_DSCP 1`**
	`UCS(ProvisionQos)#` **`setOutputGlobalPolicy`**
		Configures output global policy.
		• **match_DSCP**—Matches a particular DSCP value .
		• **value_xx**—Sets DSCP values on the controller. Use the following values:
		  • **value_00** DSCP 0. Default PHB for best effort traffic
		  value_01 DSCP 1
		  value_02 DSCP 2
		  value_03 DSCP 3
		  value_04 DSCP 4

	Command or Action	Purpose
	`outputGlobalPolicyConfig service_policy attach`	**value_05** DSCP 5 **value_06** DSCP 6 **value_07** DSCP 7 **value_08** CS1. Class Selector PHB precedence 1(DSCP 8) **value_09** DSCP 9 **value_10** AF11. Assured Forwarding PHB (DSCP 10) **value_11** DSCP 11 **value_12** AF12. Assured Forwarding PHB (DSCP 12) **value_13** DSCP 13 **value_14** AF13. Assured Forwarding PHB (DSCP 14) **value_15** DSCP 15 **value_16** CS2. Class Selector PHB precedence 1(DSCP 16) **value_16** CS2. Class Selector PHB precedence 1(DSCP 16) **value_17** DSCP 17 **value_18** AF21. Assured Forwarding PHB (DSCP 18) **value_19** DSCP 19 **value_20** AF22. Assured Forwarding PHB (DSCP 20) **value_21** DSCP 21 **value_22** AF23. Assured Forwarding PHB (DSCP 22) **value_23** DSCP 23 **value_24** CS3. Class Selector PHB precedence 1(DSCP 24) **value_25** DSCP 25 **value_26** AF31. Assured Forwarding PHB (DSCP 26) **value_27** DSCP 27 **value_28** AF32. Assured Forwarding PHB (DSCP 28) **value_29** DSCP 29 **value_30** AF33. Assured Forwarding PHB (DSCP 30) **value_31** DSCP 31 **value_32** CS4. Class Selector PHB precedence 1(DSCP 32) **value_33**DSCP 33 **value_34** AF41. Assured Forwarding PHB (DSCP 34) **value_35** DSCP 35 **value_36** AF42. Assured Forwarding PHB (DSCP 36) **value_37** DSCP 37

	Command or Action	Purpose
		value_38 AF43. Assured Forwarding PHB (DSCP 38) **value_39** DSCP 39 **value_40** CS5. Class Selector PHB precedence 1(DSCP 40) **value_41** DSCP 41 **value_42** DSCP 42 **value_43** DSCP 43 **value_44** VA. Voice Admit PHB(DSCP 44) **value_45** DSCP 45 **value_46** Expedited Forwarding PHB(DSCP 46) **value_47** DSCP 47 **value_48** CS6. Class Selector PHB precedence 1(DSCP 48) **value_49** DSCP 49 **value_50** DSCP 50 **value_51** DSCP 51 **value_52** DSCP 52 **value_53** DSCP 53 **value_54** DSCP 54 **value_55** DSCP 55 **value_56** CS7. Class Selector PHB precedence 1(DSCP 56) **value_57** DSCP 57 **value_58** DSCP 58 **value_59** DSCP 59 **value_60** DSCP 60 **value_61** DSCP 61 **value_62** DSCP 62 **value_63** DSCP 63 • **mark_DSCP—**Marks the DSCP on the controller. The valid range is from 0 to 63. 64 is invalid. • **service_policy—**Applies the service policy. • **attach—**Adds the service policy. • **detach—**Removes the service policy.

	Command or Action	Purpose

Step 3	**setOutputGlobalPolicy review**
	
	Example:
	`UCS(ProvisionQos)#`
	`setOutputGlobalPolicy review`
	
	`Commands in queue:`
	`        setOutputGlobalPolicy`
	`outputGlobalPolicyConfig match_DSCP`
	`value_01 mark_DSCP 1`
	`        setOutputGlobalPolicy`
	`outputGlobalPolicyConfig service_policy`
	` attach`
		Displays the QoS configuration on the controller.

Step 4	**setOutputGlobalPolicycommit** **Example:** `UCS(ProvisionQos)#` **`setOutputGlobalPolicy commit`**	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)#` **`exit`**	Exits the QoS provisioning mode.

Configuration Example

The example shows how to configure QoS output policy globally on the UCS controller:

```
UCS(ProvisionQos)# setOutputGlobalPolicy outputGlobalPolicyConfig match_DSCP value_00
mark_DSCP 4
UCS(ProvisionQos)# setOutputGlobalPolicy outputGlobalPolicyConfig service_policy attach
UCS(ProvisionQos)# setOutputGlobalPolicy review
Commands in queue:
        setOutputGlobalPolicy outputGlobalPolicyConfig match_DSCP value_01 mark_DSCP 1
        setOutputGlobalPolicy outputGlobalPolicyConfig service_policy attach

UCS(ProvisionQos)# setOutputGlobalPolicy commit
SetOutputGlobalPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Configuring QoS Output Policy Features at Port level on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setOutputPortPolicy** {**commit** | **flush** | **outputPortPolicyConfig** | **review**}
	
	Example:
	`UCS(ProvisionQos)# setOutputPortPolicy ?`
	`  commit                 commit setOutputPortPolicy`
	`  flush                 flush all setOutputPortPolicy`
	` commands from queue`
	`  outputPortPolicyConfig configure Output policy on`
	`Physical Port`
	`  review                 review setOutputPortPolicy`
	`commands`
		Sets the output QoS policy at port level.
		• **commit—**Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **outputPortPolicyConfig—**Sets the output policy configuration at port level on the controller.
		• **review—**Displays the configuration on the controller.

Step 2	**outputPortPolicyConfig** {**cos_marking** {**enable** | **disable**} | **globalDscpBasedDscpEgressMarking** {**enable** | **disable**} | **matchegress_class_xx** {**bandwidth** {**priority_level** *pr_level* | **remaining_ratio** *ratio*} | **mark_cos** *mark_cos_vlaue* | **shaper** {**rate** *rate_value* | **state** {**enable** | **disable**}}} | **port_number***port_number* | **port_shape** {**rate** *rate_value* | **state** {**enable** | **disable**}} |**service_policy** {**attach** | **detach**}}
	
	Example:
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig cos_marking enable`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig`
	`globalDscpBasedDscpEgressMarking enable`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig match egress_class_0 bandwidth`
	` priority_level 8`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig match egress_class_0 bandwidth`
	` remaining_ratio 20`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig match egress_class_0 mark_cos`
	` 7`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig port_number 1`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig port_shaper rate 1000`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig port_shaper state enable`
	`UCS(ProvisionQos)# setOutputPortPolicy`
	`outputPortPolicyConfig service_policy attach`
		Configures output policy at port.
		• **cos_marking—**Enables egress class marking as configured per each cos matched packet.
		• **globalDscpBasedDscpEgressMarking—**Enables DSCP based DSCP egress marking on the port as per the configured global policy.
		• **enable**—Enables the marking.
		• **disable—**Disables the marking.
		• **match—**Matches output packet COS.
		  ◦ **all-egress_classes—**Sets the egress for all queues from 0 to 7.
		  ◦ **egress_class_0—**Sets queue 0, lowest priority.
		  ◦ **egress_class_1**—Sets queue 1.
		  ◦ **egress_class_2**—Sets queue 2.
		  ◦ **egress_class_3**—Sets queue 3.
		  ◦ **egress_class_4**—Sets queue 4.
		  ◦ **egress_class_5**—Sets queue 5, higher priority.
		  ◦ **egress_class_6**—Sets queue 6, highest priority.
		• **bandwith** —Sets scheduling scheme.
		• **priority_level** *pr_level***—**Configures priority scheduling. The valid range is from 1 to 8.
		• **remaining_ratio** *ratio***—**Configures weighted round robin mode of scheduling. The vallid range is from 1 to 100.
		• **shaper** **—**Configures queue level shaper.

	Command or Action	Purpose
		• **mark_cos** *cos_vallue*—Sets the COS value for marking The valid range is from 0 to 7. COS 8 in invalid. • **port_number** *port_number*—Sets the port number. The valid range is from 1 to 6. Port 7 is invalid. • **port_shaper**—Sets port level shaper. • **rate** *kbps*—Sets shaper rate in kbps. The valid range is from 100 to 1000000. • **state**—Enables or disables the port shaper state. • **service_policy**—Sets the service policy on the port. • **attach**—Adds the service policy. • **detach**—Removes the service policy.

Step 3	**setOutputPortPolicy review**
	
	Example:
	`UCS(ProvisionQos)# setOutputPortPolicy review`
	
	`Commands in queue:`
	`setOutputPortPolicy outputPortPolicyConfig cos_marking enable`
	`setOutputPortPolicy outputPortPolicyConfig globalDscpBasedDscpEgressMarking enable`
	`setOutputPortPolicy outputPortPolicyConfig match egress_class_0 bandwidth priority_level 8`
	`setOutputPortPolicy outputPortPolicyConfig match all-egress_classes mark _cos 7`
	`setOutputPortPolicy outputPortPolicyConfig port_number 4`
	`setOutputPortPolicy outputPortPolicyConfig port_shaper rate 100`
	`setOutputPortPolicy outputPortPolicyConfig port_shaper state enable`
	`setOutputPortPolicy outputPortPolicyConfig service_policy attach`
		Displays the QoS configuration on the controller.

Step 4	**setOutputPortPolicycommit** **Example:** `UCS(ProvisionQos)# setOutputPortPolicy commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS provisioning mode.

Configuration Example

The example shows how to configure QoS output port policy on the controller:

```
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig cos_marking enable
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig globalDscpBasedDscpEgressMarking
 enable
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig match egress_class_0 bandwidth
 priority_level 8
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig match egress_class_0 bandwidth
 remaining_ratio 20
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig match egress_class_0 mark_cos
 7
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig port_number 1
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig port_shaper rate 1000
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig port_shaper state enable
UCS(ProvisionQos)# setOutputPortPolicy outputPortPolicyConfig service_policy attach
UCS(ProvisionQos)# setOutputPortPolicy review
Commands in queue:
        setOutputPortPolicy outputPortPolicyConfig cos_marking enable
        setOutputPortPolicy outputPortPolicyConfig globalDscpBasedDscpEgressMarking enable
        setOutputPortPolicy outputPortPolicyConfig match egress_class_0 bandwidth
priority_level 8
        setOutputPortPolicy outputPortPolicyConfig match all-egress_classes mark _cos 7
        setOutputPortPolicy outputPortPolicyConfig port_number 4
        setOutputPortPolicy outputPortPolicyConfig port_shaper rate 100
        setOutputPortPolicy outputPortPolicyConfig port_shaper state enable
        setOutputPortPolicy outputPortPolicyConfig service_policy attach
UCS(ProvisionQos)# setOutputPortPolicy commit
SetInputPortPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Configuring QoS Control Entry (QCE) on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setQCE** {**commit** | **flush** | **QCE_configuration** | **review**}
	
	Example:
	`UCS(ProvisionQos)# setQCE ?`
	`QCE_configuration  setQCE (default)`
	`commit  commit setQCE`
	`flush  flush all setQCE commands from queue`
	`review  review setQCE commands`
		Sets QCE configuration.
		• **commit—**Sends the QoS configuration to NID.
		• **flush—**Flushes all QoS configuration from the queue.
		• **QCE_configuration—**Sets the default QCE on the controller.
		• **review—**Displays the configuration on the controller.

Step 2	**setQCE QCE_configuration** {**control** {**actions** | **ingress_match**} | **qce_id**}	Configures QCE. • **control—**Configures QCE.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`
	`UCS(ProvisionQos)# setQCE QCEconfiguration qce-id 4`
		• **actions—**Configures QCE actions.
		• **ingress_match—**Configures ingress match.
		• **qce-id—**Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoSProvision mode.

Configuration Example

The example shows how to configure QoS QCE on the UCS controller:

```
UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4
UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos
 val_0-1
UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan
c_tagged
UCS(ProvisionQos)# setQCE review

Commands in queue:
        setQCE QCE_configuration control actions mark_COS 4
        setQCE QCE_configuration control actions mark_DSCP 3
        setQCE QCE_configuration control actions mark_egress_class 4
        setQCE QCE_configuration control ingress_match frame_type any
       setQCE QCE_configuration control ingress_match inner_tag_match match_fields inner_cos
 val_0-1

UCS(ProvisionQos)# setQCE commit
 SetQCE Commit Success!!!
UCS(ProvisionQos)# exit
```

Configuring QoS Control Entry (QCE) Control Actions on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setQCEQCE_configuration**{**control** {**actions** |**ingress_match**} | **qce-id**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`
	`UCS(ProvisionQos)# setQCE QCEconfiguration qce-id 4`
		Configures QCE.
		• **control—**Configures QCE.
		• **actions—**Configures QCE actions.
		• **ingress_match—**Configures ingress match.
		• **qce-id—**Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

Step 2	**setQCE QCE_configurationcontrol** {**actions** {**mark_COS** *cos_vlaue* | **mark_DSCP** *dscp_vlaue* | **mark_egress_class** *egress_queue*}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_DSCP 3`
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_egress_class 4`
		• **control—**Configures QCE.
		• **actions—**Configures QCE actions.
		• **mark_COS** *cos_vlaue***—**Marks the Cos packets. The valid range is from 0 to 7. Value 8 is invalid.
		• **mark_DSCP** *dscp_vlaue***—**Marks the DSCP packets. The valid range is from 0 to 63. Value 64 is invalid.
		• **mark_egress_class** *egress_queue***—**Marks the egress queue. The valid range is from 0 to 7. Value 8 is invalid.

Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the ucs controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS mode.

Configuration Example

The example shows how to configure QoS input policy globally on the ucs controller:

```
UCS(ProvisionQos)# setQCE QCEConfig match_DSCP value_00 mark_DSCP 4
UCS(ProvisionQos)# setQCE QCEConfig match_DSCP value_00 mark_egress_class 4
UCS(ProvisionQos)# setQCE QCEConfig service_policy attach
UCS(ProvisionQos)# setQCE review
```

```
Commands in queue:
        setQCE QCE_configuration control actions mark_COS 4
        setQCE QCE_configuration control actions mark_DSCP 3
        setQCE QCE_configuration control actions mark_egress_class 4

UCS(ProvisionQos)# setQCE commit
UCS(ProvisionQos)# exit
```

Configuring QCE Match Ingress Parameters on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setQCEQCE_configuration**{**control** {**actions** |**ingress_match**} | **qce-id**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`
	`UCS(ProvisionQos)# setQCE QCEconfiguration qce-id 4`
		Configures QCE.
		• **control**—Configures QCE.
		• **actions**—Configures QCE actions.
		• **ingress_match**—Configures ingress match.
		• **qce-id**—Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

Step 2	**setQCE QCE_configuration control ingress_match** {**frame_type** | **inner_tag_match** | **mac_params outer_tag_match** | **ports**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match frame_type any match_fields inner_cos val_0-1 any`
	`UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match inner_tag_match match_fields inner_cos val_0-3`
	`UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match mac_params dmac_filter any`
	`UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match outer_tag_match match_fields cos val_2-3`
	`UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match ports gigabitEthernet_2 enable`
		• **ingress_match**—Configures ingress match.
		◦ **frame_type**—Matches against frame payload.
		◦ **inner_tag_match**—Matches against inner tag.
		◦ **mac_params**—Matches against MAC filters.
		◦ **outer_tag_match**—Matches against outer tag.
		◦ **ports**—Matches against ports.

Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the ucs controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.

	Command or Action	Purpose
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS mode.

Configuration Example

The example shows how to configure QCE control ingress match parameters on the ucs controller:

```
UCS(ProvisionQos)# setQCE QCE-configuration control ingress_match frame_type any match_fields
 inner_cos val_0-1 any
UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match inner_tag_match match_fields
 inner_cos val_0-3
UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match mac_params dmac_filter
any
UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match outer_tag_match match_fields
 cos val_2-3
UCS(ProvisionQos)# setQCE QCE_configuration control ingress_match ports gigabitEthernet_2
enable
UCS(ProvisionQos)# setQCE review
Commands in queue:
       setQCE QCE_configuration control ingress_match inner_tag_match match_fields inner_cos
 val_0-3
        setQCE QCE_configuration control ingress_match mac_params dmac_filter any
        setQCE QCE_configuration control ingress_match outer_tag_match match_fields cos
val_2-3
        setQCE QCE_configuration control ingress_match ports GigabitEthernet_2 enable

UCS(ProvisionQos)# setQCE commit
UCS(ProvisionQos)# exit
```

Configuring QCE Control Ingress Match Frame Type Parameter on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setQCEQCE_configuration**{**control** {**actions** |**ingress_match**} | **qce-id**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`
	`UCS(ProvisionQos)# setQCE QCEconfiguration qce-id 4`
		Configures QCE.
		• **control—**Configures QCE.
		• **actions—**Configures QCE actions.
		• **ingress_match—**Configures ingress match.
		• **qce-id—**Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

	Command or Action	Purpose

Step 2	**setQCE QCE_configuration control ingress_match frame_type** {**any** | **ipv4** {**dest_ip_filter** {**any** | **host** *host_name* | **network** {**dest_ip_addr** *dest_add* | **dest_ip_mask** *dest_mask*}} | **dscp_filter** {**any** | **range** *range_value* | **specific** *dscp_filter*} | **fragment_type** {**any** | **frag** | **non_frag**} | **protocol** {**any** | **specific** *protocol_value* | **tcp** | **udp**} | **source_ip_filter** {**any** | **host** *host_name* | **network** {**source_ip_addr source_ip_addr** *source_add* | **source_ip_mask** *source_mask*}} | **ipv6** {**dest_ip_filter** {**any** | **specific** {**dest_ip_addr_32bits** *dest_add* | **dest_ip_mask_32bits** *dest_mask*}} | **dscp_filter** {**any** | **range** *range_value* | **specific** *dscp_filter*} | **protocol** {**any** | **specific** *protocol_value* | **tcp** | **udp**} | **source_ip-filter** {**any** | **specific source_ip_addr_32bits** *source_add* | **source_ip_mask_32bits** *source_mask*}}
	
	Example:
	
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dest_ip_filter any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dest_ip_filter host host1`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dest_ip_filter network dest_ip_addr addr2`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dscp_filter host any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dscp_filter range 3-4`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 fragment_type frag`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 protocol specific 45`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 source_ip_filter network source_ip_mask soumask`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 dest_ip_filter any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 dest_ip_filter specfic dest_ip_addr_32its dest34`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 dscp_filter specific 45 any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 protocol specific 450`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 protocol specific 45`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 source_ip_filter specific source_ip_mask source-mask`
		• **control**—Configures QCE .
		• **ingress_match**—Configures ingress match.
		◦ **frame_type**—Matches against frame payload.
		◦ **any**—Matches against any frame payload .
		◦ **ipv4**—Matches against IPv4 frames.
		• **dest_ip_filter**—Matches against destination IP address filter .
		• **dscp_filter**—Matches against DSCP filter .
		• **fragment_type**—Matches against fragment type filter .
		• **protocol**—Matches against protocol filter .
		• **source_ip_filter**—Matches against source IP address filter .
		◦ **ipv6**—Matches against IPv6 frames .
		◦ **any**—Matches against any IP address, or filter.
		◦ **host** *host_name*—Matches against a specified host .
		◦ **network**—Matches against a network .
		◦ **dest_ip_addr** *dest_add*—Matches against the destination IP address .
		◦ **dest_ip_mask** *dest_mask*—Matches against the destination IP address mask.
		◦ **range** *range_value*—Matches against the specified range .
		◦ **specific** *dscp_filter*—Matches against the specific DSCP filter .
		◦ **frag**—Matches against the specified IP fragment type .
		◦ **non_frag**—Matches against the non fragment type .
		◦ **specific** *protocol_value*—Matches against the specific protocol value .
		◦ **tcp**—Matches against the TCP value .

	Command or Action	Purpose
		◦ **udp**—Matches against the UDP value . ◦ **source_ip_addr** *source_addr*—Matches against the source IP address . ◦ **source_ip_mask** *source_mask*—Matches against the source IP address mask. ◦ **dest_ip_addr_32bits** *dest_add*—Matches against the destination IP address. ◦ **dest_ip_mask_32bits** *dest_mask*—Matches against the destination IP address mask. ◦ **source_ip_addr_32bits** *source_add*—Matches against the source IP address. ◦ **source_ip_mask_32bits** *source_mask*—Matches against the source IP address mask.
Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the ucs controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS mode.

Configuration Example

The example shows how to configure QCE Control Ingress Match frame type parameters on the controller:

```
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dest_ip_filter
any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dest_ip_filter
host host1
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dest_ip_filter
network dest_ip_addr addr2
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dscp_filter host
 any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 dscp_filter range
 3-4
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 fragment_type
frag
```

```
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 protocol specific
 45
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv4 source_ip_filter
 network source_ip_mask soumask
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 dest_ip_filter
any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 dest_ip_filter
specfic dest_ip_addr_32its dest34
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 dscp_filter
specific 45 any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 protocol specific
 450
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 protocol specific
 45
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match frame_type ipv6 source_ip_filter
 specific source_ip_mask source-mask

UCS(ProvisionQos)# setQCE review
Commands in queue:
       setQCE QCE_configuration control ingress_match inner_tag_match match_fields inner_cos
 val_0-3
        setQCE QCE_configuration control ingress_match outer_tag_match match_fields cos
val_2-3
        setQCE QCE_configuration control ingress_match frame_type ipv6 dest_ip_filter any
       setQCE QCE_configuration control ingress_match frame_type ipv6 dscp_filter specific
 45
       setQCE QCE_configuration control ingress_match frame_type ipv6 protocol pecific 450

        setQCE QCE_configuration control ingress_match frame_type ipv6 source_ip_filter
specific source_ip_mask_32bits source-mask

UCS(ProvisionQos)# setQCE commit
UCS(ProvisionQos)# exit
```

Configuring QCE Control Ingress Inner Tag Match Parameter on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setQCEQCE_configuration**{**control** {**actions** |**ingress_match**} | **qce-id**} **Example:** `UCS(ProvisionQos)#` **`setQCE QCEconfiguration control action mark_Cos 4`** `UCS(ProvisionQos)#` **`setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`** `UCS(ProvisionQos)#` **`setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`** `UCS(ProvisionQos)#` **`setQCE QCEconfiguration qce-id 4`**	Configures QCE. • **control**—Configures QCE. • **actions**—Configures QCE actions. • **ingress_match**—Configures ingress match. • **qce-id**—Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.
Step 2	**setQCE QCE_configuration control ingress_match inner_tag_match** {**match_fields** {**inner_cos inner_cos_xx** | **vlan_id_filter** {**any** | **range** *vlan_range*	• **control**—Configures QCE. • **ingress_match**—Configures ingress match.

	Command or Action	Purpose

	| **specifc** *specific_vlan*}} | **match_type** {**any** | **c-tagged** | **s_tagged** | **tagged** | **untagged**}}
	
	Example:
	
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields vlan_id_filter any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields vlan_id_filter range range1`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields vlan_id_filter specific 450`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_type c_tagged`
		◦ **inner_tag_match**—Matches against inner tag value.
		◦ **match_fields**—Matches against tag fields .
		◦ **inner-cos inner_cos_xx**—Matches against inner packet Cos value.
		◦ **val_0-1**—Specifies packet COS 0-1.
		◦ **val_0-3**—Specifies packet COS 0-3.
		◦ **val_0-only**—Specifies packet COS 0.
		◦ **val_1-only**—Specifies packet COS 1.
		◦ **val_2-3**—Specifies packet COS 2-3.
		◦ **val_2-only**—Specifies packet COS 2-only.
		◦ **val_3-only**—Specifies packet COS 3-only.
		◦ **val_4-5**—Specifies packet COS 4-5.
		◦ **val_4-7**—Specifies packet COS 4-7.
		◦ **val_4-only**—Specifies packet COS 4-only.
		◦ **val_5-only**—Specifies packet COS 5-only.
		◦ **val_6-7**—Specifies packet COS 6-7.
		◦ **val_6-only**—Specifies packet COS 6.
		◦ **val_7-only**—Specifies packet COS 7.
		◦ **val_any**—Specifies packet COS any.
		◦ **vlan_id_filter**—Matches against VLAN ID filter.
		◦ **any**—Matches against any VLAN.
		◦ **range** *vlan_range*—Matches against the specified VLAN range .
		◦ **specific** *specific_vlan*—Matches against the specific VLAN. The valid range is from 1 to 4095.
		◦ **match_type**—Matches against tag fields.
		• **any**—Matches against any tagged .
		• **c-tagged**—Matches against C tagged .
		• **s-tagged**—Matches against S tagged .
		• **tagged**—Matches against tagged .

	Command or Action	Purpose
		• **untagged**—Matches against untagged .
Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the ucs controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS pmode.

Configuration Example

The example shows how to configure QCE Control Match Ingress inner tag parameters on the ucs controller:

```
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
inner_cos val_0-1
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
vlan_id_filter any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
vlan_id_filter range range1
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
vlan_id_filter specific 450
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_type c_tagged

UCS(ProvisionQos)# setQCE review
Commands in queue:
       setQCE QCE_configuration control ingress_match inner_tag_match match_fields inner_cos
 val_0-1
        setQCE QCE_configuration control ingress_match inner_tag_match match_fields
vlan_id_filter any
        setQCE QCE_configuration control ingress_match inner_tag_match match_fields
vlan_id_filter range range1
        setQCE QCE_configuration control ingress_match inner_tag_match match_fields
vlan_id_filter specific 450
        setQCE QCE_configuration control ingress_match inner_tag_match match_type c_tagged

UCS(ProvisionQos)# setQCE commit
UCS(ProvisionQos)# exit
```

Configuring QCE Control Ingress MAC Params Parameter on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setQCEQCE_configuration**{**control** {**actions** |**ingress_match**} | **qce-id**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`
	`UCS(ProvisionQos)# setQCE QCEconfiguration qce-id 4`
		Configures QCE.
		• **control**—Configures QCE.
		• **actions**—Configures QCE actions.
		• **ingress_match**—Configures ingress match.
		• **qce-id**—Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

Step 2	**setQCE QCE_configuration** {**control ingress_match mac_params** {**dmac_filter** {**any** | **broadcast** | **multicast** | **specific** *specific_fillter* | **unicast**} |**smac_filter** {**any** | **specific** *specific_filter*}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match mac_params dmac_filter any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match mac_params dmac_filter broadcast`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match mac_params dmac_filter specific filter1`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match mac_params smac_filter any`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match mac_params smac_filter specific filter2`
		• **control**—Configures QCE.
		• **ingress_match**—Configures ingress match.
		• **mac_params**—Configures MAC filters.
		• **dmac_filter**—Configures destination MAC filters.
		• **smac_filter**—Configures source MAC filters.
		• **any**—Configures any MAC filter.
		• **broadcast**—Configures any broadcast MAC filter.
		• **multicast**—Configures any multicast MAC filter.
		• **specific** *specific_filter*—Configures specific MAC filter.

Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the ucs controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS pmode.

Configuration Example

The example shows how to configure QCE Control Match Ingress MAC params parameters on the ucs controller:

```
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
inner_cos val-0-1
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
vlan_id_filter any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
vlan_id_filter range range1
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_fields
vlan_id_filter specific 450
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match inner_tag_match match_type c-tagged

UCS(ProvisionQos)# setQCE review
Commands in queue:
        setQCE QCE_configuration control ingress_match mac_params dmac_filter broadcast
        setQCE QCE_configuration control ingress_match mac_params smac_filter specific
source1
        setQCE QCE_configuration control ingress_match mac_params dmac_filter specific
filter1
        setQCE QCE_configuration control ingress_match mac_params smac_filter specific
filter2

UCS(ProvisionQos)# setQCE commit
UCS(ProvisionQos)# exit
```

Configuring QCE Control Ingress Outer Tag Match Parameter on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setQCEQCE_configuration**{**control** {**actions** |**ingress_match**} | **qce-id**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`
	`UCS(ProvisionQos)# setQCE QCEconfiguration qce-id 4`
		Configures QCE.
		• **control**—Configures QCE.
		• **actions**—Configures QCE actions.
		• **ingress_match**—Configures ingress match.
		• **qce-id**—Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

Step 2	**setQCE QCE_configuration** {**control ingress_match outer_tag_match** {**match_fields** {**cos cos_xx** | **vlan_id_filter** {**any** | **range** *vlan_range* | **specifc** *specific_vlan*} | **match_type** {**any** | **c-tagged** | **s_tagged** | **tagged** | **untagged**}	• **control**—Configures QCE. • **ingress_match**—Configures ingress match. ◦ **outer_tag_match**—Matches against the outer tag value.

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td></td><td>

Example:

```
UCS(ProvisionQos)# setQCE QCE_configuration
ingress_match outer_tag_match match_fields cos
 val_0-1
UCS(ProvisionQos)# setQCE QCE_configuration
ingress_match outer_tag_match match_fields
vlan_id_filter any
UCS(ProvisionQos)# setQCE QCE_configuration
ingress_match outer_tag_match match_fields
vlan_id_filter range range1
UCS(ProvisionQos)# setQCE QCE_configuration
ingress_match outer_tag_match match_fields
vlan_id_filter specific 230
UCS(ProvisionQos)# setQCE QCE_configuration
ingress_match outer_tag_match match_type
c-tagged
```

</td><td>

- ◦ **match_fields**—Matches against outer tag fields .
  - ◦ **cos cos_xx**—Matches against packet Cos value.
    - ◦ **val_0-1**—Specifies packet COS 0-1.
    - ◦ **val_0-3**—Specifies packet COS 0-3.
    - ◦ **val_0-only**—Specifies packet COS 0.
    - ◦ **val_1-only**—Specifies packet COS 1.
    - ◦ **val_2-3**—Specifies packet COS 2-3.
    - ◦ **val_2-only**—Specifies packet COS 2-only.
    - ◦ **val_3-only**—Specifies packet COS 3-only.
    - ◦ **val_4-5**—Specifies packet COS 4-5.
    - ◦ **val_4-7**—Specifies packet COS 4-7.
    - ◦ **val_4-only**—Specifies packet COS 4-only.
    - ◦ **val_5-only**—Specifies packet COS 5-only.
    - ◦ **val_6-7**—Specifies packet COS 6-7.
    - ◦ **val_6-only**—Specifies packet COS 6.
    - ◦ **val_7-only**—Specifies packet COS 7.
    - ◦ **val_any**—Specifies packet COS any.
  - ◦ **vlan_id_filter**—Matches against VLAN ID filter.
  - ◦ **any**—Matches against any VLAN.
  - ◦ **range** *vlan_range*—Matches against the specified VLAN range .
  - ◦ **specific** *specific_vlan*—Matches against the specific VLAN. The valid range is from 1 to 4095.
- ◦ **match_type**—Matches against tag fields.
  - **any**—Matches against any tagged .
  - **c-tagged**—Matches against C tagged .
  - **s-tagged**—Matches against S tagged .
  - **tagged**—Matches against tagged .
  - **untagged**—Matches against untagged .

</td></tr>
</table>

	Command or Action	Purpose
Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the ucs controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS pmode.

Configuration Example

The example shows how to configure QCE Control Match Ingress outer tag parameters on the controller:

```
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match outer_tag_match match_fields cos
 val_0-1
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match outer_tag_match match_fields
vlan_id_filter any
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match outer_tag_match match_fields
vlan_id_filter range range1
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match outer_tag_match match_fields
vlan_id_filter specific 230
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match outer_tag_match match_type c-tagged

UCS(ProvisionQos)# setQCE review
Commands in queue:
       setQCE QCE_configuration control ingress_match outer_tag_match match_fields
vlan_id_filter specific 230
        setQCE QCE_configuration control ingress_match outer_tag_match match_fields
vlan_id_filter range vlan2
        setQCE QCE_configuration control ingress_match outer_tag_match match_fields cos
val_0-1
        setQCE QCE_configuration control ingress_match outer_tag_match match_type c_tagged
UCS(ProvisionQos)# setQCE commit
UCS(ProvisionQos)# exit
```

Configuring QCE Control Ingress Ports Parameter on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setQCEQCE_configuration**{**control** {**actions** |**ingress_match**} | **qce-id**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCEconfiguration control action mark_Cos 4`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_fields inner_cos val_0-1`
	`UCS(ProvisionQos)# setQCE QCEconfiguration ingress_match frame_type any match_type vlan c_tagged`
	`UCS(ProvisionQos)# setQCE QCEconfiguration qce-id 4`
		Configures QCE.
		• **control—**Configures QCE.
		• **actions—**Configures QCE actions.
		• **ingress_match—**Configures ingress match.
		• **qce-id—**Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

Step 2	**setQCE QCE_configuration control ingress_match ports** {**GigabitEthernet_1** | **GigabitEthernet_2** | **GigabitEthernet_3** | **GigabitEthernet_4** | **GigabitEthernet_5** | **GigabitEthernet_6**} {**enable** | **disable**}
	
	Example:
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match ports GigabitEthernet_1 enable`
	`UCS(ProvisionQos)# setQCE QCE_configuration ingress_match ports GigabitEthernet_3 disable`
		• **control—**Configures QCE.
		• **ingress_match—**Configures ingress match.
		• **ports—**Configures ingress ports.
		• **GigabitEthernet_1—**Configures physical port 1.
		• **GigabitEthernet_2—**Configures physical port 2.
		• **GigabitEthernet_3—**Configures physical port 3.
		• **GigabitEthernet_4—**Configures physical port 4.
		• **GigabitEthernet_5—**Configures physical port 5.
		• **GigabitEthernet_6—**Configures physical port 6.
		• **enable—**Enables the port.
		• **disable—**Disables the port.

Step 3	**setQCE review** **Example:** `UCS(ProvisionQos)# setQCE review`	Displays the QoS configuration on the ucs controller.
Step 4	**setQCE commit** **Example:** `UCS(ProvisionQos)# setQCE commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)# exit`	Exits the QoS pmode.

Configuration Example

The example shows how to configure QCE Control Match Ingress ports on the ucs controller:

```
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match ports GigabitEthernet_1 enable
UCS(ProvisionQos)# setQCE QCE_configuration ingress_match ports GigabitEthernet_3 disable

UCS(ProvisionQos)# setQCE review
Commands in queue:
        setQCE QCE_configuration control ingress_match ports GigabitEthernet_1 enable
        setQCE QCE_configuration control ingress_match ports GigabitEthernet_3 disable

UCS(ProvisionQos)# setQCE commit
UCS(ProvisionQos)# exit
```

Configuring System QoS on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setSystemQosSettings** {**commit** | **flush** | **system_qos_config** | **review**}
	
	Example:
	`UCS(ProvisionQos)# setSystemQosSettings ?`
	`commit             commit setSystemQosSettings`
	`flush              flush all setSystemQosSettings commands from queue`
	`review             review setSystemQosSettings commands`
	`system_qos_config  set System-wide QoS settings`
		Sets the system QoS configuration.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **system_qos_config**—Sets the system wide QoS settings on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**setSystemQosSettings system_qos_config WRED** {**egress_class_0** | **egress_class_1** | **egress_class_2** | **egress_class_3** | **egress_class_4** | **egress_class_5**} {**max_threshold** *threshold_value*| **min_threshold** *threshold_value* | **state** {**enable** | **disable**}}
	
	Example:
	`UCS(ProvisionQos)# setSystemQosSettings system_qos_config WRED egress_class_0 max_threshold 20`
	`UCS(ProvisionQos)# setSystemQosSettings system_qos_config WRED egress_class_1 min_threshold 40`
	`UCS(ProvisionQos)# setSystemQosSettings system_qos_config WRED egress_class_2 state enable`
		Configures system QoS.
		• **WRED**—Enables WRED algorithm for a non-priority queues on all ports.
		• **egress_class_0**—Egress queue 0.
		• **egress_class_1**—Egress queue 1.
		• **egress_class_2**—Egress queue 2.
		• **egress_class_3**—Egress queue 3.
		• **egress_class_4**—Egress queue 4.
		• **egress_class_5**—Egress queue 5.

	Command or Action	Purpose
		• **max_threshold** *threshold_value*—Sets the maximum threshold. • **min_threshold** *threshold_value*—Sets the minimum threshold. • **state**—Sets the WRED state per queue. • **enable**—Enables the WRED. • **disable**—Disables the WRED.

Step 3	**setSystemQosSettings review**
	
	Example:
	`UCS(ProvisionQos)# setSystemQosSettings review`
	
	`Commands in queue:`
	`setSystemQosSettings system_qos_config WRED egress_class_0 max_threshold 20`
	`setSystemQosSettings system_qos_config WRED egress_class_1 min_threshold 40`
	`setSystemQosSettings system_qos_config WRED egress_class_2 state enable`
		Displays the QoS configuration on the controller.

Step 4	**setSystemQosSettingscommit** **Example:** `UCS(ProvisionQos)# setSystemQosSettings commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example shows how to configure QOS system settings on the ucs controller:

```
UCS(ProvisionQos)# setSystemQosSettings system_qos_config WRED egress_class_0 max_threshold
 20
UCS(ProvisionQos)# setSystemQosSettings system_qos_config WRED egress_class_1 min_threshold
 40
UCS(ProvisionQos)# setSystemQosSettings system_qos_config WRED egress_class_2 state enable
UCS(ProvisionQos)# setSystemQosSettings review

Commands in queue:
        setSystemQosSettings system_qos_config WRED egress_class_0 max_threshold  20
        setSystemQosSettings system_qos_config WRED egress_class_1 min_threshold  40
        setSystemQosSettings system_qos_config WRED egress_class_2 state enable
UCS(ProvisionQos)# setSystemQosSettings commit
UCS(ProvisionQos)# exit
```

Configuring Hierarchical QoS on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setsetHqosId** {**commit** | **flush** | **hqos_id_config** | **review**}
	
	Example:
	
	`UCS(ProvisionQos)# setHqosId ?`
	`  commit           commit setHqosId`
	`  flush            flush all setHqosId commands from queue`
	`  hqos_id_config   setHqosId (default)`
	`  review           review setHqosId commands`
		Sets the hierarchical QoS configuration.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **hqos_id_config**—Sets the hierarchical QoS ID on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**setsetHqosId hqos_id_config** {**bandwidth** {**rate** *kbps* | **state** {**enable** | **disable**}} | **hqos_id** *hqos-id* | **match** {**egress_class_0** | **egress_class_1** | **egress_class_2** | **egress_class_3** | **egress_class_4** | **egress_class_5** | **egress_class_6** | **egress_class_7**} {**bandwidth priority-level** *priority*} | **shaper** {**rate** *kbps* | **state** {**enable** | **disable**}} | **port_number** *port-number* | **shaper** {**rate** *kbps* | **state** {**enable** | **disable**}}
	
	Example:
	
	`UCS(ProvisionQos)# sethqosid hqos_id_config bandwidth rate 100`
	`UCS(ProvisionQos)# sethqosid hqos_id_config bandwidth state enable`
	`UCS(ProvisionQos)# sethqosid hqos_id_config hqos_id 4`
	`UCS(ProvisionQos)# sethqosid hqos_id_config match egress_class_7 bandwidth priority_level 1`
	`UCS(ProvisionQos)# sethqosid hqos_id_config match egress_class_7 shaper rate 100`
	`UCS(ProvisionQos)# sethqosid hqos_id_config match egress_class_7 shaper state enable`
	`UCS(ProvisionQos)# sethqosid hqos_id_config port 2`
	`UCS(ProvisionQos)# sethqosid hqos_id_config shaper rate 100`
	`UCS(ProvisionQos)# sethqosid hqos_id_config shaper state enable`
		Configures hierarchical QoS.
		• **bandwidth**—Specifies bandwidth for logical interface.
		• **rate**—Specifies bandwidth rate in kbps. The valid range if from 100 to 1000000.
		• **state**—Specifies bandwidth state.
		• **enable**—Enables bandwidth state.
		• **disble**—Disables bandwidth state.
		• **hqos_id** *hqos-id*—Specifies HQos ID. The valid range is 0 to 256. 0 is invalid.
		• **match** —Specifies HQoS match queues.
		• **egress_class_0**—Egress queue 0; lowest priority
		• **egress_class_1**—Egress queue 1.
		• **egress_class_2**—Egress queue 2.
		• **egress_class_3**—Egress queue 3.
		• **egress_class_4**—Egress queue 4.
		• **egress_class_5**—Egress queue 5.
		• **egress_class_6**—Egress queue 6; higher priority.
		• **egress_class_7**—Egress queue 7; highest priority.

	Command or Action	Purpose
		• **bandwidhtpriority-level***priority***—**Sets the bandwidth priority scheduling level in strict mode. The valid values are 1-1. • **shaper—**Sets the queue level shaper. • **port_number** *port-number***—**Sets the port number. The valid range is from 1 to 6. Port 7 is invlaid. • **shaper—**Sets the interface level shaper. • **disable—**Disables the WRED.

Step 3	**setHqosId review**
	
	Example:
	`UCS(ProvisionQos)# setHqosId review`
	`Commands in queue:`
	`setHqosId hqos_id_config bandwidth rate 100`
	`setHqosId hqos_id_config bandwidth state enable`
	
	`setHqosId hqos_id_config hqos_id 4`
	`setHqosId hqos_id_config match egress_class_7 bandwidth priority_level 1`
	
	`setHqosId hqos_id_config match egress_class_7 shaper rate 100`
	`setHqosId hqos_id_config match egress_class_7 shaper state enable`
	`setHqosId hqos_id_config port_number 2`
	`setHqosId hqos_id_config shaper rate 100`
		Displays the HQoS configuration on the controller.

Step 4	**setHqosIdcommit** **Example:** `UCS(ProvisionQos)# setHqosId commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example shows how to configure HQOS on the ucs controller:

```
UCS(ProvisionQos)# sethqosid hqos_id_config bandwidth rate 100
UCS(ProvisionQos)# sethqosid hqos_id_config bandwidth state enable
UCS(ProvisionQos)# sethqosid hqos_id_config hqos_id 4
UCS(ProvisionQos)# sethqosid hqos_id_config match egress_class_7 bandwidth priority_level
1
UCS(ProvisionQos)# sethqosid hqos_id_config match egress_class_7 shaper rate 100
UCS(ProvisionQos)# sethqosid hqos_id_config match egress_class_7 shaper state enable
UCS(ProvisionQos)# sethqosid hqos_id_config port 2
UCS(ProvisionQos)# sethqosid hqos_id_config shaper rate 100
UCS(ProvisionQos)# sethqosid hqos_id_config shaper state enable
```

```
UCS(ProvisionQos)# setHqosId review
Commands in queue:
        setHqosId hqos_id_config bandwidth rate 100
        setHqosId hqos_id_config bandwidth state enable
        setHqosId hqos_id_config hqos_id 4
        setHqosId hqos_id_config match egress_class_7 bandwidth priority_level 1

        setHqosId hqos_id_config match egress_class_7 shaper rate 100
        setHqosId hqos_id_config match egress_class_7 shaper state enable
        setHqosId hqos_id_config port_number 2
        setHqosId hqos_id_config shaper rate 100
UCS(ProvisionQos)# setHqosId commit
SetHqosId Commit Success!!!
UCS(ProvisionQos)# exit
```

Configuring EVC Hierarchical QoS Policy on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the ucs controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setEvcHqosPolicy** {**commit** | **flush** | **hqos_id_config** | **review**}
	
	Example:
	`UCS(ProvisionQos)# setEvcHqosPolicy ?`
	`  commit                 commit setEvcHqosPolicy`
	`  evcHqosPolicyConfig    setEvcHqosPolicy (default)`
	`  flush                  flush all setEvcHqosPolicy`
	`commands from queue`
	`  review               review setEvcHqosPolicy commands`
		Sets the hierarchical QoS configuration on the EVC.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **evcHqosPolicyConfig**—Sets the EVC HQoS policy on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**setEvcHqosPolicy evcHqosPolicyConfig** {**evc_id** *evc-id* | **hqos_id** *hqos-id* | **service_policy** {**attach** | **detach**}
	
	Example:
	`UCS(ProvisionQos)# setEvcHqosPolicy evcHqosPolicyConfig evc_id 1`
	`UCS(ProvisionQos)# setEvcHqosPolicy evcHqosPolicyConfig hqos_id 2`
	`UCS(ProvisionQos)# setEvcHqosPolicy evcHqosPolicyConfig service_policy attach`
		Configures hierarchical QoS on the EVC.
		• **evc_id** *evc-id*—Specifies EVC ID. The valid range is from 1 to 1024
		• **hqos_id** *hqos-id*—Specifies HQos ID. The valid range is 0 to 256. 0 is invalid.
		• **service_policy** —Specifies service policy that should be applied or removed on the EVC.
		• **attach**—Applies the policy on the EVC.
		• **detach**—Detaches the policy on the EVC.

	Command or Action	Purpose

Step 3	**setEvcHqosPolicy review**
	
	Example:
	`UCS(ProvisionQos)# setEvcHqosPolicy review`
	`Commands in queue:`
	`        setEvcHqosPolicy evcHqosPolicyConfig evc_id 1`
	
	`        setEvcHqosPolicy evcHqosPolicyConfig`
	`service_policy attach`
	`        setEvcHqosPolicy evcHqosPolicyConfig`
	`service_policy detach`
	`        setEvcHqosPolicy evcHqosPolicyConfig hqos_id`
	`2`
	`        setEvcHqosPolicy evcHqosPolicyConfig evc_id 1`
		Displays the HQoS EVC configuration on the controller.

Step 4	**setEvcHqosPolicycommit** **Example:** `UCS(ProvisionQos)# setEvcHqosPolicy commit`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example shows how to configure EVC HQOS on the ucs controller:

```
UCS(ProvisionQos)# setEvcHqosPolicy evcHqosPolicyConfig evc_id 1
UCS(ProvisionQos)# setEvcHqosPolicy evcHqosPolicyConfig hqos_id 2
UCS(ProvisionQos)# setEvcHqosPolicy evcHqosPolicyConfig attach
UCS(ProvisionQos)# setEvcHqosPolicy review
Commands in queue:
        setEvcHqosPolicy evcHqosPolicyConfig evc_id 1
        setEvcHqosPolicy evcHqosPolicyConfig service_policy attach
        setEvcHqosPolicy evcHqosPolicyConfig service_policy detach
        setEvcHqosPolicy evcHqosPolicyConfig hqos_id 2
        setEvcHqosPolicy evcHqosPolicyConfig evc_id 1
UCS(ProvisionQos)# setEvcHqosPolicy commit
UCS(ProvisionQos)# exit
```

Reordering QoS Control Entry (QCE) on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**reorderQCEentries** {**commit** | **flush** | **reorder_qce** | **review**}
	
	Example:
	`UCS(ProvisionQoS)#` **`reorderQCEntries ?`**
	`  commit       commit reorderQCEentries`
	`  flush        flush all reorderQCEentries commands from queue`
	`  reorder_qce  reorderQCEentries (default)`
	`  review       review reorderQCEentries commands`
		Reorders the QCE entries.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **reorder_qce**—Reorders the QCE on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**reorderQCEentries reorder_qce** {**qce_id** *qce_id* | **reorder** {**after** | **before** | **last**} {**qce_id** *qce_id*}} **Example:** `UCS(ProvisionQoS)#` **`reorderQCEentries reorder_qce qce_id 3`** `UCS(ProvisionQoS)#` **`reorderQCEentries reorder reorder before qce_id 2`**	• **reorder_qce**—Reorders QCE . • **reorder**—Specifies the reorder operation. • **after**—Reorders after the specified QCE ID. • **before**—Reorders before the specified QCE ID.. • **last**—Reorders QCE ID as last. • **qce-id**— Specifies the QCE ID. The valid range is from 1 to 1024. 0 is invalid.

Step 3	**reorderQCEentries review review**
	
	Example:
	`UCS(ProvisionQoS)#` **`reorderQCEentries review`**
	`Commands in queue:`
	`        reorderQCEentries reorder_qce qce_id 3`
	`        reorderQCEentries reorder_qce qce_id 3`
	`        reorderQCEentries reorder_qce reorder before qce_id 2`
		Displays the QoS configuration on the controller.

Step 4	**setQCE commitcommit** **Example:** `UCS(ProvisionQoS)#` **`reorderQCEentries commit`** `ReorderQCEentries Commit Success!!!`	Sends the QoS configuration to the NID.
Step 5	**exit** **Example:** `UCS(ProvisionQos)#`**`exit`**	Exits the QoS mode.

Configuration Example

The example shows how to reorder QoS QCE on the ucs controller:

```
UCS(ProvisionQoS)# reorderQCEentries reorder_qce qce_id 3
UCS(ProvisionQoS)# reorderQCEentries reorder reorder before qce_id 2

UCS(ProvisionQoS)# reorderQCEentries review
Commands in queue:
        reorderQCEentries reorder_qce qce_id 3
        reorderQCEentries reorder_qce qce_id 3
        reorderQCEentries reorder_qce reorder before qce_id 2
UCS(ProvisionQoS)# reorderQCEentries commit
reorderQCEentries commit
UCS(ProvisionQoS)# exit
```

Deleting QoS Control Entry (QCE) on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**deleteQCE** {**commit** | **flush** | **delete_qce** *qce-id* | **review**}
	
	Example:
	`UCS(ProvisionQos)# deleteQCE ?`
	`commit      commit deleteQCE`
	`delete_qce  Delete a particular QCE`
	`flush       flush all deleteQCE commands from queue`
	
	`review      review deleteQCE commands`
	`UCS(ProvisionQos)# deleteQCE delete_qce 2`
		Deletes QoS configuration.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **delete_qce** *qce-id*—Deletes the QCE ID on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**deleteQCE review**
	
	Example:
	`UCS(ProvisionQos)# deleteQCE review`
	`Commands in queue:`
	`        deleteQCE delete_qce 3`
		Displays the QoS configuration on the controller.

Step 3	**deleteQCE commitcommit**
	
	Example:
	`UCS(ProvisionQos)# deleteQCE commit`
	`DeleteQCE Commit Success!!!`
		Sends the QoS configuration to the NID.

	Command or Action	Purpose
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example shows how to delete QoS QCE on the ucs controller:

```
UCS(ProvisionQos)# deleteQCE delete_qce 2
UCS(ProvisionQos)# deleteQCE review
Commands in queue:
        deleteQCE delete_qce 3
UCS(ProvisionQos)# deleteQCE commit
DeleteQCE Commit Success!!!
UCS(ProvisionQos)# exit
```

Deleting HQoS ID on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**deleteHqosId** {**commit** | **flush** | **delete_hqos_id** *hqos-id* | **review**}
	
	Example:
	`UCS(ProvisionQos)# deleteHqosId ?`
	`  commit          commit deleteHqosId`
	`  delete_hqos_id  deleteHqosId (default)`
	`  flush          flush all deleteHqosId commands from queue`
	`  review          review deleteHqosId commands`
	`UCS(ProvisionQos)# deleteHqosId delete_hqos_id 2`
		Deletes HQoS ID configuration.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **delete_hqos_id** *hqos-id*—Deletes the HQoS ID on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**deleteHqosId review**
	
	Example:
	`UCS(ProvisionQos)# deleteHqosId review`
	`Commands in queue:`
	`        deleteHqosId delete_hqos_id 2`
		Displays the HQoS ID configuration on the controller.

	Command or Action	Purpose

Step 3	**deleteHqosIdcommit**
	
	Example:
	`UCS(ProvisionQos)# deleteHqosId commit`
	`deleteHqosId Commit Success!!!`
		Sends the QoS configuration to the NID.

Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example shows how to delete HQoS IDon the ucs controller:

```
UCS(ProvisionQos)# deleteHqosId delete_hqos_id 2
UCS(ProvisionQos)# deleteHqosId review
Commands in queue:
        deleteHqosId delete_hqos_id 2
UCS(ProvisionQos)# deleteHqosId commit
DeleteHqosId Commit Success!!!
UCS(ProvisionQos)# exit
```

Negating QoS and Restoring Defaults

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**no ?**
	
	Example:
	`UCS(ProvisionQoS)# no ?`
	
	`deleteQCE               Delete a particular QCE`
	`exit                    Exit from ProvisionQos sub configuration mode`
	`getInputGlobalPolicy    Show Output QoS global features configured`
	`getInputPortPolicy      Show Input Policy configured on Physical Port`
	`getOutputGlobalPolicy   Show Global Output QoS features`
	`getOutputPortPolicy     Show Output Policy configured on Physical Port`
	`getQCE                  getQCE (default)`
	`getSystemQosSettings    getSystemQosSettings (default)`
	`reorderQCEentries       reorderQCEentries (default)`
	`setInputGlobalPolicy    configure Global Input QoS features`
	`setInputPortPolicy      configure Input policy on Physical Port`
	`setOutputGlobalPolicy   configure Global Output QoS features`
		Negates the commands and sets the default configuration.

	Command or Action	Purpose

	`setOutputPortPolicy    configure Output policy on Physical Port`
	`setQCE                 setQCE (default)`
	`setSystemQosSettings   set System-wide QoS settings`
	`showQCElist            showQCElist (default)`
	`showQueueStatistics    Display egress queue statistics`	

Step 2	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Viewing QoS Input Policy Global Features on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getInputGlobalPolicy** {**commit** | **flush** | **input** | **review**}
	
	Example:
	`UCS(ProvisionQos)# getInputGlobalPolicy ?`
	`  commit   commit getInputGlobalPolicy`
	`flush   flush all getInputGlobalPolicy commands from`
	` queue`
	`input   Show Output QoS global features configured`
	`review  review getInputGlobalPolicy commands`
	
	`UCS(ProvisionQos)# getInputGlobalPolicy input`
		View the global input QoS policy.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **input** —Displays the input policy QoS global features configured the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**getInputGlobalPolicy review**
	
	Example:
	`UCS(ProvisionQos)# getInputGlobalPolicy review`
	`Commands in queue:`
	`        getInputGlobalPolicy input`
		Displays the QoS configuration on the ucs controller.

Step 3	**getInputGlobalPolicy commit** **Example:** `UCS(ProvisionQos)# getInputGlobalPolicy commit`	Sends the QoS configuration to the NID.

	Command or Action	Purpose
Step 4	**exit** **Example:** `UCS(ProvisionQos)#`**`exit`**	Exits the QoS mode.

Configuration Example

The example displays the QoS input port policy on the ucs controller:

```
UCS(ProvisionQos)# getInputGlobalPolicy input
UCS(ProvisionQos)# getInputGlobalPolicy review
CCommands in queue:
        getInputGlobalPolicy input
UCS(ProvisionQos)# getInputGlobalPolicy commit

GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_00.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_00.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_01.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_01.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_02.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_02.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_03.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_03.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_04.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_04.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_05.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_05.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_06.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_06.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_07.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_07.mark_egress_class
= 8
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_08.mark_DSCP = 64
GetInputGlobalPolicy_Output.inputGlobalPolicyConfig.match_DSCP.value_08.mark_egress_class
= 8
.
.
!
 GetInputGlobalPolicy Commit Success!!!


UCS(ProvisionQos)# exit
```

Viewing QoS Input Policy Features at Port level on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getInputPortPolicy** {**commit** | **flush** | **input_port** *port_number*| **review**}
	
	Example:
	`UCS(ProvisionQos)# getInputPortPolicy ?`
	`  commit      commit getInputPortPolicy`
	`  flush       flush all getInputPortPolicy commands from queue`
	`  input_port  Show Input Policy configured on Physical Port`
	`  review      review getInputPortPolicy commands`
	`UCS(ProvisionQos)# getInputPortPolicy input_port 2`
		View the input QoS policy at port level.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **input_port** *port_number*—Displays the input port policy configuration at port level on the controller. The valid ports are 1 to 6. port 7 is invalid.
		• **review**—Displays the configuration on the controller.

Step 2	**getInputPortPolicy review**
	
	Example:
	`UCS(ProvisionQos)# getInputPortPolicy review`
	
	`Commands in queue:`
	`        getInputPortPolicy input_port 3`
	`        getInputPortPolicy input_port 2`
		Displays the QoS configuration on the controller.

Step 3	**getInputPortPolicy commit** **Example:** `UCS(ProvisionQos)# getInputPortPolicy commit`	Sends the QoS configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the QoS input port policy on the ucs controller:

```
UCS(ProvisionQos)# getInputPortPolicy input_port 2
UCS(ProvisionQos)# getInputPortPolicy review
Commands in queue:
        getInputPortPolicy input_port 3
        getInputPortPolicy input_port 2
UCS(ProvisionQos)# getInputPortPolicy commit
GetInputPortPolicy_Output.inputPortPolicyConfig.port_number = 2
GetInputPortPolicy_Output.inputPortPolicyConfig.port_policer.state = false
GetInputPortPolicy_Output.inputPortPolicyConfig.port_policer.cir = 1000000
GetInputPortPolicy_Output.inputPortPolicyConfig.globalDscpBasedDscpIngressMarking = false
GetInputPortPolicy_Output.inputPortPolicyConfig.globalDscpBasedEgressClassMarking = false
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_0.mark_egress_class = 1
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_1.mark_egress_class = 0
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_2.mark_egress_class = 2
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_3.mark_egress_class = 3
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_4.mark_egress_class = 4
```

```
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_5.mark_egress_class = 5
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_6.mark_egress_class = 6
GetInputPortPolicy_Output.inputPortPolicyConfig.match.cos_.value_7.mark_egress_class = 7
GetInputPortPolicy_Output.inputPortPolicyConfig.egress_class_marking = false
GetInputPortPolicy_Output.inputPortPolicyConfig.qce.address.t = 1
GetInputPortPolicy_Output.inputPortPolicyConfig.qce.address.u.source = ''
GetInputPortPolicy_Output.inputPortPolicyConfig.qce.key.t = 1
GetInputPortPolicy_Output.inputPortPolicyConfig.qce.key.u.normal = ''
GetInputPortPolicy_Output.inputPortPolicyConfig.service_policy.t = 2
GetInputPortPolicy_Output.inputPortPolicyConfig.service_policy.u.detach = ''

 GetInputPortPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Viewing QoS Output Policy Global Features on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getInputGlobalPolicy** {**commit** | **flush** | **output** | **review**}
	
	Example:
	`UCS(ProvisionQos)# getInputGlobalPolicy ?`
	`  commit   commit getOutputGlobalPolicy`
	`  flush    flush all getOutputGlobalPolicy commands from queue`
	`  output   Show Global Output QoS features`
	`  review   review getOutputGlobalPolicy commands`
	
	`UCS(ProvisionQos)# getInputGlobalPolicy output`
		View global output QoS policy.
		• **commit—**Sends the QoS configuration to NID.
		• **flush—**Flushes all QoS configuration from the queue.
		• **input —**Displays the input policy QoS global features configured the controller.
		• **review—**Displays the configuration on the controller.

Step 2	**getOutputPortPolicy review**
	
	Example:
	`UCS(ProvisionQos)# getInputGlobalPolicy review`
	`Commands in queue:`
	`        getOutputGlobalPolicy output`
		Displays the QoS configuration on the controller.

Step 3	**getOutputPortPolicy commit** **Example:** `UCS(ProvisionQos)# getInputGlobalPolicy commit`	Sends the QoS configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the QoS output policy global features on the ucs controller:

```
UCS(ProvisionQos)# getInputGlobalPolicy output
UCS(ProvisionQos)# getInputGlobalPolicy review
Commands in queue:
        getOutputGlobalPolicy output

UCS(ProvisionQos)# getInputGlobalPolicy commit
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_00.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_01.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_02.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_03.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_04.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_05.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_06.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_07.mark_DSCP = 64
GetOutputGlobalPolicy_Output.outputGlobalPolicyConfig.match_DSCP.value_08.mark_D

 GetOutputGlobalPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Viewing QoS Output Policy Features at Port level on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getOutputPortPolicy** {**commit** | **flush** | **output_port** *port_number*| **review**}
	
	Example:
	`UCS(ProvisionQos)# getOutputPortPolicy ?`
	`  ccommit       commit getOutputPortPolicy`
	`  flush       flush all getOutputPortPolicy commands`
	` from queue`
	`  output_port  Show Output Policy configured on`
	`Physical Port`
	`  review       review getOutputPortPolicy commands`
	`UCS(ProvisionQos)# getOutputPortPolicy output_port 4`
		View the output Qos policy at port level.
		• **commit—**Sends the QoS configuration to NID.
		• **flush—**Flushes all QoS configuration from the queue.
		• **output_port** *port_number***—**Displays the output port policy configuration at port level on the controller. The valid ports are 1 to 6. port 7 is invalid.
		• **review—**Displays the configuration on the controller.

Step 2	**getOutputPortPolicy review**
	
	Example:
	`UCS(ProvisionQos)# getOutputPortPolicy review`
	
	`Commands in queue:`
	`        getOutputPortPolicy output_port 4`
		Displays the QoS configuration on the controller.

	Command or Action	Purpose
Step 3	**getOutputPortPolicy commit** **Example:** `UCS(ProvisionQos)# getOutputPortPolicy commit`	Sends the QoS configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the QoS output port policy on the ucs controller:

```
UCS(ProvisionQos)# getOutputPortPolicy output_port 4
UCS(ProvisionQos)# getOutputPortPolicy review
Commands in queue:
        getOutputPortPolicy output_port 4

UCS(ProvisionQos)# getOutputPortPolicy commit
GetOutputPortPolicy_Output.outputPortPolicyConfig.port_number = 4
GetOutputPortPolicy_Output.outputPortPolicyConfig.port_shaper.state = false
GetOutputPortPolicy_Output.outputPortPolicyConfig.port_shaper.rate = 1000000
GetOutputPortPolicy_Output.outputPortPolicyConfig.globalDscpBasedDscpEgressMarking = false
GetOutputPortPolicy_Output.outputPortPolicyConfig.match.egress_class_7.bandwidth.priority_level
 = 1
GetOutputPortPolicy_Output.outputPortPolicyConfig.match.egress_class_7.shaper.state = false
GetOutputPortPolicy_Output.outputPortPolicyConfig.match.egress_class_7.shaper.rate = 1000000
GetOutputPortPolicy_Output.outputPortPolicyConfig.match.egress_class_7.mark_cos= 7
GetOutputPortPolicy_Output.outputPortPolicyConfig.match.egress_class_6.bandwidth.priority_level
 = 2
GetOutputPortPolicy_Output.outputPortPolicyConfig.match.egress_class_6.shaper.state = false
GetOutputPortPolicy_Output.outputPortPolicyConfig.match.egress_class_6.shaper.rate = 1000000
.
.
!
GetOutputPortPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Viewing QoS Control Entry (QCE) Configuration on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the Controller to Configure QoS.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**getQCE** {**commit** | **flush** | **QCE_ID** *qce_id*| **review**}	View the QCE configuration.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionQos)# getOutputPortPolicy ?`
	`  QCE_ID  getQCE (default)`
	`  commit  commit getQCE`
	`  flush   flush all getQCE commands from queue`
	`  review  review getQCE commands`
	`UCS(ProvisionQos)# getOutputPortPolicy qce-id 4`
		• **commit—**Sends the QoS configuration to NID.
		• **flush—**Flushes all QoS configuration from the queue.
		• **QCE_ID** *qce_id***—**Displays the QCE configuration for QCE ID on the controller. The valid ports are 1 to 1024.
		• **review—**Displays the configuration on the controller.

Step 2	**getQCE review**
	Example:
	`UCS(ProvisionQos)# getQCE review`
	`Commands in queue:`
	`        getQCE QCE_ID 2`
	`        getQCE QCE_ID 3`
	`        getQCE QCE_ID 23`
		Displays the QoS configuration on the controller.

Step 3	**getOutputPortPolicy commit** **Example:** `UCS(ProvisionQos)# getQCE commit`	Sends the QoS configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the QoS output port policy on the ucs controller:

```
UCS(ProvisionQos)# getOutputPortPolicy qce-id 4
UCS(ProvisionQos)# getQCE review
Commands in queue:
        getQCE QCE_ID 2
        getQCE QCE_ID 3
        getQCE QCE_ID 23
UCS(ProvisionQos)# getQCE commit
GetQCE_Output.QCE_configuration.qce_id = 0
GetQCE_Output.QCE_configuration.control.ingress_match.ports.GigabitEthernet_1 = false
GetQCE_Output.QCE_configuration.control.ingress_match.ports.GigabitEthernet_2 =false
GetQCE_Output.QCE_configuration.control.ingress_match.ports.GigabitEthernet_3 =false
GetQCE_Output.QCE_configuration.control.ingress_match.ports.GigabitEthernet_4 =false
GetQCE_Output.QCE_configuration.control.ingress_match.ports.GigabitEthernet_5 =false
GetQCE_Output.QCE_configuration.control.ingress_match.ports.GigabitEthernet_6 =false
GetQCE_Output.QCE_configuration.control.ingress_match.outer_tag_match.match_type.t = 1
GetQCE_Output.QCE_configuration.control.ingress_match.outer_tag_match.match_type.u.any =
'0'
GetQCE_Output.QCE_configuration.control.ingress_match.outer_tag_match.match_fields.vlan_id_filter.t
 = 1
GetQCE_Output.QCE_configuration.control.ingress_match.outer_tag_match.match_fields.vlan_id_filter.u.any
 = '0'
```

```
GetQCE_Output.QCE_configuration.control.ingress_match.outer_tag_match.match_fields.cos_.t
= 1
GetQCE_Output.QCE_configuration.control.ingress_match.outer_tag_match.match_fields.cos_.u.val_any
 = '0'
GetQCE_Output.QCE_configuration.control.ingress_match.inner_tag_match.match_type.t = 1
GetQCE_Output.QCE_configuration.control.ingress_match.inner_tag_match.match_type.u.any =
'0'
GetQCE_Output.QCE_configuration.control.ingress_match.inner_tag_match.match_fields.vlan_id_filter.t
 = 1
GetQCE_Output.QCE_configuration.control.ingress_match.inner_tag_match.match_fields.vlan_id_filter.u.any
 = '0'
GetQCE_Output.QCE_configuration.control.ingress_match.inner_tag_match.match_fields.inner_cos.t
 = 1
GetQCE_Output.QCE_configuration.control.ingress_match.inner_tag_match.match_fields.inner_cos.u.val_any
 = '0'
GetQCE_Output.QCE_configuration.control.ingress_match.mac_params.smac_filter.t = 1
GetQCE_Output.QCE_configuration.control.ingress_match.mac_params.smac_filter.u.any = '0'
GetQCE_Output.QCE_configuration.control.ingress_match.mac_params.dmac_filter.t = 1
GetQCE_Output.QCE_configuration.control.ingress_match.mac_params.dmac_filter.u.any = '0'
GetQCE_Output.QCE_configuration.control.ingress_match.frame_type.t = 1
GetQCE_Output.QCE_configuration.control.ingress_match.frame_type.u.any = '0'
GetQCE_Output.QCE_configuration.control.actions.mark_egress_class = 8
GetQCE_Output.QCE_configuration.control.actions.mark_COS = 8
GetQCE_Output.QCE_configuration.control.actions.mark_DSCP = 64

 GetQCE Commit Success!!!

GetOutputPortPolicy Commit Success!!!
UCS(ProvisionQos)# exit
```

Viewing System QoS Settings on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the Controller to Configure QoS.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getSystemQosSettings** {**commit** | **flush** | **QCE_ID** *qce_id*| **review**}
	
	Example:
	`UCS(ProvisionQos)# getSystemQosSettings ?`
	`commit      commit getSystemQosSettings`
	`flush       flush all getSystemQosSettings commands from queue`
	`review      review getSystemQosSettings commands`
	`system_qos  getSystemQosSettings (default)`
	
	`UCS(ProvisionQos)# getSystemQosSettings system_qos`
		View the system QoS configuration.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **system_qos**—Displays the system QOS configuration on the controller.
		• **review**—Displays the configuration on the controller.

Step 2	**getSystemQosSettings review**
	
	Example:
	`UCS(ProvisionQos)# getSystemQosSettings review`
	`Commands in queue:`
	`getSystemQosSettings system_qos`
		Displays the QoS configuration on the controller.

	Command or Action	Purpose
Step 3	**getSystemQosSettings commit** **Example:** `UCS(ProvisionQos)# getSystemQosSettings commit`	Sends the QoS configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the system QoS settings on the ucs controller:

```
UCS(ProvisionQos)# getSystemQosSettings system_qos
UCS(ProvisionQos)# getSystemQosSettings review
Commands in queue:
        getSystemQosSettings system_qos

UCS(ProvisionQos)# getSystemQosSettings commit
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_0.state = false
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_0.min_threshold = 0
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_0.max_threshold= 100
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_1.state = false
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_1.min_threshold= 0
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_1.max_threshold= 100
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_2.state = false
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_2.min_threshold= 0
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_2.max_threshold= 100
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_3.state = false
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_3.min_threshold= 0
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_3.max_threshold= 100
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_4.state = false
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_4.min_threshold= 0
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_4.max_threshold= 100
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_5.state = false
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_5.min_threshold= 0
GetSystemQosSettings_Output.system_qos_config.WRED.egress_class_5.max_threshold= 100

 GetSystemQosSettings Commit Success!!!

UCS(ProvisionQos)# exit
```

Viewing HQoS ID on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getHqosId** {**commit** | **flush** | **hqos_id_value** *hqos_id* | **review**}
	
	Example:
	`UCS(ProvisionQos)# getHqosId ?`
	`commit          commit getHqosId`
	`flush           flush all getHqosId commands from queue`
	`hqos_id_value   getHqosId (default)`
	`review          review getHqosId commands`
	
	`UCS(ProvisionQos)# getHqosId hqos_id_value 4`
		View the HQoS ID configuration.
		• **commit—**Sends the QoS configuration to NID.
		• **flush—**Flushes all QoS configuration from the queue.
		• **hqos_id_value** *hqos_id***—**Displays the HQOS ID configuration on the controller. The valid range is from 1 to 256.
		• **review—**Displays the configuration on the controller.

Step 2	**getHqosId review**
	
	Example:
	`UCS(ProvisionQos)# getHqosId review`
	`Commands in queue:`
	`        getHqosId hqos_id_value 2`
		Displays the HQoS ID configuration on the controller.

Step 3	**getHqosId commit** **Example:** `UCS(ProvisionQos)# getHqosId commit`	Sends the HQoS configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the system HQoS ID on the ucs controller:

```
UCS(ProvisionQos)# getHqosId hqos_id_value 4
UCS(ProvisionQos)# getHqosId review
Commands in queue:
        getHqosId hqos_id_value 2
UCS(ProvisionQos)# getHqosId commit
GetHqosId_Output.hqos_id_config.hqos_id = 4
GetHqosId_Output.hqos_id_config.port_number = 4
GetHqosId_Output.hqos_id_config.shaper.state = true
GetHqosId_Output.hqos_id_config.shaper.rate = 100000
GetHqosId_Output.hqos_id_config.bandwidth.state = true
GetHqosId_Output.hqos_id_config.bandwidth.rate = 10000
GetHqosId_Output.hqos_id_config.match.egress_class_7.bandwidth.priority_level = 1
GetHqosId_Output.hqos_id_config.match.egress_class_7.shaper.state = true
GetHqosId_Output.hqos_id_config.match.egress_class_7.shaper.rate = 40000
GetHqosId_Output.hqos_id_config.match.egress_class_6.bandwidth.priority_level = 2
GetHqosId_Output.hqos_id_config.match.egress_class_6.shaper.state = true
GetHqosId_Output.hqos_id_config.match.egress_class_6.shaper.rate = 50000
GetHqosId_Output.hqos_id_config.match.egress_class_5.bandwidth.t = 2
```

```
GetHqosId_Output.hqos_id_config.match.egress_class_5.bandwidth.u.remaining_ratio = 5
GetHqosId_Output.hqos_id_config.match.egress_class_4.bandwidth.t = 2
GetHqosId_Output.hqos_id_config.match.egress_class_4.bandwidth.u.remaining_ratio = 4
GetHqosId_Output.hqos_id_config.match.egress_class_3.bandwidth.t = 2
GetHqosId_Output.hqos_id_config.match.egress_class_3.bandwidth.u.remaining_ratio = 4
GetHqosId_Output.hqos_id_config.match.egress_class_2.bandwidth.t = 2
GetHqosId_Output.hqos_id_config.match.egress_class_2.bandwidth.u.remaining_ratio = 3
GetHqosId_Output.hqos_id_config.match.egress_class_1.bandwidth.t = 2
GetHqosId_Output.hqos_id_config.match.egress_class_1.bandwidth.u.remaining_ratio = 3
GetHqosId_Output.hqos_id_config.match.egress_class_0.bandwidth.t = 2
GetHqosId_Output.hqos_id_config.match.egress_class_0.bandwidth.u.remaining_ratio = 2
GetHqosId Commit Success!!!

UCS(ProvisionQos)# exit
```

Viewing EVC HQoS ID on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getEvcHqosPolicy** {**commit** | **flush** | **evcHqosPolicy evc-id***evc_id* | **review**}
	
	Example:
	`UCS(ProvisionQos)# getEvcHqosPolicy ?`
	` commit         commit getEvcHqosPolicy`
	` evcHqosPolicy  getEvcHqosPolicy (default)`
	` flush          flush all getEvcHqosPolicy commands`
	` from queue`
	` review         review getEvcHqosPolicy commands`
	
	`UCS(ProvisionQos)# getEvcHqosPolicy evcHqosPolicy evc_id 1`
		View the EVC HQoS ID configuration.
		• **commit—**Sends the QoS configuration to NID.
		• **flush—**Flushes all QoS configuration from the queue.
		• **evcHqosPolicy evc-id***evc_id***—**Displays the EVC HQOS ID configuration on the controller. The valid range is from 1 to 1024.
		• **review—**Displays the configuration on the controller.

Step 2	**getEvcHqosPolicy review**
	
	Example:
	`UCS(ProvisionQos)# getEvcHqosPolicy review`
	`Commands in queue:`
	`        getEvcHqosPolicy evcHqosPolicy evc_id 1`
		Displays the EVC HQoS ID configuration on the controller.

Step 3	**getHqosId commit** **Example:** `UCS(ProvisionQos)# getEvcHqosPolicy commit`	Sends the EVC HQoS configuration to the NID.
Step 4	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the EVC HQoS ID on the ucs controller:

```
UCS(ProvisionQos)# getEvcHqosPolicy evcHqosPolicy evc_id 1
UCS(ProvisionQos)# getEvcHqosPolicy review
Commands in queue:
        getEvcHqosPolicy evcHqosPolicy evc_id 1
UCS(ProvisionQos)# getEvcHqosPolicy commit


UCS(ProvisionQos)# exit
```

Displaying the QCE List on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showQCElist** {**commit** | **flush** | **show_qce** {**all** | **specific** *specific_QCE*} | **review**}
	
	Example:
	`UCS(ProvisionQos)#showQCElist show_qce all`
	`UCS(ProvisionQos)#showQCElist show_qce specific 2`
	`UCS(ProvisionQos)#showQCElist review`
	`UCS(ProvisionQos)#showQCElist commit`
		Displays the QCE list.
		• **show_qce**—Displays QCE list.
		• **all**—Displays entire QCE list.
		• **specific** *specific_QCE*—Displays specific QCE list.
		• **commit**—Sends the QoS configuration to the NID.
		• **flush**—Flushes all QoS configuration from the queue.
		• **review**—Displays the QoS configuration on the controller.

Step 2	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the QCE list on the ucs controller:

```
UCS(ProvisionQos)#showQCElist show_qce all
```

```
UCS(ProvisionQos)#showQCElist show_qce specific 2
UCS(ProvisionQos)#showQCElist review
Commands in queue:
        showQCElist show_qce all
        showQCElist show_qce specific 2
UCS(ProvisionQos)#showQCElist commit
ShowQCElist_Output.show_qce_configuration.QCE_List[0].qce_id = 2
ShowQCElist_Output.show_qce_configuration.QCE_List[0].status = false

 ShowQCElist Commit Success!!!
UCS(ProvisionQos)# exit
```

Displaying QoS Queue Statistics on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showQueueStatistics** {**commit** | **flush** | **queue_stats port_number** *port_num*} | **review**}
	
	Example:
	`UCS(ProvisionQoS)#showQueueStatistics queue_stats port_number 3`
	`UCS(ProvisionQoS)#showQueueStatistics review`
	`UCS(ProvisionQoS)#showQueueStatistics commit`
		Displays the QoS queue statistics.
		• **queue_stats**—Displays egress queue statistics.
		• **port_number** *port_num*—Displays statistics for specified port. The valid range is from 1 to 6.
		• **commit**—Sends the QoS configuration to NID.
		• **flush**—Flushes all QoS from the queue.
		• **review**—Displays the QoS configuration on the controller.

Step 2	**exit** **Example:** `UCS(ProvisionQos)#exit`	Exits the QoS mode.

Configuration Example

The example displays the egress queue statistics on the ucs controller:

```
UCS(ProvisionQoS)#showQueueStatistics queue_stats port_number 3
UCS(ProvisionQoS)#showQueueStatistics review
Commands in queue:
        showQueueStatistics queue_stats port_number 3

UCS(ProvisionQoS)#showQueueStatistics commit
ShowQueueStatistics_Output.queue_statistics.port_number = 3
ShowQueueStatistics_Output.queue_statistics.Queue_0.frames = 0
ShowQueueStatistics_Output.queue_statistics.Queue_1.frames = 0
ShowQueueStatistics_Output.queue_statistics.Queue_2.frames = 0
```

```
ShowQueueStatistics_Output.queue_statistics.Queue_3.frames = 0
ShowQueueStatistics_Output.queue_statistics.Queue_4.frames = 0
ShowQueueStatistics_Output.queue_statistics.Queue_5.frames = 0
ShowQueueStatistics_Output.queue_statistics.Queue_6.frames = 0
ShowQueueStatistics_Output.queue_statistics.Queue_7.frames = 0

ShowQueueStatistics Commit Success!!!
UCS(ProvisionQoS)# exit
```

Displaying the Hierarchical QoS ID List on the UCS Controller

Before You Begin

- Perform the steps to provision QoS on the UCS controller. See Provisioning the UCS Controller to Configure QoS, on page 216.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**showHqosIdList** {**commit** | **flush** | **show_hqos_id** {**all** | **specific** *specific_QCE*} | **review**} **Example:** `UCS(ProvisionQos)#`**`showHqosIdList show_hqos_id all`** `UCS(ProvisionQos)#`**`showHqosIdList show_hqos_id specific 2`** `UCS(ProvisionQos)#`**`showHqosIdList review`** `UCS(ProvisionQos)#`**`showHqosIdList commit`**	Displays the HQoS ID list. • **show_hqos_id**—Displays HQoS ID list. • **all**—Displays entire HQoS ID list. • **specific** *specific_hqos-id*—Displays specific HQos ID list. • **commit**—Sends the QoS configuration to the NID. • **flush**—Flushes all QoS configuration from the queue. • **review**—Displays the QoS configuration on the controller.
Step 2	**exit** **Example:** `UCS(ProvisionQos)#`**`exit`**	Exits the QoS mode.

Configuration Example

The example displays the HQoS ID list on the ucs controller:

```
UCS(ProvisionQos)#showHqosIdList show_hqos_id specific 2
UCS(ProvisionQos)#showHqosIdList review
Commands in queue:
        showHqosIdList show_hqos_id all
        showHqosIdList show_hqos_id specific 2
UCS(ProvisionQos)#showHqosIdList commit
ShowHqosIdList_Output.show_hqos_id_response.hqos_id_list[0].hqos_id = 2
ShowHqosIdList_Output.show_hqos_id_response.hqos_id_list[0].status = false
```

```
ShowHqosIdList Commit Success!!!
UCS(ProvisionQos)# exit
```

CHAPTER 17

Configuring Ethernet OAM, Link OAM, and CFM

Ethernet Operations, Administration, and Maintenance (OAM) is a protocol for installing, monitoring, and troubleshooting Ethernet networks to increase management capability within the context of the overall Ethernet infrastructure. The Cisco ME 1200 Series Carrier Ethernet Access Device supports IEEE 802.1ag Connectivity Fault Management (CFM), and IEEE 802.3ah Ethernet OAM discovery, link monitoring, remote fault detection, and remote loopback.

This document provides information about configuring Ethernet OAM, Link OAM, and CFM.

For more information on Ethernet OAM and CFM, see the *Cisco IOS Carrier Ethernet Configuration Guide*.

Understanding the Ethernet OAM Protocol

The Ethernet OAM protocol for installing, monitoring, and troubleshooting Metro Ethernet networks and Ethernet WANs relies on an optional sublayer in the data link layer of the OSI model. Normal link operation does not require Ethernet OAM. You can implement Ethernet OAM on any full-duplex point-to-point or emulated point-to-point Ethernet link for a network or part of a network (specified interfaces).

OAM frames, called OAM protocol data units (OAM PDUs) use the slow protocol destination MAC address 0180.c200.0002. They are intercepted by the MAC sublayer and cannot propagate beyond a single hop within an Ethernet network. Ethernet OAM is a relatively slow protocol, with a maximum transmission rate of 10 frames per second, resulting in minor impact to normal operations. However, when you enable link monitoring, because the CPU must poll error counters frequently, the number of required CPU cycles is proportional to the number of interfaces that must be polled.

OAM Features

These OAM features are defined by IEEE 802.3ah:

- Discovery identifies devices in the network and their OAM capabilities. It uses periodic OAM PDUs to advertise OAM mode, configuration, and capabilities; PDU configuration; and platform identity. An optional phase allows the local station to accept or reject the configuration of the peer OAM entity.
- Link monitoring detects and indicates link faults under a variety of conditions and uses the event notification OAM PDU to notify the remote OAM device when it detects problems on the link. Error events include when the number of symbol errors, the number of frame errors, the number of frame errors within a specified number of frames, or the number of error seconds within a specified period exceed a configured threshold.
- Remote failure indication conveys a slowly deteriorating quality of an OAM entity to its peers by communicating these conditions: Link Fault means a loss of signal, Dying Gasp means an unrecoverable condition, and Critical Event means an unspecified vendor-specific critical event. The switch can receive and process but not generate Link Fault or Critical Event OAM PDUs. It can generate Dying Gasp OAM PDUs to show when Ethernet OAM is disabled, the interface is shut down, the interface enters the error-disabled state, or the switch is reloading. It also supports Dying Gasp PDUs based on loss of power.
- Remote loopback mode to ensure link quality with a remote peer during installation or troubleshooting. In this mode, when the switch receives a frame that is not an OAM PDU or a pause frame, it sends it back on the same port. The link appears to the user to be in the up state. You can use the returned loopback acknowledgment to test delay, jitter, and throughput.

The following sections describe how to configure ethernet OAM on the Cisco ME 1200 NID.

Setting the Alarm Indication Signal (AIS)

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.

Step 3	**setAis aisConfig**{**aisAction** {**disable** | **enable** {**framerate** | **protect**} | **mepInstance***mep_instance_number*}
	
	Example:
	`UCS(OperationsMepPortType)# setAis aisConfig aisAction enable frameRate fr1s`
	`UCS(OperationsMepPortType)# setAis aisConfig`
		Enables or disables the alarm indication signal request on a Maintenance End Point (MEP) instance.
		
		• **aisAction—**Enables or disables the AIS.
		• **framerate—**Defines the frame rate, whether frames per minutes, or frames per second.

	Command or Action	Purpose

	`aisAction enable protect disable`
	`UCS(OperationsMepPortType)# setAis aisConfig aisAction disable`
	`UCS(OperationsMepPortType)# setAis aisConfig mepInstance 1`
		• **protect—**Defines whether or not AIS can be used for protection.
		• **mepInstance—**Sets the MEP instance number. The valid values are from 1 to 128.

Step 4	**setAis review** **Example:** `UCS(OperationsMepPortType)# setAis review`	Displays the setAis configuration.
Step 5	**setAis commit** **Example:** `UCS(OperationsMepPortType)# setAis commit`	Sends the setAis configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Setting Delay Measurement

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.

Step 3	**setDm dmConfig** {**dmAction** {**disable** | **enable** {**calculation** | **cast** | **interval** | **lastN** | **mode** | **priority**}} | **mepInstance***mep_instance_number*}
	
	Example:
	`UCS(OperationsMepPortType)# setDM dmConfig dmAction enable calculation rdtrp`
	`UCS(OperationsMepPortType)# setDM dmConfig dmAction enable cast uni mepId 0`
	`UCS(OperationsMepPortType)# setDM dmConfig dmAction enable interval 10`
		Enables or disables the delay measurement request.
		• **dmAction—**Enables or disables the delay measurement.
		• **calculation—**Is the delay calculation.
		• **cast—**Is either unicast or multicast.
		• **interval—**Is the interval between PDU transmission. The valid values are from 10 to 65535.

	Command or Action	Purpose

	`UCS(OperationsMepPortType)# setDM dmConfig dmAction enable lastN 10`
	`UCS(OperationsMepPortType)# setDM dmConfig dmAction enable mode twoWay`
	`UCS(OperationsMepPortType)# setDM dmConfig dmAction enable priority 0`
	`UCS(OperationsMepPortType)# setDM dmConfig dmAction disable`
	`UCS(OperationsMepPortType)# setDM dmConfig mepInstance 1`
		• **lastN—**Are the last N delays used for average last N calculation.
		• **mode—**Is either one-way mode or two-way mode.
		• **priority—**Is the priority in case of tagged OAM. In the EVC domain this is the COS-ID.
		• **mepInstance—**Is the MEP instance number. The valid values are from 1 to 128.

Step 4	**setDm review** **Example:** `UCS(OperationsMepPortType)# setDm review`	Displays the setDm configuration.
Step 5	**setDm commit** **Example:** `UCS(OperationsMepPortType)# setDm commit`	Sends the setDm configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Updating Delay Measurement

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.
Step 3	**updateDM updateDMConfig** {**mepInstance** *mep_instance_id* | **update** {**overflowReset** {**keep** | **reset**} | **synchronized** {**disable** | **enable**} | **txmode** {**proprietary** | **standardize**} | **unit** {**ns** | **us**}	Updates the delay measurement request. • **mepInstance—**Configures the MEP instance number. The valid values are from 1 to 128.

	Command or Action	Purpose
	Example: `UCS(OperationsMepPortType)# updateDM updateDmConfig update overflowReset keep UCS(OperationsMepPortType)# updateDM updateDmConfig update syncronized disable UCS(OperationsMepPortType)# updateDM updateDmConfig update txMode standardize UCS(OperationsMepPortType)# updateDM updateDmConfig update unit us`	• **update—**Updates the delay measurement parameters. • **overflowRest—**Configures all Delay Measurement results on total delay counter overflow. • **synchronized—**Synchronizes the near- and far-end in real time. • **txmode—**Configures the transmission mode. • **unit—**Configures the delay in nano seconds or microseconds.
Step 4	**updateDM review** **Example:** `UCS(OperationsMepPortType)# updateDM review`	Displays the updateDM configuration.
Step 5	**updateDM commit** **Example:** `UCS(OperationsMepPortType)# updateDM commit`	Sends the updateDM configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Setting Loss Measurement

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.
Step 3	**setlm lmConfig** {**lmAction** {**disable** | **enable** {**cast** | **flr** | **framerate** | **mode** | **priority**}} | **mepInstance***mep_instance_number*}	Enables or disables the loss measurement request. • **lmAction—**Enables or disables the loss measurement.

	Command or Action	Purpose

	Example:
	`UCS(OperationsMepPortType)# setLM lmConfig lmAction enable cast uni`
	`UCS(OperationsMepPortType)# setLM lmConfig lmAction enable flr 5`
	`UCS(OperationsMepPortType)# setLM lmConfig lmAction enable frameRate fr1s`
	`UCS(OperationsMepPortType)# setLM lmConfig lmAction enable mode single`
	`UCS(OperationsMepPortType)# setLM lmConfig lmAction enable priority 0`
	`UCS(OperationsMepPortType)# setLM lmConfig lmAction disable`
	`UCS(OperationsMepPortType)# setLM lmConfig mepInstance 1`
		• **cast—**Defines whether OAM PDU is transmitted with either unicast MAC or multicast MAC.
		• **flr—**Is the frame loss ratio. The valid values for frame loss interval ratio is from 0 to 99.
		• **framerate—**Defines the frame rate, whether 1 or 10 frames per second, 1 or 6 frames per minutes, or 6 frames per hour.
		• **mode—**Is either single mode or dual mode.
		• **priority—**Is the priority in case of tagged OAM. In the EVC domain this is the COS-ID. The valid values are from 0 to 7.
		• **mepInstance—**Is the MEP instance number. The valid values are from 1 to 128.

Step 4	**setLM review** **Example:** `UCS(OperationsMepPortType)# setLM review`	Displays the setLM configuration.
Step 5	**setLM commit** **Example:** `UCS(OperationsMepPortType)# setLM commit`	Sends the setLM configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Setting Lock Signal

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:**	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.

	Command or Action	Purpose
	UCS# OperationsMepPortType	

Step 3	**setlck lckConfig**{**lckAction** {**disable** | **enable framerate** | **mepInstance***mep_instance_number*}
	
	Example:
	`UCS(OperationsMepPortType)# setLck lckConfig lckAction enable frameRate fr1s`
	`UCS(OperationsMepPortType)# setLck lckConfig lckAction disable`
	`UCS(OperationsMepPortType)# setLck lckConfig mepInstance 1`
		Enables or disables the lock signal request.
		• **lckAction—**Enables or disables the lock signal request.
		• **framerate—**Defines the frame rate, whether frames per minutes, or frames per second.
		• **mepInstance—**Is the MEP instance number. The valid values are from 1 to 128.

Step 4	**setlck review** **Example:** `UCS(OperationsMepPortType)# setlck review`	Displays the setlck configuration.
Step 5	**setlck commit** **Example:** `UCS(OperationsMepPortType)# setlck commit`	Sends the setlck configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits from the OperationsMepPortType mode.

Setting Link Trace

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.
Step 3	**setlinkTrace linkTrace** {**ltAction** {**disable** | **enable** {**destination** | **priority** | **ttl**} | **mepInstance***mep_instance_number*}	Enables or disables the link trace request. • **ltAction—**Enables or disables the link trace.

	Command or Action	Purpose

	Example:
	`UCS(OperationsMepPortType)# setLinkTrace linkTrace ltAction enable destination mepId 0`
	`UCS(OperationsMepPortType)# setLinkTrace linkTrace ltAction enable priority 0`
	`UCS(OperationsMepPortType)# setLinkTrace linkTrace ltAction enable ttl 1`
	`UCS(OperationsMepPortType)# setLinkTrace linkTrace ltAction disable`
	`UCS(OperationsMepPortType)# setLinkTrace linkTrace mepInstance 1`
		• **enable destination—**Enables the target peer MEP.
		• **priority—**Is the priority in case of tagged OAM. In the EVC domain, this value is the COS-ID. The valid values are from 0 to 7.
		• **ttl—**Is the time-to-live value. The valid values are from 1 to 999.
		• **mepInstance—**Is the MEP instance number. The valid values are from 1 to 128.

Step 4	**setLinkTrace review** **Example:** `UCS(OperationsMepPortType)# setLinkTrace review`	Displays the setLinkTrace configuration.
Step 5	**setLinkTrace commit** **Example:** `UCS(OperationsMepPortType)# setLinkTrace commit`	Sends the setLinkTrace configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Setting Loopback

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.
Step 3	**setloopBack loopBackConfig** {**lbAction** {**disable** | **enable** {**cast** | **count** | **dei** | **interval** | **priority** | **size**}} | **mepInstance***mep_instance_number*}	Enables or disables the loopback request. • **lbAction—**Enables or disables loopback.

	Command or Action	Purpose

	Example:
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig lbAction enable cast uni mepId 0`
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig lbAction enable count 5`
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig lbAction enable dei disable`
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig lbAction enable interval 10`
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig lbAction enable priority 0`
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig lbAction enable size 100`
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig lbAction disable`
	`UCS(OperationsMepPortType)# setLoopBack loopBackConfig mepInstance 1`
		• **cast—**Is either unicast or multicast.
		• **count—**Is the number of loopback message (LBM) PDUs to send in one loop test.
		• **dei—**Is the Drop Eligible Indicator in case of tagged OAM.
		• **interval—**Is the interval between transmitting LBM protocol data unit (PDU). The valid values are from 1 to 100.
		• **priority—**Is the priority in case of tagged OAM. In the EVC domain this is the COS-ID.
		• **size—**Is the number of bytes in the LBM PDU Data Pattern TLV. The valid values are from 1 to 1400.
		• **mepInstance—**Is the MEP instance number. The valid values are from 1 to 128.

Step 4	**setloopBack review** **Example:** `UCS(OperationsMepPortType)# setloopBack review`	Displays the setloopBack configuration.
Step 5	**setloopBack commit** **Example:** `UCS(OperationsMepPortType)# setloopBack commit`	Sends the setloopBack configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits from the OperationsMepPortType mode.

Setting Test Signal

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.

Step 3	**settst tstConfig tstConfig** {**dei** {**disable** | **enable**} | **mepId** | **mepInstance***mep_instance_id* | **pattern** | **priority** | **rate** | **sequence size**}
	
	Example:
	`UCS(OperationsMepPortType)# setTst tstConfig dei disable`
	`UCS(OperationsMepPortType)# setTst tstConfig mepId 0`
	`UCS(OperationsMepPortType)# setTst tstConfig mepInstance 1`
	`UCS(OperationsMepPortType)# setTst tstConfig pattern allZero`
	`UCS(OperationsMepPortType)# setTst tstConfig priority 0`
	`UCS(OperationsMepPortType)# setTst tstConfig rate 1`
	`UCS(OperationsMepPortType)# setTst tstConfig sequence disable`
	`UCS(OperationsMepPortType)# setTst tstConfig size 64`
		Enables or disables the test signal request.
		• **tstConfig—**Enables or disables the test signal request.
		• **dei—**Defines the Drop Eligible Indicator in case of tagged OAM.
		• **mepId—**Defines peer MEP ID. The valid values are from 0 to 8191.
		• **mepInstance—**Is the MEP instance number. The valid values are from 1 to 128.
		• **pattern—**Enables the sequence number in test PDU.
		• **priority—**Is the priority in case of tagged OAM. In the EVC domain this is the COS-ID.
		• **rate—**Is the test frame transmission bit rate – in Mega bits per second. The valid values are from 1 to 400.
		• **sequence—** Enables and disables sequence number in test PDUs
		• **size—**Is the test frame size. The valid values are from 1 to 1581.

Step 4	**setTst review** **Example:** `UCS(OperationsMepPortType)# setTst review`	Displays the setTst configuration.
Step 5	**setTst commit** **Example:** `UCS(OperationsMepPortType)# setTst commit`	Sends the setTst configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Updating Test Signal

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**OperationsMepPortType** **Example:** UCS# OperationsMepPortType	Enters the OperationsMepPortType mode and enables fault management and performance monitoring on the MEP.

Step 3	**updateTst updateTstConfig** {**mepInstance***mep_instance_id* | **update** {**Rx** {**disable** | **enable**} | **Tx** {**disable** | **enable**}}
	
	Example:
	`UCS(OperationsMepPortType)# updateTst updateTstConfig update Rx enable`
	`UCS(OperationsMepPortType)# updateTst updateTstConfig update Tx enable`
	`UCS(OperationsMepPortType)# updateTst updateTstConfig mepInstance 1`
		Updates the test signal request.
		• **updateTstConfig—**Updates the test signal parameters.
		• **mepInstance—**Is the MEP instance number. The valid values are from 1 to 128.
		• **update—**Enables or disables the receive and transmit test signals.

Step 4	**updateTst review** **Example:** `UCS(OperationsMepPortType)# updateTst review`	Displays the updateTst configuration.
Step 5	**updateTst commit** **Example:** `UCS(OperationsMepPortType)# updateTst commit`	Sends the updateTst configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Understanding Link OAM

The following sections describe how to configure Link OAM on the Cisco ME 1200 NID.

Setting OAM Port Operations

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**LinkOamPortType** **Example:** UCS# LinkOamPortType	Enters the LinkOamPortType mode.

Step 3	**setLinkOamPortConfig portConfig** {**linkOam enable** | **linkmonitorSupport enable** | **loopbackOperation enable** | **loopbackSupport enable** | **mibretrivalSupport enable** | **oamMode** {**active** | **passive**} | **portNumber***port_number* | **variableRetrieve** {**localInfo** | **remoteInfo**}
	
	Example:
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig portNumber 5`
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig linkOam enable`
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig linkmonitorSupport enable`
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig loopbackOperation enable`
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig loopbackSupport enable`
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig mibretrivalSupport enable`
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig oamMode active`
	`UCS(LinkOamPortType)# setLinkOamPortConfig portConfig variableRetrieve localInfo`
		Sets the Link OAM port configuration.
		• **linkOam—**Sets the supported Link OAM.
		• **linkmonitorSupport—**Enables or disables the Link monitor support.
		• **loopbackOperation—**Sets the loopback operation.
		• **loopbackSupport—**Sets the Link OAM remote loopback support.
		• **mibretrivalSupport—**Set MIB retrieval support.
		• **oamMode—**Sets the Link OAM mode to Active or Passive.
		• **portNumber—**Sets the interface number. The valid values are from 1 to 6.
		• **variableRetrieve—**Sets the MIB variable retrieve value to local information or remote information.

Step 4	**setLinkOamPortConfig review** **Example:** `UCS(LinkOamPortType)# setLinkOamPortConfig review`	Displays the LinkOamPortType configuration.
Step 5	**setLinkOamPortConfig commit** **Example:** `UCS(LinkOamPortType)# setLinkOamPortConfig commit`	Sends the LinkOamPortType configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(LinkOamPortType)# exit`	Exits the LinkOamPortType mode.

What to Do Next

After the configuration is sent to the Cisco ME 1200 NID, use the following **get** command to view the LinkOamPortType configuration.

```
UCS(LinkOamPortType)# getLinkOamPortConfig linkOamRequest portNumber 5
UCS(LinkOamPortType)# getLinkOamPortConfig review
UCS(LinkOamPortType)# getLinkOamPortConfig commit
```

Setting Link OAM Event Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**LinkOamPortType** **Example:** UCS# LinkOamPortType	Enters the LinkOamPortType mode.

Step 3	**setlinkeventConfig linkEventConfig** {**errorFrame** {**threshold** | **window**} | **frameSeconds** {**threshold** | **window**} | **portNumber***interface_number* | **symbolPeriod** {**threshold** | **window**}}
	
	Example:
	`UCS(LinkOamPortType)# setLinkEventConfig linkEventConfig portNumber 5`
	`UCS(LinkOamPortType)# setLinkEventConfig linkEventConfig errorFrame threshold 0`
	`UCS(LinkOamPortType)# setLinkEventConfig linkEventConfig errorFrame window 1`
	`UCS(LinkOamPortType)# setLinkEventConfig linkEventConfig frameSeconds threshold 0`
	`UCS(LinkOamPortType)# setLinkEventConfig linkEventConfig frameSeconds window 10`
	`UCS(LinkOamPortType)# setLinkEventConfig linkEventConfig symbolPeriod threshold 0`
	`UCS(LinkOamPortType)# setLinkEventConfig linkEventConfig symbolPeriod window 1`
		Sets the Link Event configuration request.
		
		• **errorFrame—**Configures the frame error event thresholds and window for error frames that trigger an error-frame link event.The valid threshold values are from 0 to 4294967295 number of frames. The valid window vales to count the number of error frames is from 1 to 60 seconds.
		
		• **frameSeconds—**Configures the frame seconds summary. The valid threshold values are from 0 to 65535 number of permissible error frames. The valid window vales for monitoring the frames is from 10 to 900 seconds.
		
		• **portNumber—**Is the port number for the Link Event configuration request. The valid values are from 1 to 6.
		
		• **symbolPeriod—**Configures the window and thresholds for an error-symbol period that triggers an error-symbol period link event. The valid threshold values are from 0 to 4294967295 number of permissible error symbols. The valid window vales for monitoring the frames is from 1 to 60 seconds.

	Command or Action	Purpose
Step 4	**setLinkEventConfig review** **Example:** `UCS(LinkOamPortType)# setLinkEventConfig review`	Displays the setLinkEventConfig configuration.
Step 5	**setLinkEventConfig commit** **Example:** `UCS(LinkOamPortType)# setLinkEventConfig commit`	Sends the setLinkEventConfig configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(LinkOamPortType)# exit`	Exits the LinkOamPortType mode.

What to Do Next

After the configuration is sent to the Cisco ME 1200 NID, use the following **get** command to view the setLinkEventConfig configuration.

```
UCS(LinkOamPortType)# getLinkEventConfig linkOamRequest portNumber 5
UCS(LinkOamPortType)# getLinkEventConfig review
UCS(LinkOamPortType)# getLinkEventConfig commit
```

Setting Remote Loopback Start And Stop

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**LinkOamPortType** **Example:** UCS# LinkOamPortType	Enters the LinkOamPortType mode.
Step 3	**setremoteLoopBack remoteLoopBak** {**start portList** | **stop portList**} **Example:** `UCS(LinkOamPortType)# setRemoteLoopBack remoteLoopBack start portList 1`	Sets the remote loopback request. • **start**—Starts the remote loopback on the defined port list. • **stop**—Stops the remote loopback on the defined port list.

	Command or Action	Purpose

Step 4	**setRemoteLoopBack review**
	
	Example:
	`UCS(LinkOamPortType)# setRemoteLoopBack review`
	`Commands in queue:`
	`        setRemoteLoopBack remoteLoopBack start portList 1`
	`        setRemoteLoopBack remoteLoopBack stop portList 1`
		Displays the setRemoteLoopBack configuration.

Step 5	**setRemoteLoopBack commit** **Example:** `UCS(LinkOamPortType)# setRemoteLoopBack commit`	Sends the setRemoteLoopBack configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(LinkOamPortType)# exit`	Exits the LinkOamPortType mode.

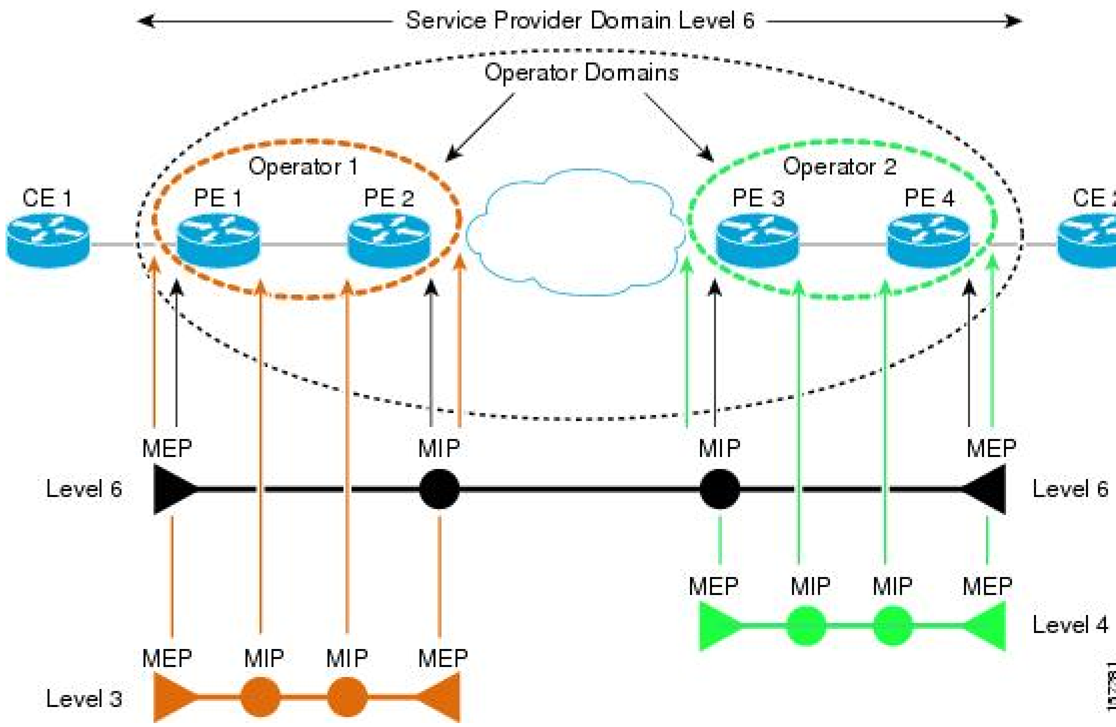
Understanding Connectivity Fault Management

Ethernet CFM is an end-to-end per VLAN Ethernet layer OAM protocol that includes proactive connectivity monitoring, fault verification, and fault isolation. End-to-end can be provider-edge-to-provider-edge (PE-to-PE) device or customer-edge-to-customer-edge (CE-to-CE) device. Ethernet CFM, as specified by IEEE 802.1ag, is the standard for Layer 2 ping, Layer 2 traceroute, and end-to-end connectivity check of the Ethernet network.

CFM Domain

A CFM maintenance domain is a management space on a network that is owned and operated by a single entity and defined by a set of ports internal to it, but at its boundary. You assign a unique maintenance level (from 0 to 7) to define the hierarchical relationship between domains. The larger the domain, the higher the level. For example, as shown in the figure below, a service-provider domain would be larger than an operator domain and might have a maintenance level of 6, while the operator domain maintenance level is 3 or 4.

Figure 9: CFM Maintenance Domains



Maintenance Associations and Maintenance Points

A maintenance association (MA) identifies a service that can be uniquely identified within the maintenance domain. The CFM protocol runs within a maintenance association. A maintenance point is a demarcation point on an interface that participates in CFM within a maintenance domain. Maintenance points drop all lower-level frames and forward all higher-level frames. There are two types of maintenance points:

- Maintenance end points (MEPs) are points at the edge of the domain that define the boundaries and confine CFM messages within these boundaries. Outward facing or Down MEPs communicate through the wire side (connected to the port). Inward facing or Up MEPs communicate through the relay function side, not the wire side.
- Maintenance intermediate points (MIPs) are internal to a domain, not at the boundary, and respond to CFM only when triggered by traceroute and loopback messages. They forward CFM frames received from MEPs and other MIPs, drop all CFM frames at a lower level (unless MIP filtering is enabled), and forward all CFM frames at a higher level and at a lower level and regardless of whether they are received from the relay or wire side. When MIP filtering is enabled, the MIP drops CFM frames at a lower level. MIPs also catalog and forward continuity check messages (CCMs), but do not respond to them.

The following sections describe how to configure CFM on the Cisco ME 1200 NID.

Adding Continuity Check and Automatic Protection Switching

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** UCS# ProvisionMepPortType	Enters the ProvisionMepPortType mode and enables provisioning of the MEP.

Step 3	**addccAps mepFunctionalConfig** {**aps** {**disable** | **enable** {**mode** {**multi** |**uni**} | **priority** | **switchingProtocol** {**laps** | **raps**}} | **cc** {**disable** | **enable** {**framerate** | **priority**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 1`
		Adds the CC or APS configuration request.
		• **mepFunctionalConfig—**Adds the Continuity Check (CC) or automatic protection switching (APS) configuration request.
		• **aps—**Enables or disables the APS parameters.
		• **mode—**Defines whether multicast or unicast.
		• **priority—**Defines the priority in case of tagged OAM. In the EVC domain, this parameter is the COS-ID. The valid values are from 0 to 7.
		• **switchingProtocol—**Sets the appropriate APS switching protocol—Linear Automatic Protection Switching protocol (LAPS) or Ring Automatic Protection Switching protocol (RAPS).
		• **cc—**Enables or disables the CC parameters.
		• **framerate—**Sets the CC frame rate.

Step 4	**addCcAps review** **Example:** `UCS(ProvisionMepPortType)# addCcAps review`	Displays the addCcAps configuration.
Step 5	**addCcAps commit** **Example:** `UCS(ProvisionMepPortType)# addCcAps commit`	Sends the addCcAps configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Adding Peer MEP IDs

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** UCS# ProvisionMepPortType	Enters the ProvisionMepPortType mode and enables provisioning of the MEP.

Step 3	**addPeerMepId mepClientConfig** {**macAddress** *mac_address* | **mepInstance***mep_instance* | **peerMepId** peer_mep_id}
	
	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId mepClientConfig aisPriority aisHighest`
	`UCS(ProvisionMepPortType)# addPeerMepId mepClientConfig domain VLAN`
	`UCS(ProvisionMepPortType)# addPeerMepId mepClientConfig flowId 21`
	`UCS(ProvisionMepPortType)# addPeerMepId mepClientConfig mepInstance 1`
		Adds the client configuration request.
		• **mepClientConfig—**Adds the client configuration request.
		• **macAddress—**The peer MAC address. This MAC address will be overwritten by any learned MAC address through CCM reception.
		• **mepInstance—**Sets the MEP instance number. The valid values are from 1 to 128.
		• **peerMepId—**Sets the peer MEP ID. The valid values are from 1 to 8191.

Step 4	**addPeerMepId review** **Example:** `UCS(ProvisionMepPortType)# addPeerMepId review`	Displays the addPeerMepId configuration.
Step 5	**addPeerMepId commit** **Example:** `UCS(ProvisionMepPortType)# addPeerMepId commit`	Sends the addPeerMepId configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Adding Client Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** UCS# ProvisionMepPortType	Enters the ProvisionMepPortType mode and enables provisioning of the MEP.

Step 3	**addClient mepClientConfig** {**aisPriority** {**aisHighest** | **priority**} | **domain** {**evc** |**vlan**} | **flowID** *flow_id* | **lckPriority** {**lckHighest** | **priority**} | **level***meg_level* | **mepInstance***mep_instance*}
	
	Example:
	`UCS(ProvisionMepPortType)# addClient mepClientConfig aisPriority aisHighest`
	`UCS(ProvisionMepPortType)# addClient mepClientConfig domain VLAN`
	`UCS(ProvisionMepPortType)# addClient mepClientConfig flowId 21`
	`UCS(ProvisionMepPortType)# addClient mepClientConfig mepInstance 1`
		Adds the client configuration request.
		• **mepClientConfig—**Adds the client configuration request.
		• **aisPriority—**Sets the alarm indication signal priority. The AIS priority can be set to either the highest priority or any other priority between 0 and 7.
		• **domain—**Sets the domain**—**whether EVC or VLAN.
		• **flowID—**Sets the ID of the flow. MEP is related to this flow.
		• **lckPriority—**Sets the lock priority. The lock priority can be set to either the highest priority or any other priority between 0 and 7.
		• **level—**Sets the MEG level of the MEP. The valid values are from 0 to 7.
		• **mepInstance—**Sets the MEP instance number. The valid values are from 1 to 128.

Step 4	**addClient review** **Example:** `UCS(ProvisionMepPortType)# addClient review`	Displays the addClient configuration.
Step 5	**addClient commit** **Example:** `UCS(ProvisionMepPortType)# addClient commit`	Sends the addClient configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Creating MEP Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** UCS# ProvisionMepPortType	Enters the ProvisionMepPortType mode and enables provisioning of the MEP.

Step 3	**createMep createMepConfig** {**direction** {**DOWN** | **UP**} | **domain** {**EVC** | **PORT** | **VLAN**} | **flowId***flow_id* | **level***meg_level* | **megDomain** {**maName** | **megIdFormat** {**ieee** | **ituCcMeg** | **ituMeg**}} | **mepId***mep_id* | **mepInstance***mep_instance* | **mode** {**mep** | **mip**} | **residencePort** *port* | **vid***vid_number* | **voe** {**disable** | **enable**}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction UP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain VLAN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig flowId 21`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 1`
	`UCS(ProvisionMepPortType)# createMep createMepConfig voe disable`
		Creates the Maintenance End Point configuration.
		• **createMepConfig—**Creates the MEP configuration.
		• **direction—**Sets the direction of the MEP—whether down (Down MEP) or up (Up MEP).
		• **domain—**Sets the domain—whether EVC, Port, or VLAN.
		• **flowID—**Sets the ID of the flow. MEP is related to this flow.
		• **level—**Sets the MEG level of the MEP. The valid values are from 0 to 7.
		• **megDomain—**Sets the maintenance domain configuration to either **maName** (ITU/IEEE MEG-ID) or **megIdFormat**.
		• **mepId—**Sets the MEP ID. The valid values are from 0 to 8191.
		• **mepInstance—**Sets the MEP instance number. The valid values are from 1 to 128.
		• **mode—**Sets the mode of the MEP instance—whether Maintenance Entity End Point (MEP) or Maintenance Entity Intermediate Point (MIP).
		• **residencePort—**Defines the port that MEP is monitoring. The valid values are from 1 to 6.
		• **vid—**The valid values are from 0 to 4094.
		Note If the MEP is a port Up-MEP or an EVC customer MIP, the VID must be provided.
		• **voe—**Enables or disables the MEP VOE.

	Command or Action	Purpose
Step 4	**createMep review** **Example:** `UCS(ProvisionMepPortType)# createMep review`	Displays the createMep configuration.
Step 5	**createMep commit** **Example:** `UCS(ProvisionMepPortType)# createMep commit`	Sends the createMep configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Updating MEP Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** UCS# ProvisionMepPortType	Enters the ProvisionMepPortType mode and enables provisioning of the MEP.

Step 3	**updateMep mepUpdateConfig** {**mepInstance** *mep_instance* | **update** {**level***meg_level* | **megDomain** {**maName** | **megIdFormat** {**ieee** | **ituCcMeg** | **ituMeg**}} | **mepId***mep_id* | **performanceMonitoring** {**disable** | **enable**} | **vid***vid_number* | **voe** {**disable** | **enable**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction UP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain VLAN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig flowId 21`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep`
		Updates the Maintenance End Point configuration.
		• **mepInstance—**Sets the MEP instance number. The valid values are from 1 to 128.
		• **update—**Updates the MEP configuration.
		• **level—**Sets the MEG level of the MEP. The valid values are from 0 to 7.
		• **megDomain—**Sets the maintenance domain configuration to either **maName** (ITU/IEEE MEG-ID) or **megIdFormat**.
		• **mepId—**Sets the MEP ID. The valid values are from 0 to 8191.

	Command or Action	Purpose
	`createMepConfig residencePort 1` `UCS(ProvisionMepPortType)# createMep createMepConfig voe disable`	• **performanceMonitoring—**Enables or disables performance monitoring • **vid—**The valid values are from 0 to 4094. **Note** If the MEP is a port Up-MEP or an EVC customer MIP, the VID must be provided. • **voe—**Enables or disables the MEP VOE.
Step 4	**updateMep review** **Example:** `UCS(ProvisionMepPortType)# updateMep review`	Displays the updateMep configuration.
Step 5	**updateMep commit** **Example:** `UCS(ProvisionMepPortType)# updateMep commit`	Sends the updateMep configuration to the Cisco ME 1200 NID.
Step 6	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Configuration Example: Loopback

Consider the following topology:

(Gi1/5)NID-3(Gi1/3)====(Gi1/3)NID-4(Gi1/6)

Configuration on Cisco ME 1200 NID-3

```
ProvisionPortVlanPortType
        createVlanCommand createVlanReq vlan_list 2000
createVlanCommand commit
        modifySwPort modifySWPortConfig interaface 3
        modifySwPort modifySWPortConfig mode trunk native vlan 1
        modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2000
modifySwPort commit
        modifySwPort modifySWPortConfig interaface 4
        modifySwPort modifySWPortConfig mode trunk native vlan 1
        modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2000
modifySwPort commit
exit
ProvisionMepPortType
        createMep createMepConfig mepInstance 100
        createMep createMepConfig direction DOWN
        createMep createMepConfig domain vlan
        createMep createMepConfig level 0
        createMep createMepConfig megDomain maName ERPS-128
        createMep createMepConfig megDomain megIdFormat ituMeg
        createMep createMepConfig mepId 100
        createMep createMepConfig mode MEP
        createMep createMepConfig residencePort 3
```

```
        createMep createMepConfig flow 2000
 createMep commit
       addPeerMepId peerMepConfig mepInstance 100
        addPeerMepId peerMepConfig peerMepId 101
addPeerMepId commit
       addCcAps mepFunctionalConfig mepInstance 100
        addCcAps mepFunctionalConfig cc enable priority 7
        addCcAps mepFunctionalConfig cc enable frameRate fr1s
        addCcAps mepFunctionalConfig aps enable mode multi
        addCcAps mepFunctionalConfig aps enable priority 7
        addCcAps mepFunctionalConfig aps enable switchingProtocol raps octet 1
addCcAps commit
exit

        setLoopBack loopBackConfig mepInstance 100
        setLoopBack loopBackConfig lbAction enable cast multi
        setLoopBack loopBackConfig lbAction enable count 10
        setLoopBack loopBackConfig lbAction enable dei disable
        setLoopBack loopBackConfig lbAction enable interval 1
        setLoopBack loopBackConfig lbAction enable priority 7
        setLoopBack loopBackConfig lbAction enable size 70
        setloopBack commit
```

Configuration on the Cisco ME 1200 NID-4

```
ProvisionPortVlanPortType
        createVlanCommand createVlanReq vlan_list 2000
createVlanCommand commit
        modifySwPort modifySWPortConfig interaface 3
        modifySwPort modifySWPortConfig mode trunk native vlan 1
        modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2000
modifySwPort commit
        modifySwPort modifySWPortConfig interaface 5
        modifySwPort modifySWPortConfig mode trunk native vlan 1
        modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2000
modifySwPort commit
exit
ProvisionMepPortType
        createMep createMepConfig mepInstance 100
        createMep createMepConfig direction DOWN
        createMep createMepConfig domain vlan
        createMep createMepConfig level 0
        createMep createMepConfig megDomain maName ERPS-128
        createMep createMepConfig megDomain megIdFormat ituMeg
        createMep createMepConfig mepId 101
        createMep createMepConfig mode MEP
        createMep createMepConfig residencePort 3
        createMep createMepConfig flow 2000
 createMep commit
       addPeerMepId peerMepConfig mepInstance 100
        addPeerMepId peerMepConfig peerMepId 100
addPeerMepId commit
        addCcAps mepFunctionalConfig mepInstance 100
        addCcAps mepFunctionalConfig cc enable priority 7
        addCcAps mepFunctionalConfig cc enable frameRate fr1s
        addCcAps mepFunctionalConfig aps enable mode multi
        addCcAps mepFunctionalConfig aps enable priority 7
        addCcAps mepFunctionalConfig aps enable switchingProtocol raps octet 1
addCcAps commit
exit
```

Loopback in Cisco ME 1200 NID-3

```
showloopBack mepRequest mepInstance 100
showloopBack commit
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.mepInstance = 100
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.dei.t = 2
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.dei.u.disable = 'DEI Disable'
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.priority = 7
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.cast.t = 2
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.cast.u.multi = 'MULTI'
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.count = 10
```

```
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.size = 70
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.interval = 1
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.mepInstance = 32
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.transactionId = 11
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.txLBM.upper = 0
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.txLBM.lower = 10
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].rcvMac = '00-3A-99-FD-47-2F'
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].received.upper = 0
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].received.lower = 10
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].outOfOrder.upper = 0
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].outOfOrder.lower = 0

 showLoopBack Commit Success!!!
```

Configuration Example: Loss Measurement–Single Ended

Consider the following topology:

TG1====(Gi1/5)NID-3(Gi1/4)====(Gi0/1)UPE NID Controller(Gi0/20)====(Gi1/5)NID-4(Gi1/6)====TG2

Cast: Multi

Ended: Single

Configuration on Cisco ME 1200 NID-3

```
ProvisionEVC
        addEVC evcConfiguration instance 1024
        addEVC evcConfiguration internal_vid 1024
        addEVC evcConfiguration learning enable
        addEVC evcConfiguration nni_ports GigabitEthernet_4_NNI enable
        addEVC evcConfiguration nni_vid 1024
addEVC commit
exit
ProvisionEVC
 addECE ece_configuration ece_id 1024
        addECE ece_configuration control ingress_match uni_ports GigabitEthernet_5_UNI
enable
        addECE ece_configuration control ingress_match outer_tag_match match_type tagged
        addECE ece_configuration control ingress_match outer_tag_match match_fields
vlan_id_filter specific 1024
        addECE ece_configuration control egress_outer_tag mode enabled
        addECE ece_configuration control egress_outer_tag pcp_mode fixed
        addECE ece_configuration control egress_outer_tag pcp_value 7
        addECE ece_configuration control actions class specific 7
        addECE ece_configuration control actions evc_id specific 1024
addECe commit
exit
ProvisionMepPortType
        createMep createMepConfig mepInstance 98
        createMep createMepConfig direction DOWN
        createMep createMepConfig domain EVC
        createMep createMepConfig flowId 1024
        createMep createMepConfig level 0
        createMep createMepConfig megDomain maName LM-Check
        createMep createMepConfig megDomain megIdFormat ituMeg
        createMep createMepConfig mepId  105
        createMep createMepConfig mode MEP
        createMep createMepConfig residencePort 4
createMep createMepConfig voe enable
createMep commit
       addPeerMepId peerMepConfig mepInstance 98
        addPeerMepId peerMepConfig peerMepId 106
addPeerMepId commit
        addCcAps mepFunctionalConfig mepInstance 98
        addCcAps mepFunctionalConfig cc enable priority 7
        addCcAps mepFunctionalConfig cc enable frameRate fr1s
```

```
addCcAps commit
exit
opearationsMepPortType
     setLM lmConfig mepInstance 98
        setLM lmConfig lmAction enable cast multi
        setLM lmConfig lmAction enable frameRate fr1s
        setLM lmConfig lmAction enable mode single
        setLM lmConfig lmAction enable priority 7
setLM commit
exit
```

Configuration on Cisco ME 1200 NID-4

```
ProvisionEVC
        addEVC evcConfiguration instance 1024
        addEVC evcConfiguration internal_vid 1024
        addEVC evcConfiguration learning enable
        addEVC evcConfiguration nni_ports GigabitEthernet_5_NNI enable
        addEVC evcConfiguration nni_vid 1024
addEVC commit
exit
ProvisionEVC
 addECE ece_configuration ece_id 1024
        addECE ece_configuration control ingress_match uni_ports GigabitEthernet_6_UNI
enable
        addECE ece_configuration control ingress_match outer_tag_match match_type tagged
        addECE ece_configuration control ingress_match outer_tag_match match_fields
vlan_id_filter specific 1024
        addECE ece_configuration control egress_outer_tag mode enabled
        addECE ece_configuration control egress_outer_tag pcp_mode fixed
        addECE ece_configuration control egress_outer_tag pcp_value 7
        addECE ece_configuration control actions class specific 7
        addECE ece_configuration control actions evc_id specific 1024
addECe commit
exit
ProvisionMepPortType
        createMep createMepConfig mepInstance 98
        createMep createMepConfig direction DOWN
        createMep createMepConfig domain EVC
        createMep createMepConfig flowId 1024
        createMep createMepConfig level 0
        createMep createMepConfig megDomain maName LM-Check
        createMep createMepConfig megDomain megIdFormat ituMeg
        createMep createMepConfig mepId  106
        createMep createMepConfig mode MEP
        createMep createMepConfig residencePort 5
createMep createMepConfig voe enable
createMep commit
       addPeerMepId peerMepConfig mepInstance 98
        addPeerMepId peerMepConfig peerMepId 105
addPeerMepId commit
        addCcAps mepFunctionalConfig mepInstance 98
        addCcAps mepFunctionalConfig cc enable priority 7
        addCcAps mepFunctionalConfig cc enable frameRate fr1s
addCcAps commit
exit
opearationsMepPortType
     setLM lmConfig mepInstance 98
        setLM lmConfig lmAction enable cast multi
        setLM lmConfig lmAction enable frameRate fr1s
        setLM lmConfig lmAction enable mode single
        setLM lmConfig lmAction enable priority 7
setLM commit
exit
```

Configuration on the UPE NID Controller

```
Controller-Switch# show policy-map lm-v1024
  Policy Map lm-v1024
    Class lm-v1024
     police cir 1000000 bc 31250
       conform-action transmit
```

```
       exceed-action drop

Controller-Switch# show class-map lm-v1024
 Class Map match-all lm-v1024 (id 2)
   Match   dscp af12 (12)

Controller-Switch#

!
interface GigabitEthernet0/1
 switchport trunk allowed vlan none
 switchport mode trunk
 !
 service instance 1024 ethernet
  encapsulation dot1q 1024
  bridge-domain 1024
 !

!
interface GigabitEthernet0/20
 switchport trunk allowed vlan none
 switchport mode trunk
 service-policy input lm-v1024
 !
 service instance 1024 ethernet
  encapsulation dot1q 1024
  bridge-domain 1024
 !
```

Send 20Mbps traffic from TG2 on VLAN 1024 with DSCP set to af12

```
Controller-Switch(config-controller-OpearationsMepPortType)# showLM mepRequest mepInstance
 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.single = 'single'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 85
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 85
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 180123
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 0
```

To view loss measurement:

```
Controller-Switch(config)# controller nid 0/2
Controller-Switch(config-controller)# OpearationsMepPortType
Controller-Switch(config-controller-OpearationsMepPortType)# showlm mepRequest mepInstance
 98
Controller-Switch(config-controller-OpearationsMepPortType)# showlm review
Commands in queue:
        showLM mepRequest mepInstance 98
Controller-Switch(config-controller-OpearationsMepPortType)# showlm commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.single = 'single'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 137
```

```
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 137
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 1105217
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 94

 showLM Commit Success!!!
Controller-Switch(config-controller-OpearationsMepPortType)#
```

To stop traffic, do the following:

```
Controller-Switch# show policy-map int gi 0/20
 GigabitEthernet0/20

  Service-policy input: lm-v1024

    Class-map: lm-v1024 (match-all)
      2175126 packets, 287116632 bytes
      5 minute offered rate 5839000 bps, drop rate 5512000 bps
      Match:  dscp af12 (12)
      police:
        cir 1000000 bps, bc 31250 bytes
        conform-action transmit
        exceed-action drop
      conform: 122168 (packets) 16126176 (bytes)
      exceed: 2052958 (packets) 270990456 (bytes)
      conform: 331000 bps, exceed: 5512000 bps
              Input Policer:
                Policer Packets Drop: 2052958
                Policer Bytes Drop: 270990456

    Class-map: class-default (match-any)
      3606 packets, 293801 bytes
      5 minute offered rate 10000 bps, drop rate 0000 bps
      Match: any
```

To view loss measurement:

```
Controller-Switch(config-controller-OpearationsMepPortType)# showlm review
Commands in queue:
        showLM mepRequest mepInstance 98
Controller-Switch(config-controller-OpearationsMepPortType)# showlm commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.single = 'single'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 349
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 349
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 2052958
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 0

 showLM Commit Success!!!

Controller-Switch(config-controller-OpearationsMepPortType)# controller nid 0/2

Controller-Switch(config-controller)#OpearationsMepPortType
Controller-Switch(config-controller-OpearationsMepPortType)# showlm review
Commands in queue:
        showLM mepRequest mepInstance 98
Controller-Switch(config-controller-OpearationsMepPortType)# showlm commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
```

```
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.single = 'single'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 358
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 358
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 2052958
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 0

 showLM Commit Success!!!
Controller-Switch(config-controller-OpearationsMepPortType)#
```

Configuration Example: Loss Measurement–Dual Ended

Consider the following topology:

TG1====(Gi1/5)NID-3(Gi1/4)====(Gi0/1)UPE NID Controller(Gi0/20)====(Gi1/5)NID-4(Gi1/6)====TG2

Cast: Multi

Ended: Dual

Configuration on Cisco ME 1200 NID-3

```
ProvisionEVC
        addEVC evcConfiguration instance 1022
        addEVC evcConfiguration internal_vid 1022
        addEVC evcConfiguration learning enable
        addEVC evcConfiguration nni_ports GigabitEthernet_4_NNI enable
        addEVC evcConfiguration nni_vid 1022
addEVC commit
exit
ProvisionEVC
 addECE ece_configuration ece_id 1022
        addECE ece_configuration control ingress_match uni_ports GigabitEthernet_5_UNI
enable
        addECE ece_configuration control ingress_match outer_tag_match match_type tagged
        addECE ece_configuration control ingress_match outer_tag_match match_fields
vlan_id_filter specific 1022
        addECE ece_configuration control egress_outer_tag mode enabled
        addECE ece_configuration control egress_outer_tag pcp_mode fixed
        addECE ece_configuration control egress_outer_tag pcp_value 7
        addECE ece_configuration control actions class specific 7
        addECE ece_configuration control actions evc_id specific 1022
addECe commit
exit
ProvisionMepPortType
        createMep createMepConfig mepInstance 94
        createMep createMepConfig direction DOWN
        createMep createMepConfig domain EVC
        createMep createMepConfig flowId 1022
        createMep createMepConfig level 0
        createMep createMepConfig megDomain maName LM-Dual
        createMep createMepConfig megDomain megIdFormat ituMeg
        createMep createMepConfig mepId  102
        createMep createMepConfig mode MEP
        createMep createMepConfig residencePort 4
createMep createMepConfig voe enable
createMep commit
       addPeerMepId peerMepConfig mepInstance 94
        addPeerMepId peerMepConfig peerMepId 103
addPeerMepId commit
```

```
        addCcAps mepFunctionalConfig mepInstance 94
        addCcAps mepFunctionalConfig cc enable priority 7
        addCcAps mepFunctionalConfig cc enable frameRate fr1s
addCcAps commit
exit
opearationsMepPortType
     setLM lmConfig mepInstance 94
        setLM lmConfig lmAction enable cast multi
        setLM lmConfig lmAction enable frameRate fr1s
        setLM lmConfig lmAction enable mode dual
        setLM lmConfig lmAction enable priority 7
setLM commit
exit

ProvisionEVC
        addEVC evcConfiguration instance 1021
        addEVC evcConfiguration internal_vid 1021
        addEVC evcConfiguration learning enable
        addEVC evcConfiguration nni_ports GigabitEthernet_4_NNI enable
        addEVC evcConfiguration nni_vid 1021
addEVC commit
exit
```

Configuration on Cisco ME 1200 NID-4

```
ProvisionEVC
        addEVC evcConfiguration instance 1022
        addEVC evcConfiguration internal_vid 1022
        addEVC evcConfiguration learning enable
        addEVC evcConfiguration nni_ports GigabitEthernet_5_NNI enable
        addEVC evcConfiguration nni_vid 1022
addEVC commit
exit
ProvisionEVC
 addECE ece_configuration ece_id 1022
        addECE ece_configuration control ingress_match uni_ports GigabitEthernet_6_UNI
enable
        addECE ece_configuration control ingress_match outer_tag_match match_type tagged
        addECE ece_configuration control ingress_match outer_tag_match match_fields
vlan_id_filter specific 1022
        addECE ece_configuration control egress_outer_tag mode enabled
        addECE ece_configuration control egress_outer_tag pcp_mode fixed
        addECE ece_configuration control egress_outer_tag pcp_value 7
        addECE ece_configuration control actions class specific 7
        addECE ece_configuration control actions evc_id specific 1022
addECe commit
exit
ProvisionMepPortType
        createMep createMepConfig mepInstance 94
        createMep createMepConfig direction DOWN
        createMep createMepConfig domain EVC
        createMep createMepConfig flowId 1022
        createMep createMepConfig level 0
        createMep createMepConfig megDomain maName LM-Dual
        createMep createMepConfig megDomain megIdFormat ituMeg
        createMep createMepConfig mepId  103
        createMep createMepConfig mode MEP
        createMep createMepConfig residencePort 5
createMep createMepConfig voe enable
createMep commit
       addPeerMepId peerMepConfig mepInstance 94
        addPeerMepId peerMepConfig peerMepId 102
addPeerMepId commit
        addCcAps mepFunctionalConfig mepInstance 94
        addCcAps mepFunctionalConfig cc enable priority 7
        addCcAps mepFunctionalConfig cc enable frameRate fr1s
addCcAps commit
exit
opearationsMepPortType
     setLM lmConfig mepInstance 94
        setLM lmConfig lmAction enable cast multi
        setLM lmConfig lmAction enable frameRate fr1s
```

```
        setLM lmConfig lmAction enable mode dual
        setLM lmConfig lmAction enable priority 7
setLM commit
exit
```

Configuration on the UPE NID Controller

```
Controller-Switch# show policy-map lm-v1022
 Policy Map lm-v1022
    Class lm-v1022
     police cir 1000000 bc 31250
       conform-action transmit
       exceed-action drop

Controller-Switch#
!
interface GigabitEthernet0/1
 switchport trunk allowed vlan none
 switchport mode trunk
!
 service instance 1022 ethernet
  encapsulation dot1q 1022
  bridge-domain 1022
 !
!
interface GigabitEthernet0/20
 switchport trunk allowed vlan none
 switchport mode trunk
 service-policy output lm-v1022
 !
 service instance 1022 ethernet
  encapsulation dot1q 1022
  bridge-domain 1022
 !
```

Send 20Mbps traffic from TG2 on VLAN 1022 with DSCP set to af11

```
Controller-Switch(config-controller-ProvisionMepPortType)# controller nid 0/2
Controller-Switch(config-controller)#ProvisionMepPortType
Controller-Switch(config-controller-ProvisionMepPortType)# showMepConfig flush
Controller-Switch(config-controller-ProvisionMepPortType)# showMepalar mepRequest mepInstance
 94
Controller-Switch(config-controller-ProvisionMepPortType)# showMepalar commit
ShowMepAlarms_Output.mepState.mepInst[0].mepInstance = 94
ShowMepAlarms_Output.mepState.mepInst[0].cLevel = false
ShowMepAlarms_Output.mepState.mepInst[0].cMeg = false
ShowMepAlarms_Output.mepState.mepInst[0].cMep = false
ShowMepAlarms_Output.mepState.mepInst[0].cAis = false
ShowMepAlarms_Output.mepState.mepInst[0].cLck = false
ShowMepAlarms_Output.mepState.mepInst[0].cSsf = false
ShowMepAlarms_Output.mepState.mepInst[0].aBlk = false
ShowMepAlarms_Output.mepState.mepInst[0].atsf = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].peerMepId = 102
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cLoc = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cRdi = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cPeriod = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cPrio = false

 showMepAlarms Commit Success!!!
```

To view loss measurement:

```
Controller-Switch(config)# controller nid 0/2
Controller-Switch(config-controller)# OpearationsMepPortType
Controller-Switch(config-controller-OpearationsMepPortType)# showlm mepRequest mepInstance
 98
Controller-Switch(config-controller-OpearationsMepPortType)# showlm review
Commands in queue:
        showLM mepRequest mepInstance 98
Controller-Switch(config-controller-OpearationsMepPortType)# showlm commit
```

```
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.single = 'single'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 137
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 137
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 1105217
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 94

 showLM Commit Success!!!
Controller-Switch(config-controller-OpearationsMepPortType)#
Controller-Switch(config-controller-ProvisionMepPortType)#controller nid 0/1

Controller-Switch(config-controller)#ProvisionMepPortType
Controller-Switch(config-controller-ProvisionMepPortType)#showMepConfig flush
Controller-Switch(config-controller-ProvisionMepPortType)#showMepalar mepRequest mepInstance
 94
Controller-Switch(config-controller-ProvisionMepPortType)#showMepalar commit
ShowMepAlarms_Output.mepState.mepInst[0].mepInstance = 94
ShowMepAlarms_Output.mepState.mepInst[0].cLevel = false
ShowMepAlarms_Output.mepState.mepInst[0].cMeg = false
ShowMepAlarms_Output.mepState.mepInst[0].cMep = false
ShowMepAlarms_Output.mepState.mepInst[0].cAis = false
ShowMepAlarms_Output.mepState.mepInst[0].cLck = false
ShowMepAlarms_Output.mepState.mepInst[0].cSsf = false
ShowMepAlarms_Output.mepState.mepInst[0].aBlk = false
ShowMepAlarms_Output.mepState.mepInst[0].atsf = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].peerMepId = 103
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cLoc = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cRdi = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cPeriod = false
ShowMepAlarms_Output.mepState.mepInst[0].peerMepState[0].cPrio = false

 showMepAlarms Commit Success!!!
Controller-Switch(config-controller-ProvisionMepPortType)#

Controller-Switch(config-controller-ProvisionMepPortType)#controller nid 0/1
Controller-Switch(config-controller)#OpearationsMepPortType
Controller-Switch(config-controller-OpearationsMepPortType)#showlm flush
Controller-Switch(config-controller-OpearationsMepPortType)#showlm mepRequest mepInstance
94
Controller-Switch(config-controller-OpearationsMepPortType)#showlm commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 1
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.dual = 'dual'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 64
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 47
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 1
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 586684
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 94

 showLM Commit Success!!!
Controller-Switch(config-controller-OpearationsMepPortType)#controller nid 0/2
Controller-Switch(config-controller)#OpearationsMepPortType
Controller-Switch(config-controller-OpearationsMepPortType)#showlm flush
Controller-Switch(config-controller-OpearationsMepPortType)#showlm mepRequest mepInstance
94
```

```
Controller-Switch(config-controller-OpearationsMepPortType)#showlm commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 1
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.dual = 'dual'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 70
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 61
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 811684
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 1
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 0

 showLM Commit Success!!!
Controller-Switch(config-controller-OpearationsMepPortType)#
```

To stop traffic, do the following:

```
Service-policy output: lm-v1022

    Class-map: lm-v1022 (match-all)
      3389497 packets, 447413604 bytes
      5 minute offered rate 8626000 bps, drop rate 8126000 bps
      Match:  dscp af11 (10)
      police:
        cir 1000000 bps, bc 31250 bytes
        conform-action transmit
        exceed-action drop
      conform: 196188 (packets) 25112064 (bytes)
      exceed: 3193309 (packets) 408743552 (bytes)
      conform: 492000 bps, exceed: 7880000 bps
          Queue-limit current-queue-depth 0 bytes
              Output Queue:
                Default Queue-limit 49152 bytes
                Tail Packets Drop: 3193309
                Tail Bytes Drop: 421516788

    Class-map: class-default (match-any)
      2491 packets, 170276 bytes
      5 minute offered rate 6000 bps, drop rate 0000 bps
      Match: any
```

To view loss measurement:

```
Controller-Switch(config)# controller nid 0/1
Controller-Switch(config-controller)#
Controller-Switch(config-controller)# OpearationsMepPortType
Controller-Switch(config-controller-OpearationsMepPortType)# showlm review
Commands in queue:
        showLM mepRequest mepInstance 94
Controller-Switch(config-controller-OpearationsMepPortType)# showlm commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 1
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.dual = 'dual'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 262
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 262
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 3193309
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 0
```

```
 showLM Commit Success!!!

Controller-Switch(config)# controller nid 0/2
Controller-Switch(config-controller)# OpearationsMepPortType
Controller-Switch(config-controller-OpearationsMepPortType)# showlm review
Commands in queue:
        showLM mepRequest mepInstance 94
Controller-Switch(config-controller-OpearationsMepPortType)# showlm commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 1
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.dual = 'dual'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 94
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 277
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 276
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 3193309
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 0

 showLM Commit Success!!!
Controller-Switch(config-controller-OpearationsMepPortType)#
```

CHAPTER 18

Configuring Performance Monitoring

Performance Monitor is a carrier Ethernet software feature which provides:

- Monitoring delay measurements
- Monitoring loss measurements
- Monitoring ECE
- Monitoring EVC
- Storing the monitored data
- Transferring the monitored data

For information on configuring EVC, see How to Configure Ethernet Virtual Circuit.

For information on configuring MEP, see Creating MEP Configuration.

Restrictions for Configuring Performance Monitoring

- Performance monitoring is not enabled by default for loss measurement (LM), delay measurement (DM), EVC, and ECE.

ITU-T Y.1731 Performance Monitoring in a Service Provider Network

ITU-T Y.1731 performance monitoring provides standard-based Ethernet performance monitoring that encompasses the measurement of Ethernet frame delay, frame-delay variation, and throughput as outlined in the ITU-T Y.1731 specification and interpreted by the Metro Ethernet Forum (MEF). Service providers offer service level agreements (SLAs) that describe the level of performance customers can expect for services. This document describes the Ethernet performance management aspect of SLAs.

Frame Delay and Frame-Delay Variation

The Frame Delay parameter can be used for on-demand OAM measurements of frame delay and frame-delay variation. When a maintenance end point (MEP) is enabled to generate frames with frame-delay measurement (ETH-DM) information, it periodically sends frames with ETH-DM information to its peer MEP in the same maintenance entity. Peer MEPs perform frame-delay and frame-delay variation measurements through this periodic exchange during the diagnostic interval.

An MEP requires the following specific configuration information to support ETH-DM:

- MEG level—MEG level at which the MEP exists
- Priority
- Drop eligibility—marked drop ineligible
- Transmission rate
- Total interval of ETH-DM
- MEF10 frame-delay variation algorithm

A MEP transmits frames with ETH-DM information using the TxTimeStampf information element. TxTimeStampf is the time stamp for when the ETH-DM frame was sent. A receiving MEP can compare the TxTimeStampf value with the RxTimef value, which is the time the ETH-DM frame was received, and calculate one-way delay using the formula *frame delay = RxTimef – TxTimeStampf*.

One-way frame-delay measurement (1DM) requires that clocks at both the transmitting MEP and the receiving MEPs are synchronized. Measuring frame-delay variation does not require clock synchronization and the variation can be measured using 1DM or a frame-delay measurement message (DMM) and a frame-delay measurement reply (DMR) frame combination.

If it is not practical to have clocks synchronized, only two-way frame-delay measurements can be made. In this case, the MEP transmits a frame containing ETH-DM request information and the TxTimeStampf element, and the receiving MEP responds with a frame containing ETH-DM reply information and the TxTimeStampf value copied from the ETH-DM request information.

Two-way frame delay is calculated as *frame delay = RxTimeb – TxTimeStampf*, where RxTimeb is the time that the frame with ETH-DM reply information was received. Two-way frame delay and variation can be measured using only DMM and DMR frames.

To allow more precise two-way frame-delay measurement, the MEP replying to a frame with ETH-DM request information can also include two additional time stamps in the ETH-DM reply information:

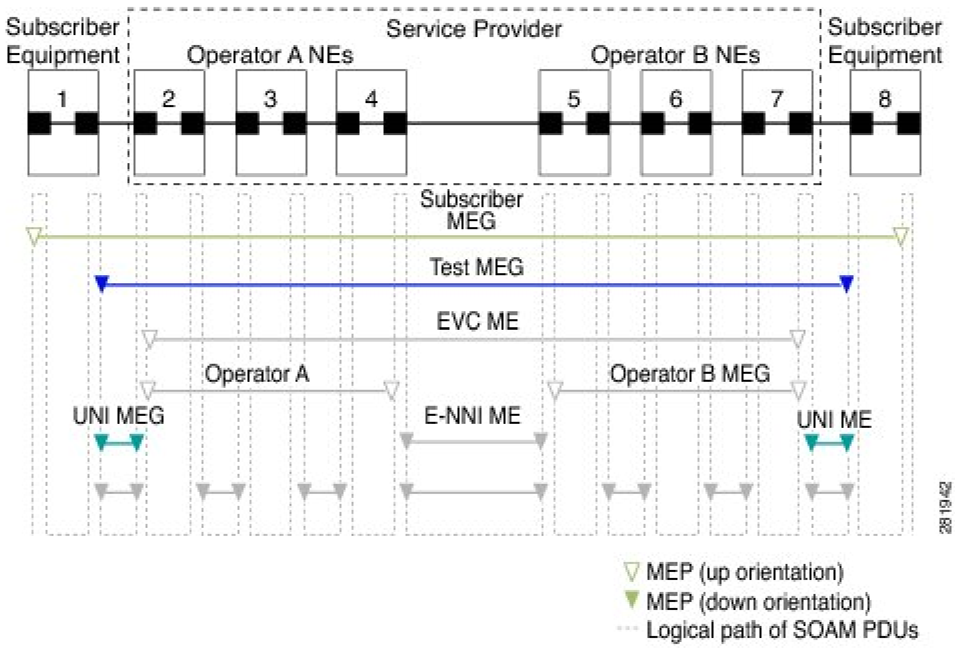
- RxTimeStampf—Time stamp of the time at which the frame with ETH-DM request information was received.
- TxTimeStampb—Time stamp of the time at which the transmitting frame with ETH-DM reply information was sent.

Note The frame-delay and frame-delay variation measurement processes are aborted when faults related to continuity and availability occur or when known network topology changes occur.

An MIP is transparent to the frames with ETH-DM information; therefore, an MIP does not require information to support the ETH-DM function.

The figure below shows a functional overview of a typical network in which Y.1731 performance monitoring is used.

Figure 10: Y.1731 Performance Monitoring



How to Configure Performance Monitoring

Provisioning the UCS Controller to Configure Performance Monitoring

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**OperationsMepPortType** **Example:** `UCS# OperationsMepPortType`	Enters performance monitoring provisioning (PM) mode.

Step 3	**OperationsMepPortType** {**clearMepStats** | **default** | **exit**| **no** | **setAis** | **setDM** | **setLM** | **setLck** | **setLinkTrace** | **setLoopBack** | **setTst** | **showAis** | **showDM** | **showLM** | **showLck** | **showLinkTrace** | **showLoopBack** | **showTst** | **updateDM** | **updateTst**}
	
	Example:
	`UCS(OperationsMepPortType)# ?`
	`OperationsMepPortType sub-mode commands:`
	`clearMepStats  Clear mep statistics request`
	`default        Set a command to its defaults`
	`exit           Exit from OperationsMepPortType sub configuration mode`
	
	`no             Negate a command or set its defaults`
	`setAis         Enable or Disable AIS request`
	`setDM          Enable or Disable delay measurement request`
	`setLM          Enable or Disable loss measurement request`
	`setLck         Enable or Disable lock signal request`
	`setLinkTrace   Enable or Disable linktrace request`
	`setLoopBack    Enable/Disable loopback`
	`setTst         Enable or Disable test signal request`
	`showAis        Show AIS configuration request`
	`showDM         Show delay measurement statistics request`
	`showLM         Show LM statistics request`
	`showLck        Show lock signal current configuration request`
	`showLinkTrace  Show linktrace state and current configuration request`
	
	`showLoopBack   Show loopback state and current configuration request`
	`showTst        Show test signal statistics and current configuration`
	`request`
	`updateDM       Update DM parameters request`
	`updateTst      Update Tst signal request`
		Displays the supported configurations for performance monitoring.

Step 4	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Configuration Example

The following example shows the supported PM configuration:

```
UCS(OperationsMepPortType)# ?
OperationsMepPortType sub-mode commands:
  clearMepStats  Clear mep statistics request
  default        Set a command to its defaults
  exit           Exit from OpearationsMepPortType sub configuration mode
  no             Negate a command or set its defaults
  setAis         Enable or Disable AIS request
  setDM          Enable or Disable delay measurement request
  setLM          Enable or Disable loss measurement request
  setLck         Enable or Disable lock signal request
  setLinkTrace   Enable or Disable linktrace request
  setLoopBack    Enable/Disable loopback
  setTst         Enable or Disable test signal request
  showAis        Show AIS configuration request
  showDM         Show delay measurement statistics request
```

```
showLM         Show LM statistics request
showLck        Show lock signal current configuration request
showLinkTrace  Show linktrace state and current configuration request
showLoopBack   Show loopback state and current configuration request
showTst        Show test signal statistics and current configuration request
updateDM       Update DM parameters request
updateTst      Update Tst signal request
```

Configuring Performance Monitoring with Default Configuration

You can set the default performance monitoring configurations on the UCS controller.

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**default**{**clearMepStats** | **setAis** | **setDM** | **setLM** |**setLck** | **setLinkTrace** | **setLoopBack** | **setTst** | **showAis** | **showDM** | **showLM** | **showLck** | **showLinkTrace** | **showLoopBack** | **showTst** | **updateDM** | **updateTst** |**exit**}
	
	Example:
	`UCS(OperationsMepPortType)# default ?`
	
	`clearMepStats  Clear mep statistics request`
	`exit           Exit from OpearationsMepPortType sub configuration mode`
	`setAis         Enable or Disable AIS request`
	`setDM          Enable or Disable delay measurement request`
	`setLM          Enable or Disable loss measurement request`
	`setLck         Enable or Disable lock signal request`
	
	`setLinkTrace   Enable or Disable linktrace request`
	`setLoopBack    Enable/Disable loopback`
	`setTst         Enable or Disable test signal request`
	
	`showAis        Show AIS configuration request`
	`showDM         Show delay measurement statistics request`
	`showLM         Show LM statistics request`
	`showLck        Show lock signal current configuration request`
	`showLinkTrace  Show linktrace state and current configuration request`
	`showLoopBack   Show loopback state and current configuration request`
	`showTst        Show test signal statistics and current configuration request`
	`updateDM       Update DM parameters request`
	`updateTst      Update Tst signal request`
		Sets the default configuration.
		
		• **clearMepStats—**Clears MEP statistics.
		• **setAis—**Enables or disables AIS.
		• **setDM—**Enables or disables delay measurement.
		• **setLM—**Enables or disables loss measurement.
		• **setLck—**Enables or disables lock signals
		• **setLinkTrace—**Enables or disables link traces.
		• **setLoopBack—**Enables or disables loopback
		• **setTst—**Enables or disables the test signal.
		• **showAis—**Displays AIS configuration request.
		• **showDM—**Displays delay measurement statistics.
		• **showLM—**Displays loss measurement statistics.
		• **showLck—**Displays current configured loss signals.
		• **showLinkTrace—**Displays current configured link trace state.
		• **showLoopBack—**Displays current configured loopback state.
		• **showTst—**Displays current configured test signals statistics.
		• **updateDM—**Updates the delay measurement parameters.
		• **updateTst—**Updates test signal parameters.

	Command or Action	Purpose
		• **exit**—Exits from OperationsMepPortType configuration mode.
Step 2	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuring Alarm Information Signal (AIS) on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setAis** {**commit** | **flush** | **aisConfig** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# setAis ?`
	`  aisConfig  Enable or Disable AIS request`
	`  commit     commit setAis`
	`  flush      flush all setAis commands from qu`
	`  review     review setAis commands`
		Configures alarm information signal (AIS).
		• **commit**—Sends the configuration to NID.
		• **flush**—Flushes all configuration from the queue.
		• **aisConfig**—Enables or disables the AIS configuration.
		• **review**—Displays the configuration on the UCS controller.

Step 2	**setAis aisConfig** {**aisaction** {**enable** {**frameRate** {**fr1m** | **fr1s**} | **disable**} | **protect** {**enable** | **disable**} } | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# setAis aisConfig aisaction enable frameRate fr1m`
	`UCS(OperationsMepPortType)# setAis aisConfig aisaction enable frameRate fr1s`
	`UCS(OperationsMepPortType)# setAis aisConfig aisaction enable protect enable`
	`UCS(OperationsMepPortType)# setAis aisConfig mepInstance 20`
		Configures AIS.
		• **aisaction**—Enables or disables AIS on the UCS controller.
		• **enable**—Enables lock signal configuration.
		• **disable**—Disables lock signal configuration.
		• **frameRate**—Indicates the frame rate.
		• **fr1m**—Indicates the frame rate is 1 f/m.
		• **fr1s**—Indicates the frame rate is 1 f/s.
		• **mepinstance***instance_no*—Indicates the MEP instance. The valid values are from 1 to 128.

	Command or Action	Purpose

Step 3	**setAis review**
	
	Example:
	`UCS(OperationsMepPortType)# setAis review`
	
	`Commands in queue:`
	`setAis aisConfig aisAction enable protect enable`
	`setAis aisConfig aisAction enable frameRate fr1s`
	`setAis aisConfig aisAction enable frameRate fr1m`
	`setAis aisConfig mepInstance 20`
		Displays the AIS configuration on the UCS controller.

Step 4	**setAiscommit** **Example:** `UCS(OperationsMepPortType)# setAis commit`	Sends the AIS configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to configure AIS on the US controller:

```
UCS(OperationsMepPortType)# setAis aisConfig aisaction enable frameRate fr1m
UCS(OperationsMepPortType)# setAis aisConfig aisaction enable frameRate fr1s
UCS(OperationsMepPortType)# setAis aisConfig aisaction enable protect enable
UCS(OperationsMepPortType)# setAis review
UCS(OperationsMepPortType)# setAis aisConfig mepInstance 20

Commands in queue:
        setAis aisConfig aisAction enable protect enable
        setAis aisConfig aisAction enable frameRate fr1s
        setAis aisConfig aisAction enable frameRate fr1m
        setAis aisConfig mepInstance 20

Commands in queue:
        Commands in queue:
        setAis aisConfig mepInstance 2

UCS(OperationsMepPortType)# setAis commit
SetAis_Output.mepResponse = 34537474

 SetAis Commit Success!!!
UCS(OperationsMepPortType)#end
```

Configuring Delay Measurement (DM) on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setDM** {**commit** | **flush** | **dmConfig** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# setDM ?`
	`  commit    commit setDM`
	`  dmConfig  Enable or Disable delay measurement request`
	`  flush     flush all setDM commands from queue`
	`  review    review setDM commands`
		Configures delay measurement.
		• **commit**—Sends the configuration to NID.
		• **flush**—Flushes all configuration from the queue.
		• **dmConfig**—Enables or disables the delay measurement configuration.
		• **review**—Displays the configuration on the UCS controller.

Step 2	**setDM dmConfig** {**dmaction** {**enable** {**calculation** {**flow** | **rdtrp**} | **cast** {**mutli** | **uni mepId***mep_Id* } | **interval***interval_no* | **lastN***delay_calc* | **mode** {**oneway** | **twoway**} | **priority***priority_no* } | **disable** } | **mepInstance** *instance_no* }
	
	Example:
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable calculation flow`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable calculation rdtrp`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable cast multi`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable cast uni mepId 23`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable interval 20`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable lastN 200`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable mode oneway`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable mode twoway`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable priority 3`
	`UCS(OperationsMepPortType)# setDM dmConfig dmaction enable calculation rdtrp`
	`UCS(OperationsMepPortType)# setDM dmConfig mepInstance 1`
		Sets DM parameters
		• **dmaction—**Enables or disables DM on the UCS controller.
		• **enable**—Enables delay measurement configuration.
		• **disable—**Disables delay measurement configuration.
		• **calculation**—Calculates delay.
		• **flow—**Two-way delay is calculated as round trip symmetrical flow. Far end resistance time is subtracted.
		• **rdtrp**—Two-way delay is calculated as round trip delay. Far end resistance time is *not* subtracted.
		• **cast**—Specifies transmission mode.
		• **multi—**Specifies OAM protocol data units (PDU) transmission with multicast MAC.
		• **uni**—Specifies OAM protocol data units (PDU) transmission with unicast MAC. The MAC is procured from the peer MEP MAC database.
		• **mepId***mep_id*—Specifies Peer MEP ID for unicast MAC.
		• **interval***interval_no*—Specifies the interval time between the PDU transmission in ms. The valid range is from 0 to 65535. The minimum value is 10ms.
		• **lastN—**Specifies the latest N delays for calculation. The valid range is from 10 to 2000.
		• **mode—**Specifies the mode of delay measurement.
		• **oneway—**Specifies mode on 1DM PDU measurement.
		• **twoway—**Specifies mode on DMM or DMR PDU.

	Command or Action	Purpose
		• **priority***priority_no*—Priority in case of tagged OAM. In the EVC domain this is the COS-ID. The valid range is from 0 to 7. • **mepinstance***instance_no*—Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**setDM review**
	
	Example:
	`UCS(OperationsMepPortType)# setDM review`
	
	`Commands in queue:`
	`setDM dmConfig dmAction enable mode oneWay`
	`setDM dmConfig dmAction enable interval 10`
	`setDM dmConfig dmAction enable lastN 20`
	`setDM dmConfig dmAction enable calculation flow`
	`setDM dmConfig dmAction enable calculation rdtrp`
	`setDM dmConfig dmAction enable priority 2`
	`setDM dmConfig dmAction enable calculation flow`
	`setDM dmConfig dmAction enable calculation rdtrp`
	`setDM dmConfig dmAction enable cast multi`
	`setDM dmConfig dmAction enable cast uni mepId 23`
	`setDM dmConfig dmAction enable interval 20`
	`setDM dmConfig dmAction enable priority 3`
	`setDM dmConfig mepInstance 2`
		Displays the DM configuration on the UCS controller.

Step 4	**setDMcommit**
	
	Example:
	`UCS(OperationsMepPortType)# setDM commit`
	
	`SetDM Commit Success!!!`
		Sends the DM configuration to the NID.

Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to configure DM on the UCS controller:

```
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable calculation flow
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable calculation rdtrp
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable cast multi
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable cast uni mepId 23
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable interval 20
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable lastN 200
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable mode oneway
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable mode twoway
```

```
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable priority 3
UCS(OperationsMepPortType)# setDM dmConfig dmaction enable calculation rdtrp
UCS(OperationsMepPortType)# setDM dmConfig mepInstance 1
UCS(OperationsMepPortType)# setDM review

Commands in queue:
        setDM dmConfig dmAction enable mode oneWay
        setDM dmConfig dmAction enable interval 10
        setDM dmConfig dmAction enable lastN 20
        setDM dmConfig dmAction enable calculation flow
        setDM dmConfig dmAction enable calculation rdtrp
        setDM dmConfig dmAction enable priority 2
        setDM dmConfig dmAction enable calculation flow
        setDM dmConfig dmAction enable calculation rdtrp
        setDM dmConfig dmAction enable cast multi
        setDM dmConfig dmAction enable cast uni mepId 23
        setDM dmConfig dmAction enable interval 20
        setDM dmConfig dmAction enable priority 3
        setDM dmConfig mepInstance 2

UCS(OperationsMepPortType)# setDM commit
DM.dmConfig.mepInstance = 119
DM.dmConfig.dmAction.t = 1
DM.dmConfig.dmAction.u.enable.priority = 4
DM.dmConfig.dmAction.u.enable.cast.t = 2
DM.dmConfig.dmAction.u.enable.cast.u.multi = 'any <b z="1">test</b> element'
DM.dmConfig.dmAction.u.enable.mode.t = 2
DM.dmConfig.dmAction.u.enable.mode.u.twoWay = 'any <b z="1">test</b> element'
DM.dmConfig.dmAction.u.enable.calculation.t = 2
DM.dmConfig.dmAction.u.enable.calculation.u.flow = 'any <b z="1">test</b> elemen
t'
DM.dmConfig.dmAction.u.enable.interval = 42689
DM.dmConfig.dmAction.u.enable.lastN = 1573
DM.dmConfig.mepInstance = 119
DM.dmConfig.dmAction.t = 1
DM.dmConfig.dmAction.u.enable.priority = 0
DM.dmConfig.dmAction.u.enable.cast.t = 1
DM.dmConfig.dmAction.u.enable.cast.u.uni.mepId = 23
DM.dmConfig.dmAction.u.enable.mode.t = 1
DM.dmConfig.dmAction.u.enable.mode.u.oneWay = 'one-way'
DM.dmConfig.dmAction.u.enable.calculation.t = 1
DM.dmConfig.dmAction.u.enable.calculation.u.rdtrp = 'rdtrp'
DM.dmConfig.dmAction.u.enable.interval = 42689
DM.dmConfig.dmAction.u.enable.lastN = 1573
SetDM_Output.mepResponse = 0

UCS(OperationsMepPortType)#end
```

Configuring Loss Measurement (LM) on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setLM** {**commit** | **flush** | **lmConfig** | **review**} **Example:** `UCS(OperationsMepPortType)# setLM ?`	Configures loss measurement (LM). • **commit**—Sends the configuration to NID. • **flush**—Flushes all configuration from the queue.

	Command or Action	Purpose

	`commit    commit setLM`
	`lmConfig  Enable or Disable loss measurement request`
	`flush     flush all setLM commands from queue`
	`review    review setLM commands`
		• **lmConfig**—Enables or disables the loss measurement configuration.
		• **review**—Displays the configuration on the UCS controller.

Step 2	**setLM LmConfig** { **lmaction** {**enable** {**cast** {**multi** | **uni**} | **flr***frame_interval* | **frameRate** {**fr10s** | **fr1m** | **fr1s** | **fr6h** | **fr6m**} | **mode** {**dual** | **single**} | **priority***priority_no* } | **disable** } | **mepInstance***instance_no* }
	
	Example:
	`UCS(OperationsMepPortType)# setLM lmConfig lmaction enable cast multi`
	`UCS(OperationsMepPortType)# setLM lmConfig lmaction enable cast uni`
	`UCS(OperationsMepPortType)# setLM lmConfig lmaction enable flr`
	`UCS(OperationsMepPortType)# setLM lmConfig lmaction frameRate fr10s`
	`UCS(OperationsMepPortType)# setLM lmConfig lmaction enable mode dual`
	`UCS(OperationsMepPortType)# setLM lmConfig lmaction enable priority 4`
	`UCS(OperationsMepPortType)# setLM lmConfig mepInstance 1`
		Sets LM parameters.
		• **lmaction**—Enables or disables LM on the UCS controller.
		• **enable**—Enables loss measurement configuration.
		• **disable**—Disables loss measurement configuration.
		• **cast**—Specifies transmission mode.
		• **multi**—Specifies OAM protocol data units (PDU) transmission with multicast MAC.
		• **uni**—Specifies OAM protocol data units (PDU) transmission with unicast MAC. The MAC is procured from the peer MEP MAC database.
		• **flr***frame_interval*—Specifies the frame loss ratio interval time. The valid range is from 0 to 99.
		• **frameRate**—Specifies the LM frame rate.
		◦ **fr10s** —Specifies the frame rate as 10 f/s.
		◦ **fr1m** —Specifies the frame rate as 1 f/min.
		◦ **fr1s** —Specifies the frame rate as 1 f/s.
		◦ **fr6h** —Specifies the frame rate as 6 f/hour.
		◦ **fr6m** —Specifies the frame rate as 6 f/min.
		• **mode**—Specifies the mode of delay measurement.
		• **dual**—Specifies dual LM mode on CCM PDU.
		• **single**—Specifies single LM mode on LMM or LMR PDU.
		• **priority***priority_no*—Priority in case of tagged OAM. In the EVC domain this is the COS-ID. The valid range is from 0 to 7.
		• **mepinstance***instance_no*—Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**setLM review**
	
	Example:
	`UCS(OperationsMepPortType)# setLM review`
	
	`Commands in queue:`
		Displays the LM configuration on the UCS controller.

	Command or Action	Purpose

	`setLM lmConfig lmAction enable cast multi`
	`setLM lmConfig lmAction enable cast uni`
	`setLM lmConfig lmAction enable flr 2`
	`setLM lmConfig lmAction enable frameRate fr10s`
	`setLM lmConfig lmAction enable mode dual`
	`setLM lmConfig lmAction enable priority 4`	

Step 4	**setLMcommit** **Example:** `UCS(OperationsMepPortType)#` **`setLM commit`** `SetLM Commit Success!!!`	Sends the LM configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#`**`exit`**	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to configure LM on the UCS controller:

```
UCS(OperationsMepPortType)# setLM lmConfig lmaction enable cast multi
UCS(OperationsMepPortType)# setLM lmConfig lmaction enable cast uni
UCS(OperationsMepPortType)# setLM lmConfig lmaction enable flr
UCS(OperationsMepPortType)# setLM lmConfig lmaction frameRate fr10s
UCS(OperationsMepPortType)# setLM lmConfig lmaction enable mode dual
UCS(OperationsMepPortType)# setLM lmConfig lmaction enable priority 4
UCS(OperationsMepPortType)# setLM lmConfig mepInstance 1
UCS(OperationsMepPortType)# setLM review

Commands in queue:
        setLM lmConfig lmAction enable cast multi
        setLM lmConfig lmAction enable cast uni
        setLM lmConfig lmAction enable flr 2
        setLM lmConfig lmAction enable frameRate fr10s
        setLM lmConfig lmAction enable mode dual
        setLM lmConfig lmAction enable priority 4

UCS(OperationsMepPortType)# setLM commit
SetLM-Output.mepResponse = 0
SetLM Commit Success!!!

UCS(OperationsMepPortType)#end
```

Configuring Lock Signal on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setLck** {**commit** | **flush** | **lckConfig** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# setLck ?`
	`commit     commit setLck`
	`flush      flush all setLck commands from queue`
	`lckConfig  Enable or Disable lock signal request`
	`review     review setLck commands`
		Configures lock signal.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **lckConfig—**Enables or disables the lock signal configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**setLck lckConfig** {**lckaction** {**enable frameRate** {**fr1m** | **fr1s**} | **disable**} | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# setlck lckConfig lckaction enable frameRate fr1m`
	`UCS(OperationsMepPortType)# setlck lckConfig lckaction mepInstance 1`
		Sets lock signal parameters.
		• **lckaction—**Enables or disables lock signal on the UCS controller.
		• **enable—**Enables lock signal configuration.
		• **frameRate—**Configures the frame rate.
		◦ **fr1m—**Specifies frame rate as 1 f/m.
		◦ **fr1s—**Specifies frame rate as 1 f/s.
		• **disable—**Disables lock signal configuration.
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**setlck review**
	
	Example:
	`UCS(OperationsMepPortType)# setlck review`
	
	`Commands in queue:`
	`setLck lckConfig lckAction enable frameRate fr1m`
	`setLck lckConfig mepInstance 1`
		Displays the lock signal configuration on the UCS controller.

Step 4	**setlckcommit** **Example:** `UCS(OperationsMepPortType)# setlck commit`	Sends the lock signal configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to configure lock signal on the UCS controller:

```
UCS(OperationsMepPortType)# setlck lckConfig lckaction enable frameRate frm1
UCS(OperationsMepPortType)# setlck lckConfig lckaction mepInstance 1
UCS(OperationsMepPortType)# setlck review

Commands in queue:
        setLck lckConfig lckAction enable frameRate fr1m
        setLck lckConfig mepInstance 1

UCS(OperationsMepPortType)# setlck commit
SetLck_Output.mepResponse = 0
SetLck Commit Success!!!UCS(OperationsMepPortType)
UCS(OperationsMepPortType)#end
```

Configuring LoopBack on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setLoopBack** {**commit** | **flush** | **loopBackConfig** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# setLoopBack ?`
	`  commit           commit setLoopBack`
	`  flush            flush all setLoopBack commands from queue`
	`  loopBackConfig   Enable/Disable loopback`
	`  review           review setLoopBack commands`
		Configures loopback.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **loopBackConfig—**Enables or disables the loopback configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**setLoopBack loopBackConfig** {**lbAction** {**enable** {**cast** {**multi** | **uni** {**destination** {**macAddress***target_MAC* | **mepid***mep_id*} } } | **count***count_no* | **dei** { **enable** | **disable**} | **interval** *interval* | **priority***priority_no* | **size** *bytes*} | **disable**} | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable cast multi`
	`UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable cast uni destination macAddress mac1`
	`UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable cast unidestination mepId 3`
	`UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable count 345`
		Sets loopback parameters.
		• **lbaction—**Enables or disables loop back on the UCS controller.
		• **enable—**Enables loop back trace configuration.
		• **cast—**Specifies the type of loop back configuration.
		• **multi—**Specifies OAM protocol data units (PDU) transmission with multicast MAC.
		• **uni—**Specifies OAM protocol data units (PDU) transmission with unicast MAC. The MAC is procured from the peer MEP MAC database.
		• **destination—**Specifies the target peer MEP.

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td></td><td>

```
UCS(OperationsMepPortType)# setLoopBack
loopbackConfig lbaction enable dei enable
UCS(OperationsMepPortType)# setLoopBack
loopbackConfig lbaction enable interval 20
UCS(OperationsMepPortType)# setLoopBack
loopbackConfig lbaction enable priority 7
UCS(OperationsMepPortType)# setLoopBack
loopbackConfig lbaction enable size 1400
UCS(OperationsMepPortType)# setLoopBack
loopbackConfig mepInstance 125
```

</td><td>

- **macAddress***target_MAC*—Specifies the MAC address for LT in MEP.
- **mepId***mep_id*—Specifies Peer MEP ID. The valid range is from 0 to 8191.
- **dei**—Specifies drop eligible indicator for tagged OAM.
- **enable**—Enables drop eligible indicator configuration.
- **disable**—Disables drop eligible indicator configuration.
- **count***count_no*—Specifies the number of loop back PDU sent in a single loop test .
- **interval***interval_no*—Specifies the interval time between the PDU transmission in ms. The valid range is from 0 to 65535. The minimum value is 10ms.
- **priority***priority_no*—Specifies the priority for tagged OAM. In EVC domain, it indicates the COS-ID. The valid range is from 0 to 7.
- **size** *frames*—Specifies the number of bytes. The valid range is from 1 to 1400.
- **disable**—Disables loop back configuration.
- **mepInstance***instance_no*—Indicates the MEP instance. The valid values are from 1 to 128.

</td></tr>
<tr><td>Step 3</td><td>

setloopback review

Example:

```
UCS(OperationsMepPortType)# setloopback review
Commands in queue:
        setLoopBack loopBackConfig lbAction enable
cast multi
        setLoopBack loopBackConfig lbAction enable
cast uni destination macAddress mac1
        setLoopBack loopBackConfig lbAction enable
cast uni destination mepId 3
        setLoopBack loopBackConfig lbAction enable
count 345
        setLoopBack loopBackConfig lbAction enable
dei enable
        setLoopBack loopBackConfig lbAction enable
interval 20
        setLoopBack loopBackConfig lbAction enable
priority 7
        setLoopBack loopBackConfig lbAction enable
size 1400
        setLoopBack loopBackConfig mepInstance 125
```

</td><td>Displays the loop back configuration on the UCS controller.</td></tr>
<tr><td>Step 4</td><td>

setlckcommit

Example:

```
UCS(OperationsMepPortType)# setloopback commit
```

</td><td>Sends the loop back configuration to the NID.</td></tr>
</table>

	Command or Action	Purpose
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to configure loop back on the UCS controller:

```
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable cast multi
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable cast uni destination
 macAddress mac1
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable cast unidestination
 mepId 3
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable count 345
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable dei enable
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable interval 20
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable priority 7
UCS(OperationsMepPortType)# setLoopBack loopbackConfig lbaction enable size 1400
UCS(OperationsMepPortType)# setLoopBack loopbackConfig mepInstance 125
UCS(OperationsMepPortType)# setsetloopback review

Commands in queue:
        setLoopBack loopBackConfig lbAction enable cast multi
        setLoopBack loopBackConfig lbAction enable cast uni destination macAddre
ss mac1
        setLoopBack loopBackConfig lbAction enable cast uni destination mepId 3
        setLoopBack loopBackConfig lbAction enable cast uni destination mepId 3
        setLoopBack loopBackConfig lbAction enable count 345
        setLoopBack loopBackConfig lbAction enable dei enable
        setLoopBack loopBackConfig lbAction enable interval 20
        setLoopBack loopBackConfig lbAction enable priority 7
        setLoopBack loopBackConfig lbAction enable size 1400

UCS(OperationsMepPortType)# setsetloopback commit
SetLoopBack_Output.mepResponse = 34275330


 SetLoopBack Commit Success!!!
UCS(OperationsMepPortType)#end
```

Configuring Link Trace on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setLinkTrace** {**commit** | **flush** | **linkTrace** | **review**}	Configures link trace.

	Command or Action	Purpose

	Example:
	`UCS(OperationsMepPortType)# setLinkTrace ?`
	`commit     commit setLinkTrace`
	`flush      flush all setLinkTrace commands from queue`
	`linkTrace  Enable or Disable linktrace request`
	`review     review setLinkTrace commands`
		• **commit**—Sends the configuration to NID.
		• **flush**—Flushes all configuration from the queue.
		• **linkTrace**—Enables or disables the link trace configuration.
		• **review**—Displays the configuration on the UCS controller.

Step 2	**setLinkTrace linkTrace** {**ltAction** {**enable** {**destination** {**macAddress***target_MAC* | **mepId***mep_id*} | **priority***priority_no* | **ttl***ttl_time* **disable**}} | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable destination macAddress mac1`
	`UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable destination mepId 3`
	`UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable priority 2`
	`UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable ttl 3`
		Sets link trace parameters.
		• **ltaction**—Enables or disables link trace on the UCS controller.
		• **enable**—Enables link trace configuration.
		• **destination**—Specifies the target peer MEP.
		  ◦ **macAddress***target_MAC*—Specifies the link trace MAC address for LT in MEP.
		  ◦ **mepId***mep_id*—Specifies Peer MEP ID for link trace. The valid range is from 0 to 8191.
		• **priority***priority_no*—Specifies the priority for tagged OAM. In EVC domain, it indicates the COS-ID. The valid range is from 0 to 7.
		• **ttl***ttl_time*—Specifies the time to live. The valid range is from 1 to 999.
		• **disable**—Disables link trace signal configuration.
		• **mepInstance***instance_no*—Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**setLinkTrace review** **Example:** `UCS(OperationsMepPortType)# setlinkTrace review`	Displays the link trace configuration on the UCS controller.
Step 4	**setlckcommit** **Example:** `UCS(OperationsMepPortType)# setlinkTrace commit`	Sends the link trace configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to configure link trace on the UCS controller:

```
UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable destination macAddress
 mac1
UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable destination mepId 3
UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable priority 2
UCS(OperationsMepPortType)# setlinkTrace linkTrace ltkaction enable ttl 3
UCS(OperationsMepPortType)# setlinkTrace review


UCS(OperationsMepPortType)# setlinkTrace commit
SetLinkTrace_Output.mepResponse = 34340866

 SetLinkTrace Commit Success!!!
UCS(OperationsMepPortType)#end
```

Configuring Test Signal on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setTst** {**commit** | **flush** | **tstConfig** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# setTst ?`
	`commit     commit setTst`
	`flush      flush all setTst commands from queue`
	`review     review setTst commands`
	`tstConfig  Enable or Disable test signal request`
		Configures test signal.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **tstConfig—**Enables or disables the test signal configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**setTst tstConfig** {**dei** {**enable** | **disable**} | **mepId***mep_Id* | **mepInstance***mep_instance* | **pattern** {**allOne** | **allZero** | **oneZero**} | **priority***priority_no* | **rate***bit_rate* | **sequence** {**enable** | **disable**} | **size** *frames*}
	
	Example:
	`UCS(OperationsMepPortType)# setTst tstConfig dei enable`
	`UCS(OperationsMepPortType)# setTst tstConfig mepid`
		Sets test signal parameters.
		• **dei—**Specifies drop eligible indicator for tagged OAM.
		• **enable—**Enables drop eligible indicator configuration.
		• **disable—**Disables drop eligible indicator configuration.
		• **mepId***mep_instance***—**Specifies Peer MEP ID. The valid range is from 0 to 8191.

	Command or Action	Purpose

	`2`
	`UCS(OperationsMepPortType)# setTst tstConfig mepinstance 2`
	`UCS(OperationsMepPortType)# setTst tstConfig pattern allOne`
	`UCS(OperationsMepPortType)# setTst tstConfig pattern allZero`
	`UCS(OperationsMepPortType)# setTst tstConfig sequence enable`
	`UCS(OperationsMepPortType)# setTst tstConfig rate 400`
	`UCS(OperationsMepPortType)# setTst tstConfig size 45`
		• **mepInstance***instance_no*—Indicates the MEP instance. The valid values are from 1 to 128.
		• **priority***priority_no*—Specifies the priority for tagged OAM. In EVC domain, it indicates the COS-ID. The valid range is from 0 to 7.
		• **pattern** —Specifies the sequence number is test PDU.
		• **allOne**—Specifies the test pattern to all one.
		• **allZero**—Specifies the test pattern to all zero.
		• **oneZero**—Specifies the test pattern to one zero.
		• **rate***bit_rate*—Specifies the test frame transmission rate in Megabits per second. The valid range is from 1 to 1518.
		• **sequence**—Enables or disables the sequence in test PDU.
		• **size** *frames*—Specifies the test pattern size in frames. The valid range is from 1 to 1518.

Step 3	**setTst review**
	
	Example:
	`UCS(OperationsMepPortType)# setTst review`
	`Commands in queue:`
	`setTst tstConfig dei enable`
	`setTst tstConfig mepId 2`
	`setTst tstConfig mepInstance 2`
	`setTst tstConfig pattern allOne`
	`setTst tstConfig pattern allZero`
	`setTst tstConfig priority 5`
	`setTst tstConfig pattern allZero`
	`setTst tstConfig rate 400`
	`setTst tstConfig size 45`
		Displays the test signal configuration on the UCS controller.

Step 4	**setTstcommit** **Example:** `UCS(OperationsMepPortType)# setTst commit`	Sends the link trace configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to configure test signal on the UCS controller:

```
UCS(OperationsMepPortType)# setTst tstConfig dei enable
UCS(OperationsMepPortType)# setTst tstConfig mepid 2
UCS(OperationsMepPortType)# setTst tstConfig mepinstance 2
UCS(OperationsMepPortType)# setTst tstConfig pattern allOne
UCS(OperationsMepPortType)# setTst tstConfig pattern allZero
UCS(OperationsMepPortType)# setTst tstConfig rate 400
UCS(OperationsMepPortType)# setTst tstConfig sequence enable
UCS(OperationsMepPortType)# setTst tstConfig size 45

UCS(OperationsMepPortType)# setTst review
Commands in queue:
        setTst tstConfig dei enable
        setTst tstConfig mepId 2
        setTst tstConfig mepInstance 2
        setTst tstConfig pattern allOne
        setTst tstConfig pattern allZero
        setTst tstConfig priority 5
        setTst tstConfig pattern allZero
        setTst tstConfig rate 400
        setTst tstConfig size 45

UCS(OperationsMepPortType)# setTst commit
SetTst_Output.mepResponse = 34471938
SetTst Commit Success!!!
UCS(OperationsMepPortType)#end
```

Viewing Alarm Information Signal (AIS) on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showAis** {**commit** | **flush** | **mepRequest** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# showAis ?`
	`commit      commit showAis`
	`flush       flush all showAis commands from queue`
	`mepRequest  Show AIS configuration request`
	`review      review showAis commands`
		Displays alarm information signal configuration.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **mepRequest—**Displays the alarm configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showAis mepRequest** {**all** | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# showAis mepRequest all`
	
	`UCS(OperationsMepPortType)# showAis mepRequest mepInstance 120`
		• **all—**Displays AIS configuration for all MEPs on the UCS controller.
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

	Command or Action	Purpose

Step 3	**showAis review**
	
	Example:
	`UCS(OperationsMepPortType)# showAis review`
	
	`Commands in queue:`
	`        showAis mepRequest all`
	`        showAis mepRequest mepInstance 120`
		Displays the configuration on the UCS controller.

Step 4	**showAis commit** **Example:** `UCS(OperationsMepPortType)# showAis commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to display the AIS on the UCS controller:

```
UCS(OperationsMepPortType)# showAis mepRequest all
UCS(OperationsMepPortType)# showAis mepRequest mepInstance 120
UCS(OperationsMepPortType)# showAis review

Commands in queue:
        showAis mepRequest all
        showAis mepRequest mepInstance 120

UCS(OperationsMepPortType)# showAis commit
Ais_Output.aisInfo.mepInst[0].config.mepInstance = 100
ShowAis_Output.aisInfo.mepInst[0].config.frameRate.t = 2
ShowAis_Output.aisInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowAis_Output.aisInfo.mepInst[0].config.protect.t = 1
ShowAis_Output.aisInfo.mepInst[0].config.protect.u.enable = 'enable'

 showAis Commit Success!!!
UCS(OperationsMepPortType)# end
```

Viewing Delay Measurement (DM) Statistics on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showDM** {**commit** | **flush** | **mepRequest** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# showDM ?`
	`  commit      commit showDM`
	`  flush       flush all showDM commands from queue`
	`  mepRequest  Show delay measurement statistics request`
	`  review      review showDM commands`
		Displays delay measurement (DM).
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **mepRequest—**Displays the configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showDM mepRequest** {**all** | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# showDM mepRequest all`
	`UCS(OperationsMepPortType)# showDM mepRequest mepInstance 100`
		• **all—**Displays DM configuration for all MEPs on the UCS controller.
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**showDM review**
	
	Example:
	`UCS(OperationsMepPortType)# showDM review`
	`Commands in queue:`
	`        showDM mepRequest all`
	`        showDM mepRequest all`
	`        showDM mepRequest mepInstance 100`
		Displays the configuration on the UCS controller.

Step 4	**showDM commit** **Example:** `UCS(OperationsMepPortType)# showDM commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to display the delay measurement statistics on the UCS controller:

```
UCS(OperationsMepPortType)# showDM mepRequest all
UCS(OperationsMepPortType)# showDM mepRequest mepInstance 100
UCS(OperationsMepPortType)# showDM review
Commands in queue:
        showDM mepRequest all
        showDM mepRequest all
        showDM mepRequest mepInstance 100

UCS(OperationsMepPortType)# showDM commit
ShowDM_Output.dmInfo.mepInst[0].config.mepInstance = 98
ShowDM_Output.dmInfo.mepInst[0].config.priority = 7
```

```
ShowDM_Output.dmInfo.mepInst[0].config.cast.t = 1
ShowDM_Output.dmInfo.mepInst[0].config.cast.u.uni.mepId = 106
ShowDM_Output.dmInfo.mepInst[0].config.mode.t = 2
ShowDM_Output.dmInfo.mepInst[0].config.mode.u.twoWay = 'two-way'
ShowDM_Output.dmInfo.mepInst[0].config.txMode.t = 1
ShowDM_Output.dmInfo.mepInst[0].config.txMode.u.standardize = 'standardize'
ShowDM_Output.dmInfo.mepInst[0].config.calculation.t = 1
ShowDM_Output.dmInfo.mepInst[0].config.calculation.u.rdtrp = 'rdtrp'
ShowDM_Output.dmInfo.mepInst[0].config.interval = 10
ShowDM_Output.dmInfo.mepInst[0].config.lastN = 10
ShowDM_Output.dmInfo.mepInst[0].config.unit.t = 2
ShowDM_Output.dmInfo.mepInst[0].config.unit.u.us = 'micro seconds'
ShowDM_Output.dmInfo.mepInst[0].config.syncronized.t = 2
ShowDM_Output.dmInfo.mepInst[0].config.syncronized.u.disable = 'Disable'
ShowDM_Output.dmInfo.mepInst[0].config.overflowReset.t = 1
ShowDM_Output.dmInfo.mepInst[0].config.overflowReset.u.keep = 'keep'
ShowDM_Output.dmInfo.mepInst[0].state.mepInstance = 106
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.tx = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.rxTimeout = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.rx = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.rxError = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.avgTotal = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.avgLastN = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.min = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.max = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.avgVariationTotal = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.avgVariationLastN = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.minVar = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.maxVar = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.F_to_N.overflow = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.tx = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.rxTimeout = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.rx = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.rxError = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.avgTotal = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.avgLastN = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.min = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.max = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.avgVariationTotal = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.avgVariationLastN = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.minVar = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.maxVar = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.oneWay.N_to_F.overflow = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.tx = 793
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.rxTimeout = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.rx = 793
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.rxError = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.avgTotal = 17
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.avgLastN = 17
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.min = 17
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.max = 18
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.avgVariationTotal = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.avgVariationLastN = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.minVar = 0
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.maxVar = 1
ShowDM_Output.dmInfo.mepInst[0].state.mode.twoWay.overflow = 0
 showDM Commit Success!!!

UCS(OperationsMepPortType)# end
```

Viewing Loss Measurement (LM) Statistics on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLM** {**commit** | **flush** | **mepRequest** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# `**`showLM ?`**
	`  commit      commit showLM`
	`  flush       flush all showLM commands from queue`
	`  mepRequest  Show LM statistics request`
	`  review      review showLM commands`
		Displays loss measurement configuration.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **mepRequest—**Displays the configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showLM mepRequest** {**all** | **mepInstance***instance_no*} **Example:** `UCS(OperationsMepPortType)# `**`showLM mepRequest all`** `UCS(OperationsMepPortType)# `**`showLM mepRequest mepInstance 100`**	• **all—**Displays LM statistics for all MEPs on the UCS controller. • **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**showLM review**
	
	Example:
	`UCS(OperationsMepPortType)# `**`showLM review`**
	`Commands in queue:`
	`        showLM mepRequest all`
	`        showLM mepRequest mepInstance 100`
		Displays the configuration on the UCS controller.

Step 4	**showLM commit** **Example:** `UCS(OperationsMepPortType)# `**`showLM commit`**	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#`**`exit`**	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to display the loss measurement statistics on the UCS controller:

```
UCS(OperationsMepPortType)# showLM mepRequest all
UCS(OperationsMepPortType)# showLM mepRequest mepInstance 100
UCS(OperationsMepPortType)# showLM review

CCommands in queue:
        showLM mepRequest all
        showLM mepRequest mepInstance 100

UCS(OperationsMepPortType)# showLM commit
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.priority = 7
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.t = 2
```

```
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.cast.u.multi = 'multi'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.t = 2
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.mode.u.single = 'single'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.t = 3
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.frameRate.u.fr1s = 'fr1s'
ShowLM_Output.lossMeasurentInfo.mepInst[0].config.flr = 5
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.mepInstance = 98
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.tx = 137
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.rx = 137
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearCount = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farCount = 1105217
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.nearRatio = 0
ShowLM_Output.lossMeasurentInfo.mepInst[0].state.farRatio = 94
 showLM Commit Success!!!

UCS(OperationsMepPortType)# exit
```

Viewing Lock Signal on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showlck** {**commit** | **flush** | **mepRequest** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# showAis ?`
	`commit      commit showLck`
	`flush       flush all showLck commands from queue`
	`mepRequest  Show lock signal current configuration request`
	`review      review showLck commands`
		Displays lock signal information.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **mepRequest—**Displays the configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showlck mepRequest** {**all** | **mepInstance** *instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# showlck mepRequest all`
	
	`UCS(OperationsMepPortType)# showlck mepRequest mepInstance 20`
		• **all—**Displays lock signal configuration for all MEPs on the UCS controller.
		• **mepInstance** *instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**showlck review**
	
	Example:
	`UCS(OperationsMepPortType)# showlck review`
	`Commands in queue:`
	`showLck mepRequest all`
	`showLck mepRequest mepInstance 20`
		Displays the configuration on the UCS controller..

	Command or Action	Purpose
Step 4	**setlck commit** **Example:** `UCS(OperationsMepPortType)# showlck commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)# exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to display the lock signal on the UCS controller:

```
UCS(OperationsMepPortType)# showlck mepRequest all
UCS(OperationsMepPortType)# showlck mepRequest mepInstance 20
UCS(OperationsMepPortType)# showlck review

Commands in queue:
       showLck mepRequest all
       showLck mepRequest mepInstance 20

UCS(OperationsMepPortType)# showlck commit
SetLck_Output.mepResponse = 0

 SetLck Commit Success!!!
UCS(OperationsMepPortType)# end
```

Viewing Loopback State on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLoopBack** {**commit** | **flush** | **mepRequest** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# showLoopBack ?`
	`   commit      commit showLoopBack`
	`   flush       flush all showLoopBack commands from queue`
	`   mepRequest  Show loopback state and current configuration request`
	`   review      review showLoopBack commands`
		Display loopback information.
		• **commit**—Sends the configuration to NID.
		• **flush**—Flushes all configuration from the queue.
		• **mepRequest**—Displays the configuration.
		• **review**—Displays the configuration on the UCS controller.

	Command or Action	Purpose

Step 2	**showLoopBack mepRequest** {**all** | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# showLoopBack mepRequest all`
	`UCS(OperationsMepPortType)# showLoopBack mepRequest mepInstance 30`
		• **all—**Displays loopback configuration for all MEPs on the UCS controller.
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**showLoopBack review**
	
	Example:
	`UCS(OperationsMepPortType)# showLoopBack review`
	
	`Commands in queue:`
	`showLoopBack mepRequest all`
	`showLoopBack mepRequest mepInstance 30`
		Displays the configuration on the UCS controller.

Step 4	**showLoopBack commit** **Example:** `UCS(OperationsMepPortType)# showLoopBack commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to display the loop back state on the controller:

```
UCS(OperationsMepPortType)# showLoopBack mepRequest all
UCS(OperationsMepPortType)# showLoopBack mepRequest mepInstance 30
UCS(OperationsMepPortType)# showLoopBack review

Commands in queue:
        showLoopBack mepRequest all
        showLoopBack mepRequest mepInstance 30

UCS(OperationsMepPortType)# showLoopBack commit
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.mepInstance = 100
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.dei.t = 2
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.dei.u.disable = 'DEI Disable'
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.priority = 7
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.cast.t = 2
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.cast.u.multi = 'MULTI'
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.count = 10
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.size = 70
ShowLoopBack_Output.loopbackInfo.mepInst[0].config.interval = 1
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.mepInstance = 32
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.transactionId = 11
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.txLBM.upper = 0
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.txLBM.lower = 10
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].rcvMac = '00-3A-99-FD-47-2F'
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].received.upper = 0
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].received.lower = 10
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].outOfOrder.upper = 0
```

```
ShowLoopBack_Output.loopbackInfo.mepInst[0].state.reply[0].outOfOrder.lower = 0

UCS(OperationsMepPortType)# exit
```

Viewing Link Trace State on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showLinkTrace** {**commit** | **flush** | **mepRequest** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# showLinkTrace ?`
	`  commit      commit showLinkTrace`
	`  flush       flush all showLinkTrace commands from queue`
	`  mepRequest  Show linktrace state and current configuration request`
	`  review      review showLinkTrace commands`
		Displays link trace configuration.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **mepRequest—**Displays the configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showLinkTrace mepRequest** {**all** | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# showLinkTrace mepRequest all`
	`UCS(OperationsMepPortType)# showLinkTrace mepRequest mepInstance 40`
		• **all—**Displays link trace state configuration for all MEPs on the UCS controller.
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**showLinkTrace review**
	
	Example:
	`UCS(OperationsMepPortType)# showLinkTrace review`
	
	`Commands in queue:`
	`        showLinkTrace mepRequest all`
	`        showLinkTrace mepRequest mepInstance 120`
		Displays the configuration on the UCS controller.

Step 4	**showLinkTrace commit** **Example:** `UCS(OperationsMepPortType)# showLinkTrace commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to display the link trace state on the UCS controller:

```
UCS(OperationsMepPortType)# showLinkTrace mepRequest all
UCS(OperationsMepPortType)# showLinkTrace mepRequest mepInstance 40
UCS(OperationsMepPortType)# showLinkTrace review

Commands in queue:
        showLinkTrace mepRequest all
        showLinkTrace mepRequest mepInstance 40

UCS(OperationsMepPortType)# showLinkTrace commit
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].config.mepInstance = 100
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].config.priority = 7
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].config.mepId = 101
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].config.macAddress = '00-00-00-00-00-00'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].config.ttl = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].transactionId = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].ttl = 0
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].mode.t = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].mode.u.MEP = 'MEP'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].direction.t = 2
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].direction.u.DOWN = 'DOWN'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].forwarded.t = 2
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].forwarded.u.NO = 'Not
forwarded'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].relay = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].lastMac = '00-3A-99-FD-4A-53'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[0].reply[0].nextMac = '00-3A-99-FD-47-2F'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].transactionId = 2
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].ttl = 0
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].mode.t = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].mode.u.MEP = 'MEP'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].direction.t = 2
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].direction.u.DOWN = 'DOWN'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].forwarded.t = 2
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].forwarded.u.NO = 'Not
forwarded'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].relay = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].lastMac = '00-3A-99-FD-4A-53'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[1].reply[0].nextMac = '00-3A-99-FD-47-2F'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].transactionId = 3
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].ttl = 0
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].mode.t = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].mode.u.MEP = 'MEP'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].direction.t = 2
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].direction.u.DOWN = 'DOWN'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].forwarded.t = 2
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].forwarded.u.NO = 'Not
forwarded'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].relay = 1
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].lastMac = '00-3A-99-FD-4A-53'
ShowLinkTrace_Output.linkTraceInfo.mepInst[0].state[2].reply[0].nextMac = '00-3A-99-FD-47-2F'

 showLinkTrace Commit Success!!!

UCS(OperationsMepPortType)# end
```

Viewing Test Signal Statistics on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showTst** {**commit** | **flush** | **mepRequest** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# showTst ?`
	`commit commit showTst`
	`flush flush all showTst commands from queue`
	`mepRequest Show test signal statistics and current configuration request`
	`review review showTst commands`
		Displays test signal statistics.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **mepRequest—**Displays the configuration.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**showTst mepRequest** {**all** | **mepInstance***instance_no*}
	
	Example:
	`UCS(OperationsMepPortType)# showTst mepRequest all`
	`UCS(OperationsMepPortType)# showTst mepRequest mepInstance 50`
		• **all—**Displays test signal statistics configuration for all MEPs on the UCS controller.
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.

Step 3	**showTst review**
	
	Example:
	`UCS(OperationsMepPortType)# showTst review`
	
	`Commands in queue:`
	`showTst mepRequest all`
	`showTst mepRequest mepInstance 50`
		Displays the configuration on the UCS controller.

Step 4	**showTstcommit** **Example:** `UCS(OperationsMepPortType)# showTst commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to display the test signal statistics on the UCS controller:

```
UCS(OperationsMepPortType)# showTst mepRequest all
UCS(OperationsMepPortType)# showTst mepRequest mepInstance 120
UCS(OperationsMepPortType)# showTst review

Commands in queue:
        showTst mepRequest all
        showTst mepRequest mepInstance 50

UCS(OperationsMepPortType)# showTst commit
ShowTst_Output.tstInfo.mepInst[0].config.mepInstance = 100
ShowTst_Output.tstInfo.mepInst[0].config.dei.t = 2
```

```
ShowTst_Output.tstInfo.mepInst[0].config.dei.u.disable = 'Disable'
ShowTst_Output.tstInfo.mepInst[0].config.priority = 7
ShowTst_Output.tstInfo.mepInst[0].config.mepId = 101
ShowTst_Output.tstInfo.mepInst[0].config.rate = 1000
ShowTst_Output.tstInfo.mepInst[0].config.size = 64
ShowTst_Output.tstInfo.mepInst[0].config.pattern.t = 1
ShowTst_Output.tstInfo.mepInst[0].config.pattern.u.allZero = 'all-zero'
ShowTst_Output.tstInfo.mepInst[0].config.sequence.t = 1
ShowTst_Output.tstInfo.mepInst[0].config.sequence.u.enable = 'Enable'
ShowTst_Output.tstInfo.mepInst[0].config.Tx.t = 1
ShowTst_Output.tstInfo.mepInst[0].config.Tx.u.enable = 'Enable'
ShowTst_Output.tstInfo.mepInst[0].config.Rx.t = 1
ShowTst_Output.tstInfo.mepInst[0].config.Rx.u.enable = 'Enable'
ShowTst_Output.tstInfo.mepInst[0].state.txFrameCount.upper = 0
ShowTst_Output.tstInfo.mepInst[0].state.txFrameCount.lower = 241803
ShowTst_Output.tstInfo.mepInst[0].state.rxFrameCount.upper = 0
ShowTst_Output.tstInfo.mepInst[0].state.rxFrameCount.lower = 0
ShowTst_Output.tstInfo.mepInst[0].state.rxRate = 0
ShowTst_Output.tstInfo.mepInst[0].state.testTime = 162
 showTst Commit Success!!!

UCS(OperationsMepPortType)# end
```

Updating Delay Measurement (DM) on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**updateDM** {**commit** | **flush** | **updateDmConfig** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# updateDM ?`
	`  commit           commit updateDM`
	`  flush            flush all updateDM commands from queue`
	`  review           review updateDM commands`
	`  updateDmConfig   Update DM parameters request`
		Updates delay measurement (DM).
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **updateDmConfig—**Updates the delay measurement parameters.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**updateDM updateDmConfig** {**mepInstance***instance_no* | **update** {**overflowReset** {**keep** | **reset**} | **syncronized** {**enable** | **disable**} | **txMode** {**proprietary** | **standardize**} | **unit** {**ns** | **us**}}}
	
	Example:
	`UCS(OperationsMepPortType)# updateDM updateDmConfig mepInstance 100`
	`UCS(OperationsMepPortType)# updateDM updateDmConfig update overflowReset keep`
	`UCS(OperationsMepPortType)# updateDM updateDmConfig update overflowReset reset`
	`UCS(OperationsMepPortType)# updateDM updateDmConfig update synchronized enable`
		Updates DM parameters.
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.
		• **update—**Updates DM parameters for all MEPs on the UCS controller.
		• **overflowReset—**Reset all delay Measurement results on total delay counters.
		• **keep—**Retains all delay Measurement results.
		• **reset—**Resets all delay Measurement results.

	Command or Action	Purpose

	`UCS(OperationsMepPortType)# updateDM updateDmConfig update overflowReset keep`
	`UCS(OperationsMepPortType)# updateDM updateDmConfig update txMode proprietary`
	`UCS(OperationsMepPortType)# updateDM updateDmConfig update txMode standardize`
	`UCS(OperationsMepPortType)# updateDM updateDmConfig update unit ns`
	`UCS(OperationsMepPortType)# updateDM updateDmConfig update unit us`
		• **syncronized—**Synchronizes near end and far end time intervals.
		• **enable—**Enables synchronization of near and far end time interval.
		• **disable—**Disables synchronization of near and far end time interval.
		• **txMode—**Sets the Tx mode.
		• **proprietary—**Sets the proprietary delay measurement parameters .
		• **standardize—**Sets the Y.1731 standards to transmit 1DM/DMR delay measurement parameters.
		• **unit—**Sets the delay in units.
		• **ns—**Specifies nanoseconds.
		• **us—**Sets microseconds.

Step 3	**updateDM review**
	
	Example:
	`UCS(OperationsMepPortType)# updateDM review`
	`Commands in queue:`
	`        updateDM updateDmConfig mepInstance 1`
	`        updateDM updateDmConfig update overflowReset keep`
	`        updateDM updateDmConfig update overflowReset reset`
	`        updateDM updateDmConfig update syncronized enable`
	`        updateDM updateDmConfig update txMode proprietary`
	`        updateDM updateDmConfig update txMode standardize`
	`        updateDM updateDmConfig update txMode standardize`
	`        updateDM updateDmConfig update unit ns`
	`        updateDM updateDmConfig update unit us`
		Displays the configuration on the UCS controller.

Step 4	**updateDM commit** **Example:** `UCS(OperationsMepPortType)# updateDM commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to update the delay measurement parameters on the UCS controller:

```
UCS(OperationsMepPortType)# updateDM updateDmConfig mepInstance 100
UCS(OperationsMepPortType)# updateDM updateDmConfig update overflowReset keep
UCS(OperationsMepPortType)# updateDM updateDmConfig update overflowReset reset
UCS(OperationsMepPortType)# updateDM updateDmConfig update synchronized enable
UCS(OperationsMepPortType)# updateDM updateDmConfig update overflowReset keep
UCS(OperationsMepPortType)# updateDM updateDmConfig update txMode proprietary
UCS(OperationsMepPortType)# updateDM updateDmConfig update txMode standardize
UCS(OperationsMepPortType)# updateDM updateDmConfig update unit ns
UCS(OperationsMepPortType)# updateDM updateDmConfig update unit us
UCS(OperationsMepPortType)# updateDM review
Commands in queue:
        updateDM updateDmConfig mepInstance 1
        updateDM updateDmConfig update overflowReset keep
        updateDM updateDmConfig update overflowReset reset
        updateDM updateDmConfig update syncronized enable
        updateDM updateDmConfig update txMode proprietary
        updateDM updateDmConfig update txMode standardize
        updateDM updateDmConfig update txMode standardize
        updateDM updateDmConfig update unit ns
        updateDM updateDmConfig update unit us

UCS(OperationsMepPortType)# updateDM commit
UCS(OperationsMepPortType)# end
```

Updating Test Signal Parameters on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**updateTst** {**commit** | **flush** | **updateTstConfig** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# updateTst ?`
	`  commit            commit updateTst`
	`  flush             flush all updateTst commands from queue`
	`  review            review updateTst commands`
	`  updateTstConfig   Update Tst signal request`
		Updates the test signal parameters.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **updateTstConfig—**Updates the test signal parameters.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**updateTst updateTstConfig** {**mepInstance***instance_no* | **update** {**Rx** | **Tx**} {**enable** | **disable**}}
	
	Example:
	`UCS(OperationsMepPortType)# updateTst updatetstConfig mepInstance 100`
	`UCS(OperationsMepPortType)# updateTst updatetstConfig`
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.
		• **update—**Updates DM parameters for all MEPs.
		• **Rx—**Sets the Rx mode.
		• **Tx—**Sets the Tx mode.

	Command or Action	Purpose
	`update Rx enable` `UCS(OperationsMepPortType)# updateTst updatetstConfig update Tx enable`	• **enable**—Enables the mode. • **disable**—Disables the mode.
Step 3	**updateTst review** **Example:** `UCS(OperationsMepPortType)# updateTst review` `Commands in queue:` `updateTst updateTstConfig mepInstance 2` `updateTst updateTstConfig update Rx enable` `updateTst updateTstConfig update Tx enable` `updateTst updateTstConfig update Tx enable`	Displays the configuration on the UCS controller.
Step 4	**updateTst commit** **Example:** `UCS(OperationsMepPortType)# updateTst commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to update the test signal parameters on the UCS controller:

```
UCS(OperationsMepPortType)# updateTst updatetstConfig mepInstance 100
UCS(OperationsMepPortType)# updateTst updatetstConfig update Rx enable
UCS(OperationsMepPortType)# updateTst updatetstConfig update Tx enable

UCS(OperationsMepPortType)# updateTst review
Commands in queue:
        updateTst updateTstConfig mepInstance 2
        updateTst updateTstConfig update Rx enable
        updateTst updateTstConfig update Tx enable
        updateTst updateTstConfig update Tx enable

UCS(OperationsMepPortType)# updateTst commit
UCS(OperationsMepPortType)# end
```

Clearing MEP Statistics on the UCS Controller

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**clearMepStats** {**commit** | **flush** | **clearStats** | **review**}
	
	Example:
	`UCS(OperationsMepPortType)# clearMepStats ?`
	`   clearStats  Clear mep statistics request`
	`   commit      commit clearMepStats`
	`   flush       flush all clearMepStats commands from queue`
	`   review      review clearMepStats commands`
		Clears the MEP statistics.
		• **commit—**Sends the configuration to NID.
		• **flush—**Flushes all configuration from the queue.
		• **clearStats—**Clears the MEP statistics.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**clearMepStats cleatStats** {**mepInstance***instance_no* | **StatsType** {**DM** | **LM** | **TST**}}
	
	Example:
	`UCS(OperationsMepPortType)# clearMepStats clearStats mepInstance 25`
	`UCS(OperationsMepPortType)# clearMepStats clearStats statstype DM`
		• **mepInstance***instance_no***—**Indicates the MEP instance. The valid values are from 1 to 128.
		• **Statstype—**Indicates the protocol type.
		• **DM—**Specifies the delay measurement statistics.
		• **LM—**Specifies the loss measurement statistics.
		• **TST—**Specifies the test signal statistics.

Step 3	**clearMepStats review**
	
	Example:
	`UCS(OperationsMepPortType)# clearMepStats review`
	
	`Commands in queue:`
	`        clearMepStats clearStats mepInstance 23`
		Displays the configuration on the UCS controller.

Step 4	**clearMepStats commit** **Example:** `UCS(OperationsMepPortType)# clearMepStats commit`	Sends the configuration to the NID.
Step 5	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Configuration Example

The example shows how to clear the MEP statistics on the UCS controller:

```
UCS(OperationsMepPortType)# clearMepStats clearStats mepInstance 25
UCS(OperationsMepPortType)# clearMepStats clearStats statstype DM
UCS(OperationsMepPortType)# clearMepStats review

Commands in queue:
        clearMepStats clearStats mepInstance 23
```

```
UCS(OperationsMepPortType)# clearMepStats commit
UCS(OperationsMepPortType)#end
```

Negating Performance Monitoring Configuration and Restoring Defaults

Before You Begin

- Perform the steps to provision performance monitoring on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**no ?**
	
	Example:
	`UCS(OperationsMepPortType)# no ?`
	
	`  clearMepStats  Clear mep statistics request`
	`  exit           Exit from OperationsMepPortType sub configuration mode`
	`  setAis         Enable or Disable AIS request`
	`  setDM          Enable or Disable delay measurement request`
	`  setLM          Enable or Disable loss measurement request`
	`  setLck         Enable or Disable lock signal request`
	`  setLinkTrace   Enable or Disable linktrace request`
	`  setLoopBack    Enable/Disable loopback`
	`  setTst         Enable or Disable test signal request`
	`  showAis        Show AIS configuration request`
	`  showDM         Show delay measurement statistics request`
	`  showLM         Show LM statistics request`
	`  showLck        Show lock signal current configuration request`
	`  showLinkTrace  Show linktrace state and current configuration request`
	`  showLoopBack   Show loopback state and current configuration request`
	`  showTst        Show test signal statistics and current configuration`
	`request`
	`  updateDM       Update DM parameters request`
	`  updateTst      Update Tst signal request`
		Negates the commands and sets the default configuration.

Step 2	**exit** **Example:** `UCS(OperationsMepPortType)#exit`	Exits the OperationsMepPortType mode.

Verifying Performance Monitoring

Use the **show perf-mon** commands to verify the Performance Monitoring status on the UCS controller.

show perf-mon {**current** | **interval-id** *unit* **instance***instance_id* | **interval-info** | **id** *unit*} {**feature** {**dm** | **ece** | **evc** | **lm**}

- This command displays the current delay measurement status. The following is a sample output from the command:
  ```
  UCS# show perf-mon current feature dm
  ```

- This command displays the current loss measurement status The following is a sample output from the command:
  ```
  UCS# show perf-mon current feature lm
  ```

- This command displays the delay measurement status for interval-id. The following is a sample output from the command:
  ```
  UCS# show perf-mon interval-id id 3 instance 4 feature dm
  ```

- This command displays the loss measurement status for interval-info. The following is a sample output from the command:
  ```
  UCS# show perf-mon interval-info 5 feature lm
  ```

CHAPTER 19

Configuring EPS

This document describes the Ethernet Protection Switching (EPS) feature and configuration steps to implement protection switching mechanisms for Ethernet layer topologies.

Prerequisites for Configuring EPS

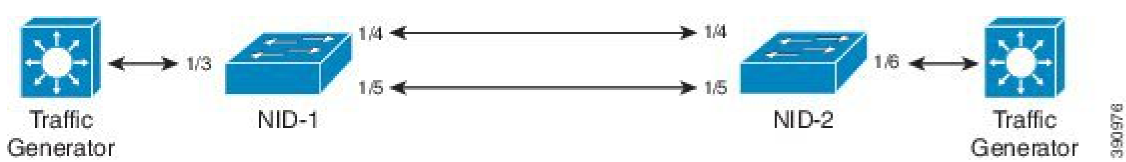
-

Information About EPS

EPS is a fully allocated protection mechanism that ensures the route and bandwidth of the protection entity are reserved for a selected working entity. It provides a fast and simple protection mechanism. It is easier for the network administrators to monitor the status of the network (e.g., active network topology) with EPS when compared with other protocols such as Rapid Spanning Tree Protocol (RSTP).

The following figure shows the topology used for provisioning EPS on NID-1 and NID-2.

Figure 11: EPS Topology



How to Provision EPS

Creating MEP on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** `UCS# ProvisionMepPortType`	Enters the ProvisionMepPortType mode.

Step 3	**createMep** {**createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level***level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 20`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepId 12`
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 4`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg`
	`UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112`
		Creates MEP configuration.
		• **mepinstance—**Specifies the MEP instance number.
		• **mode—**Specifies the mode of the MEP instance.
		• **mep—**Specifies the maintenance entity end point.
		• **mip—**Specifies the maintenance entity intermediate point.
		• **direction—**Selects the direction of the MEP.
		• **up—**Specifies an Up MEP - monitoring egress OAM and traffic on residence port.
		• **down—**Specifies a Down MEP - monitoring ingress OAM and traffic on residence port.
		• **domain—**Selects the domain of the MEP.
		• **port—**Specifies a MEP in the Port Domain. Flow Instance is a Port.
		• **evc—**Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created.
		• **vlan—**Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created.
		• **flowId—**Specifies the flow related to the MEP.
		• **vid—**In case the MEP is a port Up-MEP or a EVC customer MIP the VID must be given.
		• **level—**Specifies the MEG level of the MEP.
		• *level_number*—MEG level number.

	Command or Action	Purpose
		• **residencePort—**Specifies the port monitored by MEP. • *port_number*—Residence port number. • **mepId—**Specifies MEP ID. • *id_number*—MEP ID number. • **megdomain—**Specifies the maintenance domain configuration. • **maName—**Specifies the ITU/IEEE MEG-ID (short MA name). • *ma_name*—Short MA name. • **megIdFormat—**Selects the MEG ID format. • **ituMeg—**Specifies the MEG-ID using ITU format (ICC - UMC). • **ituCcMeg—**Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC). • **ieee—**Specifies the MEG-ID (Short MA Name) using IEEE Character String format.

Step 4	**addPeerMepId** {**commit** | **flush** | **peerMepConfig** {**macAddress** | **mepInstance** | **peerMepId**}}
	
	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 20`
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 11`
		Adds peer MEP request.
		• **commit—**Commits addPeerMepId.
		• **flush—**Flushes all addPeerMepId commands from queue.
		• **peerMepConfig—**Adds peer mep request.
		• **macAddress—**Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception.
		• **mepInstance—**Specifies the mep instance number.
		• **peerMepId—**Specifies the peer MEP-ID.

Step 5	**addCcAps** {**commit** | **flush** | **mepFunctionalConfig** {**aps** {**enable** | **disable**} | **cc** {**enable** | **disable**} | **mepInstance***mep_instance_number*} | **review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 20`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
		Adds CC/APS configuration request.
		• **commit—**Commits addCcAps.
		• **flush—**Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig—**Adds CC/APS configuration request.
		• **aps—**Specifies APS protocol.
		• **enable—**Enables APS.
		• **disbale—**Disables APS.
		• **cc—**Specifies continuity check.

	Command or Action	Purpose
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps`	• **enable**—Enables CC. • **disbale**—Disables CC. • **mepInstance**—Specifies the mep instance number. • *mep_instance_number*—MEP instance number.

Step 6	**createMep createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level***level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 21`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepId 14`
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 5`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg`
	`UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112`
		Creates MEP configuration.
		• **mepinstance**—Specifies the MEP instance number.
		• **mode**—Specifies the mode of the MEP instance.
		• **mep**—Specifies the maintenance entity end point.
		• **mip**—Specifies the maintenance entity intermediate point.
		• **direction**—Selects the direction of the MEP.
		• **up**—Specifies an Up MEP - monitoring egress OAM and traffic on residence port.
		• **down**—Specifies a Down MEP - monitoring ingress OAM and traffic on residence port.
		• **domain**—Selects the domain of the MEP.
		• **port**—Specifies a MEP in the Port Domain. Flow Instance is a Port.
		• **evc**—Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created.
		• **vlan**—Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created.
		• **flowId**—Specifies the flow related to the MEP.
		• **vid**—In case the MEP is a port Up-MEP or a EVC customer MIP the VID must be given.
		• **level**—Specifies the MEG level of the MEP.
		• *level_number*—MEG level number.
		• **residencePort**—Specifies the port monitored by MEP.
		• *port_number*—Residence port number.
		• **mepId**—Specifies MEP ID.
		• *id_number*—MEP ID number.
		• **megdomain**—Specifies the maintenance domain configuration.
		• **maName**—Specifies the ITU/IEEE MEG-ID(short MA name)

	Command or Action	Purpose
		• *ma_name*—Short MA name. • **megIdFormat—**Selects the MEG ID format. • **ituMeg—**Specifies the MEG-ID using ITU format (ICC - UMC). • **ituCcMeg—**Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC). • **ieee—**Specifies the MEG-ID (Short MA Name) using IEEE Character String format.

Step 7	**addPeerMepId** {**commit** | **flush** | **peerMepConfig** {**macAddress** | **mepInstance** | **peerMepId**}}
	
	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 21`
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 13`
		Adds peer MEP request.
		• **commit—**Commits addPeerMepId.
		• **flush—**Flushes all addPeerMepId commands from queue.
		• **peerMepConfig—**Adds peer mep request.
		• **macAddress—**Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception.
		• **mepInstance—**Specifies the mep instance number.
		• **peerMepId—**Specifies the peer MEP-ID.

Step 8	**addCcAps** {**commit** | **flush** | **mepFunctionalConfig** {**aps** {**enable** | **disable**} | **cc** {**enable** | **disable**} | **mepInstance***mep_instance_number*} | **review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 21`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps`
		Adds CC/APS configuration request.
		• **commit—**Commits addCcAps.
		• **flush—**Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig—**Adds CC/APS configuration request.
		• **aps—**Specifies APS protocol.
		• **enable—**Enables APS.
		• **disbale—**Disables APS.
		• **cc—**Specifies continuity check.
		• **enable—**Enables CC.
		• **disbale—**Disables CC.
		• **mepInstance—**Specifies the mep instance number.
		• *mep_instance_number*—MEP instance number.

Step 9	**addCcAps review** **Example:** `UCS(ProvisionMepPortType)# addCcAps review`	Displays the configuration.

	Command or Action	Purpose
Step 10	**addCcAps commit** **Example:** `UCS(ProvisionMepPortType)# addCcAps commit`	Sends the configuration to NID.
Step 11	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Configuration Example

The example shows how to create MEP on NID-1:

```
UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 20
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 12
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 4
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg
UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 20
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 11
UCS(ProvisionMepPortType)# addPeerMepId commit

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 20
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps

UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 21
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 14
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 5
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg
UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 21
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 13

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 21
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps

UCS(ProvisionMepPortType)# addCcAps review
UCS(ProvisionMepPortType)# addCcAps commit
UCS(ProvisionMepPortType)# exit
```

Creating MEP on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionMepPortType** **Example:** `UCS# ProvisionMepPortType`	Enters the ProvisionMepPortType mode.

Step 3	**createMep createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level***level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 20`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepId 11`
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 4`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg`
	`UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112`
		Creates MEP configuration.
		• **mepinstance**—Specifies the MEP instance number.
		• **mode**—Specifies the mode of the MEP instance.
		• **mep**—Specifies the maintenance entity end point.
		• **mip**—Specifies the maintenance entity intermediate point.
		• **direction**—Selects the direction of the MEP.
		• **up**—Specifies an Up MEP - monitoring egress OAM and traffic on residence port.
		• **down**—Specifies a Down MEP - monitoring ingress OAM and traffic on residence port.
		• **domain**—Selects the domain of the MEP.
		• **port**—Specifies a MEP in the Port Domain. Flow Instance is a Port.
		• **evc**—Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created.
		• **vlan**—Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created.
		• **flowId**—Specifies the flow related to the MEP.
		• **vid**—In case the MEP is a port Up-MEP or a EVC customer MIP, the VID must be given.
		• **level**—Specifies the MEG level of the MEP.
		• *level_number*—MEG level number.
		• **residencePort**—Specifies the port monitored by MEP.
		• *port_number*—Residence port number.

	Command or Action	Purpose
		• **mepId—**Specifies MEP ID. • *id_number*—MEP ID number. • **megdomain—**Specifies the maintenance domain configuration. • **maName—**Specifies the ITU/IEEE MEG-ID (short MA name). • *ma_name*—Short MA name. • **megIdFormat—**Selects the MEG ID format. • **ituMeg—**Specifies the MEG-ID using ITU format (ICC - UMC). • **ituCcMeg—**Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC). • **ieee—**Specifies the MEG-ID (Short MA Name) using IEEE Character String format.

Step 4	**addPeerMepId** {**commit** | **flush** | **peerMepConfig** {**macAddress** | **mepInstance** | **peerMepId**}}
	
	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 20`
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 12`
		Adds peer MEP request.
		• **commit—**Commits addPeerMepId.
		• **flush—**Flushes all addPeerMepId commands from queue.
		• **peerMepConfig—**Adds peer mep request.
		• **macAddress—**Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception.
		• **mepInstance—**Specifies the mep instance number.
		• **peerMepId—**Specifies the peer MEP-ID.

Step 5	**addCcAps** {**commit** | **flush** | **mepFunctionalConfig** {**aps** {**enable** | **disable**} | **cc** {**enable** | **disable**} | **mepInstance***mep_instance_number*} | **review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 20`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps`
		Adds CC/APS configuration request.
		• **commit—**Commits addCcAps.
		• **flush—**Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig—**Adds CC/APS configuration request.
		• **aps—**Specifies APS protocol.
		• **enable—**Enables APS.
		• **disbale—**Disables APS.
		• **cc—**Specifies continuity check.
		• **enable—**Enables CC.
		• **disbale—**Disables CC.

	Command or Action	Purpose
		• **mepInstance—**Specifies the mep instance number. • *mep_instance_number*—MEP instance number.

Step 6	**createMep createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level***level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 21`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepId 13`
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 5`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg`
	`UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112`
		Creates MEP configuration.
		• **mepinstance—**Specifies the MEP instance number.
		• **mode—**Specifies the mode of the MEP instance.
		• **mep—**Specifies the maintenance entity end point.
		• **mip—**Specifies the maintenance entity intermediate point.
		• **direction—**Selects the direction of the MEP.
		• **up—**Specifies an Up MEP - monitoring egress OAM and traffic on residence port.
		• **down—**Specifies a Down MEP - monitoring ingress OAM and traffic on residence port.
		• **domain—**Selects the domain of the MEP.
		• **port—**Specifies a MEP in the Port Domain. Flow Instance is a Port.
		• **evc—**Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created.
		• **vlan—**Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created.
		• **flowId—**Specifies the flow related to the MEP.
		• **vid—**In case the MEP is a port Up-MEP or a EVC customer MIP the VID must be given.
		• **level—**Specifies the MEG level of the MEP.
		• *level_number*—MEG level number.
		• **residencePort—**Specifies the port monitored by MEP.
		• *port_number*—Residence port number.
		• **mepId—**Specifies MEP ID.
		• *id_number*—MEP ID number.
		• **megdomain—**Specifies the maintenance domain configuration.
		• **maName—**Specifies the ITU/IEEE MEG-ID(short MA name)
		• *ma_name*—Short MA name.
		• **megIdFormat—**Selects the MEG ID format.

	Command or Action	Purpose
		• **ituMeg—**Specifies the MEG-ID using ITU format (ICC - UMC). • **ituCcMeg—**Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC). • **ieee—**Specifies the MEG-ID (Short MA Name) using IEEE Character String format.

Step 7	**addPeerMepId** {**commit** | **flush** | **peerMepConfig** {**macAddress** | **mepInstance** | **peerMepId**}}
	
	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 21`
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 14`
		Adds peer MEP request.
		• **commit—**Commits addPeerMepId.
		• **flush—**Flushes all addPeerMepId commands from queue.
		• **peerMepConfig—**Adds peer mep request.
		• **macAddress—**Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception.
		• **mepInstance—**Specifies the mep instance number.
		• **peerMepId—**Specifies the peer MEP-ID.

Step 8	**addCcAps** {**commit** | **flush** | **mepFunctionalConfig** {**aps** {**enable** | **disable**} | **cc** {**enable** | **disable**} | **mepInstance***mep_instance_number*} | **review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 21`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps`
		Adds CC/APS configuration request.
		• **commit—**Commits addCcAps.
		• **flush—**Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig—**Adds CC/APS configuration request.
		• **aps—**Specifies APS protocol.
		• **enable—**Enables APS.
		• **disbale—**Disables APS.
		• **cc—**Specifies continuity check.
		• **enable—**Enables CC.
		• **disbale—**Disables CC.
		• **mepInstance—**Specifies the mep instance number.
		• *mep_instance_number*—MEP instance number.

Step 9	**addCcAps review** **Example:** `UCS(ProvisionMepPortType)# addCcAps review`	Displays the configuration.

	Command or Action	Purpose
Step 10	**addCcAps commit** **Example:** `UCS(ProvisionMepPortType)# addCcAps commit`	Sends the configuration to NID.
Step 11	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Configuration Example

The example shows how to create MEP on NID2:

```
UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 20
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 11
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 4
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg
UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 20
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 12
UCS(ProvisionMepPortType)# addPeerMepId commit

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 20
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps

UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 21
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 13
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain PORT
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 5
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName nid-nid
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg
UCS(ProvisionMepPortType)# createMep createMepConfig vid 1112

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 21
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 14

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 21
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode uni
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol laps

UCS(ProvisionMepPortType)# addCcAps review
UCS(ProvisionMepPortType)# addCcAps commit
UCS(ProvisionMepPortType)# exit
```

Configuring Bidirectional EPS on NID-2

Before You Begin

- Architecture a1plus1 bidirectional
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

Step 3	**setEpsInstConfig epsConfig** {**epsInst***epsInst_number* | **domain** {**port** | **evc**} | **architecture** {**a1plus1** | **a1for1**} | **workflow** {**inst***inst_number* | **portNo***port_number*} | **protectFlow** {**inst***inst_number* | **portNo***port_number*} | **mepWork***mepWork_number* | **mepProtect***mepProtect_number* | **mepAps***mepAPS_number*}
	
	Example:
	`UCS(EpsPortType)# setEpsInstance epsConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstance epsConfig architecture a1plus1`
	`UCS(EpsPortType)# setEpsInstance epsConfig domain port`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20`
	`UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5`
	`UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4`
		Sets EPS configuration.
		• **epsConfig—**Specifies the EPS configuration.
		• **epsInst—** Specifies the EPS instance.
		• *epsInst_number*—EPS instance number.
		• **domain—**Specifies the domain of the EPS.
		• **port—**Specifies that this EPS is protecting in the port domain.
		• **evc—**Specifies that this EPS is protecting in the EVC domain.
		• **architecture—**Specifies the EPS architecture.
		• **a1plus1—**Specifies that the architecture is 1 plus 1.
		• **a1for1—**Specifies that the architecture is 1 for 1.
		• **workflow—**Specifies the working flow instance for the related EPS.
		• **inst—**Specifies the working flow instance number when not in the port domain.
		• *inst_number*—Working flow instance number.
		• **portNo—**Specifies port ID.
		• *port_number*—Port ID number.
		• **protectFlow—**Specifies the protect flow instance for the related EPS.

	Command or Action	Purpose
		• **inst—**Specifies the protect flow instance number when not in the port domain. • *inst_number*—Protect flow instance number. • **portNo—**Specifies port ID. • *port_number*—Port ID number. • **mepWork—**Specifies working MEP instance. • *mepWork_number*—Working MEP number. • **mepProtect—**Specifies protect MEP instance. • *mepProtect_number*—Protect MEP number. • **mepAps—**Specifies APS MEP instance. • *mepAPS_number*—APS MEP number.

Step 4	**setEpsInstProperties epsInstconfig** {**epsInst***eps_instance_number* | **protectionType** {**uni** | **bi**} | **aps** {**enable** | **disable**} | **revertive** {**enable** | **disable**} | **wtrTime w***time*[**m** | **s**] **holdoff**}
	
	Example:
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig holdoff 1`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType bi`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig revertive enable`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig wtrTime w10s`
		Adds CC/APS configuration request.
		• **epsInst—**Specifies the EPS instance.
		• *ep_instance_number*—EPS instance number.
		• **protectionType—**Specifies the protection type in case of 1plus1.
		• **uni—**Specifies unidirectional.
		• **bi—** Specifies bidirectional.
		• **aps—**Specifies EPS 1+1 unidirectional with APS protection type.
		• **enable—**Enables APS protection.
		• **disable—**Disables APS protection.
		• **revertive—**Specifies revertive EPS.
		• **enable—**Enables revertive EPS.
		• **disable—** Disables revertive EPS.
		• **wtrTime—** Specifies the WTR time.
		• *time*—WTR time in minutes or seconds.
		• **m—** Time in minutes. Valid values are from 5 to 12.
		• **s—** Time in seconds. Valid values are 10 and 30.
		• **holdoff—** Specifies the hold off timer.

	Command or Action	Purpose
Step 5	**setEpsInstProperties review** **Example:** `UCS(EpsPortType)# setEpsInstProperties review`	Displays the configuration.
Step 6	**setEpsInstProperties commit** **Example:** `UCS(EpsPortType)# setEpsInstProperties commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to configure bidirectional EPS on NID-2:

```
UCS(EpsPortType)# setEpsInstance epsConfig epsInst 30
UCS(EpsPortType)# setEpsInstance epsConfig architecture a1plus1
UCS(EpsPortType)# setEpsInstance epsConfig domain port
UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21
UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21
UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20
UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5
UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4

UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30
UCS(EpsPortType)# setEpsInstProperties epsInstConfig holdoff 1
UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType bi
UCS(EpsPortType)# setEpsInstProperties epsInstConfig revertive enable
UCS(EpsPortType)# setEpsInstProperties epsInstConfig wtrTime w10s

UCS(EpsPortType)# setEpsInstProperties review
UCS(EpsPortType)# setEpsInstProperties commit
UCS(EpsPortType)# exit
```

Configuring Bidirectional EPS on NID-1

Before You Begin

- Architecture a1plus1 bidirectional
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

Step 3	**setEpsInstConfig epsConfig** {**epsInst***epsInst_number* | **domain** {**port** | **evc**} | **architecture** {**a1plus1** | **a1for1**} | **workflow** {**inst***inst_number* | **portNo***port_number*} | **protectFlow** {**inst***inst_number* | **portNo***port_number*} | **mepWork***mepWork_number* | **mepProtect***mepProtect_number* | **mepAps***mepAPS_number*}
	
	Example:
	`UCS(EpsPortType)# setEpsInstance epsConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstance epsConfig architecture a1plus1`
	`UCS(EpsPortType)# setEpsInstance epsConfig domain port`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20`
	`UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5`
	`UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4`
		Sets EPS configuration.
		• **epsConfig**—Specifies the EPS configuration.
		• **epsInst**— Specifies the EPS instance.
		• *epsInst_number*—EPS instance number.
		• **domain**—Specifies the domain of the EPS.
		• **port**—Specifies that this EPS is protecting in the port domain.
		• **evc**—Specifies that this EPS is protecting in the EVC domain.
		• **architecture**—Specifies the EPS architecture.
		• **a1plus1**—Specifies that the architecture is 1 plus 1.
		• **a1for1**—Specifies that the architecture is 1 for 1.
		• **workflow**—Specifies the working flow instance for the related EPS.
		• **inst**—Specifies the working flow instance number when not in the port domain.
		• *inst_number*—Working flow instance number.
		• **portNo**—Specifies port ID.
		• *port_number*—Port ID number.
		• **protectFlow**—Specifies the protect flow instance for the related EPS.
		• **inst**—Specifies the protect flow instance number when not in the port domain.
		• *inst_number*—Protect flow instance number.
		• **portNo**—Specifies port ID.
		• *port_number*—Port ID number.
		• **mepWork**—Specifies working MEP instance.
		• *mepWork_number*—Working MEP number.

	Command or Action	Purpose
		• **mepProtect—**Specifies protect MEP instance. • *mepProtect_number*—Protect MEP number. • **mepAps—**Specifies APS MEP instance. • *mepAPS_number*—APS MEP number.

Step 4	**setEpsInstProperties epsInstconfig** {**epsInst** | **protectionType** {**uni** | **bi**} | **aps** {**enable** | **disable**} | **revertive** {**enable** | **disable**} | **wtrTime w***time*[**m** | **s**] **holdoff**}
	
	Example:
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig holdoff 1`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType bi`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig revertive enable`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig wtrTime w10s`
		Adds CC/APS configuration request.
		• **epsInst—**Specifies the EPS instance.
		• *ep_instance_number*—EPS instance number.
		• **protectionType—**Specifies the protection type in case of 1plus1.
		• **uni—**Specifies unidirectional.
		• **bi—** Specifies bidirectional.
		• **aps—**Specifies EPS 1+1 unidirectional with APS protection type.
		• **enable—**Enables APS protection.
		• **disable—**Disables APS protection.
		• **revertive—**Specifies revertive EPS.
		• **enable—**Enables revertive EPS.
		• **disable—** Disables revertive EPS.
		• **wtrTime—** Specifies the WTR time.
		• *time*—WTR time in minutes or seconds.
		• **m—** Time in minutes. Valid values are from 5 to 12.
		• **s—** Time in seconds. Valid values are 10 and 30.
		• **holdoff—** Specifies the hold off timer.

Step 5	**setEpsInstProperties review** **Example:** `UCS(EpsPortType)# setEpsInstProperties review`	Displays the configuration.
Step 6	**setEpsInstProperties commit** **Example:** `UCS(EpsPortType)# setEpsInstProperties commit`	Sends the configuration to NID.

	Command or Action	Purpose
Step 7	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to configure bidirectional EPS on NID-1:

```
UCS(EpsPortType)# setEpsInstance epsConfig epsInst 30
UCS(EpsPortType)# setEpsInstance epsConfig architecture a1plus1
UCS(EpsPortType)# setEpsInstance epsConfig domain port
UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21
UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21
UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20
UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5
UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4

UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30
UCS(EpsPortType)# setEpsInstProperties epsInstConfig holdoff 1
UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType bi
UCS(EpsPortType)# setEpsInstProperties epsInstConfig revertive enable
UCS(EpsPortType)# setEpsInstProperties epsInstConfig wtrTime w10s

UCS(EpsPortType)# setEpsInstProperties review
UCS(EpsPortType)# setEpsInstProperties commit
UCS(EpsPortType)# exit
```

Configuring Unidirectional EPS on NID-2

Before You Begin

- Architecture a1plus1 unidirectional aps enable
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

	Command or Action	Purpose

Step 3	**setEpsInstConfig epsConfig** {**epsInst***epsInst_number*| **domain** {**port** | **evc**} | **architecture** {**a1plus1** | **a1for1**} | **workflow** {**inst***inst_number* | **portNo***port_number*} | **protectFlow** {**inst***inst_number* | **portNo***port_number*} | **mepWork***mepWork_number* | **mepProtect***mepProtect_number* | **mepAps***mepAPS_number*}
	
	Example:
	`UCS(EpsPortType)# setEpsInstance epsConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstance epsConfig architecture a1plus1`
	`UCS(EpsPortType)# setEpsInstance epsConfig domain port`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20`
	`UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5`
	`UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4`
		Sets EPS configuration.
		• **epsConfig—**Specifies the EPS configuration.
		• **epsInst—** Specifies the EPS instance.
		• *epsInst_number*—EPS instance number.
		• **domain—**Specifies the domain of the EPS.
		• **port—**Specifies that this EPS is protecting in the port domain.
		• **evc—**Specifies that this EPS is protecting in the EVC domain.
		• **architecture—**Specifies the EPS architecture.
		• **a1plus1—**Specifies that the architecture is 1 plus 1.
		• **a1for1—**Specifies that the architecture is 1 for 1.
		• **workflow—**Specifies the working flow instance for the related EPS.
		• **inst—**Specifies the working flow instance number when not in the port domain.
		• *inst_number*—Working flow instance number.
		• **portNo—**Specifies port ID.
		• *port_number*—Port ID number.
		• **protectFlow—**Specifies the protect flow instance for the related EPS.
		• **inst—**Specifies the protect flow instance number when not in the port domain.
		• *inst_number*—Protect flow instance number.
		• **portNo—**Specifies port ID.
		• *port_number*—Port ID number.
		• **mepWork—**Specifies working MEP instance.
		• *mepWork_number*—Working MEP number.
		• **mepProtect—**Specifies protect MEP instance.
		• *mepProtect_number*—Protect MEP number.
		• **mepAps—**Specifies APS MEP instance.
		• *mepAPS_number*—APS MEP number.

Step 4	**setEpsInstProperties epsInstconfig** {**epsInst***eps_instance_number* | **protectionType** {**uni**	Adds CC/APS configuration request. • **epsInst—**Specifies the EPS instance.

	Command or Action	Purpose

	| **bi**} | **aps** {**enable** | **disable**} | **revertive** {**enable** | **disable**} | **wtrTime w***time*[**m** | **s**] **holdoff**}
	
	Example:
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig aps enable`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType uni`
		• *eps_instance_number*—EPS instance number.
		• **protectionType—**Specifies the protection type in case of 1plus1.
		• **uni—**Specifies unidirectional.
		• **bi—** Specifies bidirectional.
		• **aps—**Specifies EPS 1+1 unidirectional with APS protection type.
		• **enable—**Enables APS protection.
		• **disable—**Disables APS protection.
		• **revertive—**Specifies revertive EPS.
		• **enable—**Enables revertive EPS.
		• **disable—** Disables revertive EPS.
		• **wtrTime—** Specifies the WTR time.
		• *time*—WTR time in minutes or seconds.
		• **m—** Time in minutes. Valid values are from 5 to 12.
		• **s—** Time in seconds. Valid values are 10 and 30.
		• **holdoff—** Specifies the hold off timer.

Step 5	**setEpsInstProperties review** **Example:** `UCS(EpsPortType)# setEpsInstProperties review`	Displays the configuration.
Step 6	**setEpsInstProperties commit** **Example:** `UCS(EpsPortType)# setEpsInstProperties commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to configure unidirectional EPS on NID-2:

```
UCS(EpsPotType)# setEpsInstance epsConfig epsInst 30
UCS(EpsPortType)# setEpsInstance epsConfig architecture a1plus1
UCS(EpsPortType)# setEpsInstance epsConfig domain port
UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21
UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21
```

```
UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20
UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5
UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4

UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30
UCS(EpsPortType)# setEpsInstProperties epsInstConfig aps enable
UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType uni

UCS(EpsPortType)# setEpsInstProperties review
UCS(EpsPortType)# setEpsInstProperties commit
UCS(EpsPortType)# exit
```

Configuring Bidirectional EPS on NID-2

Before You Begin

- Architecture a1for1 bidirectional
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

Step 3	**setEpsInstConfig epsConfig** {**epsInst***epsInst_number* | **domain** {**port** | **evc**} | **architecture** {**a1plus1** | **a1for1**} | **workflow** {**inst***inst_number* | **portNo***port_number*} | **protectFlow** {**inst***inst_number* | **portNo***port_number*} | **mepWork***mepWork_number* | **mepProtect***mepProtect_number* | **mepAps***mepAPS_number*}
	
	Example:
	`UCS(EpsPortType)# setEpsInstance epsConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstance epsConfig architecture a1for1`
	`UCS(EpsPortType)# setEpsInstance epsConfig domain port`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21`
	`UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20`
	`UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5`
		Sets EPS configuration.
		• **epsConfig—**Specifies the EPS configuration.
		• **epsInst—** Specifies the EPS instance.
		• *epsInst_number*—EPS instance number.
		• **domain—**Specifies the domain of the EPS.
		• **port—**Specifies that this EPS is protecting in the port domain.
		• **evc—**Specifies that this EPS is protecting in the EVC domain.
		• **architecture—**Specifies the EPS architecture.
		• **a1plus1—**Specifies that the architecture is 1 plus 1.
		• **a1for1—**Specifies that the architecture is 1 for 1.
		• **workflow—**Specifies the working flow instance for the related EPS.
		• **inst—**Specifies the working flow instance number when not in the port domain.

	Command or Action	Purpose
	`UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4`	• *inst_number*—Working flow instance number. • **portNo—**Specifies port ID. • *port_number*—Port ID number. • **protectFlow—**Specifies the protect flow instance for the related EPS. • **inst—**Specifies the protect flow instance number when not in the port domain. • *inst_number*—Protect flow instance number. • **portNo—**Specifies port ID. • *port_number*—Port ID number. • **mepWork—**Specifies working MEP instance. • *mepWork_number*—Working MEP number. • **mepProtect—**Specifies protect MEP instance. • *mepProtect_number*—Protect MEP number. • **mepAps—**Specifies APS MEP instance. • *mepAPS_number*—APS MEP number.

Step 4	**setEpsInstProperties epsInstconfig** {**epsInst***eps_instance_number* | **protectionType** {**uni** | **bi**} | **aps** {**enable** | **disable**} | **revertive** {**enable** | **disable**} | **wtrTime w***time*[**m** | **s**] **holdoff**}
	
	Example:
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType bi`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig revertive enable`
	`UCS(EpsPortType)# setEpsInstProperties epsInstConfig wtrTime w10s`
		Adds CC/APS configuration request.
		• **epsInst—**Specifies the EPS instance.
		• *eps_instance_number*—EPS instance number.
		• **protectionType—**Specifies the protection type in case of 1plus1.
		• **uni—**Specifies unidirectional.
		• **bi—** Specifies bidirectional.
		• **aps—**Specifies EPS 1+1 unidirectional with APS protection type.
		• **enable—**Enables APS protection.
		• **disable—**Disables APS protection.
		• **revertive—**Specifies revertive EPS.
		• **enable—**Enables revertive EPS.
		• **disable—** Disables revertive EPS.
		• **wtrTime—** Specifies the WTR time.
		• *time*—WTR time in minutes or seconds.

	Command or Action	Purpose
		• **m—** Time in minutes. Valid values are from 5 to 12. • **s—** Time in seconds. Valid values are 10 and 30. • **holdoff—** Specifies the hold off timer.
Step 5	**setEpsInstProperties review** **Example:** `UCS(EpsPortType)# setEpsInstProperties review`	Displays the configuration.
Step 6	**setEpsInstProperties commit** **Example:** `UCS(EpsPortType)# setEpsInstProperties commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to configure bidirectional EPS on NID-2:

```
UCS(EpsPortType)# setEpsInstance epsConfig epsInst 30
UCS(EpsPortType)# setEpsInstance epsConfig architecture a1for1
UCS(EpsPortType)# setEpsInstance epsConfig domain port
UCS(EpsPortType)# setEpsInstance epsConfig mepAps 21
UCS(EpsPortType)# setEpsInstance epsConfig mepProtect 21
UCS(EpsPortType)# setEpsInstance epsConfig mepWork 20
UCS(EpsPortType)# setEpsInstance epsConfig protectFlow portNo 5
UCS(EpsPortType)# setEpsInstance epsConfig workFlow portNo 4

UCS(EpsPortType)# setEpsInstProperties epsInstConfig epsInst 30
UCS(EpsPortType)# setEpsInstProperties epsInstConfig protectionType bi
UCS(EpsPortType)# setEpsInstProperties epsInstConfig revertive enable
UCS(EpsPortType)# setEpsInstProperties epsInstConfig wtrTime w10s

UCS(EpsPortType)# setEpsInstProperties review
UCS(EpsPortType)# setEpsInstProperties commit
UCS(EpsPortType)# exit
```

Displaying EPS

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.
Step 3	**getEpsInstProperties epsRequest epsInst***eps_instance_number* **Example:** `UCS(EpsPortType)# getEpsInstProperties epsRequest epsInst 30`	Sets EPS configuration. • **epsRequest—**Specifies EPS get request parameter. • **epsInst—** Specifies the EPS instance. • *eps_instance_number*—EPS instance number.
Step 4	**getEpsInstance epsRequest** {**epsInst***eps_instance_number* **Example:** `UCS(EpsPortType)# getEpsInstance epsRequest epsInst 30`	Adds CC/APS configuration request. • **epsRequest—**Specifies EPS get request parameter. • **epsInst—** Specifies the EPS instance. • *eps_instance_number*—EPS instance number.
Step 5	**showEpsConfig showEpsReq epsInstList***eps_instance_list_number* **Example:** `UCS(EpsPortType)# showEpsConfig showEpsReq epsInstList 30`	Adds CC/APS configuration request. • **showEpsReq—**Displays the EPS configuration. • **epsInstList—**Specifies the EPS instance list. • *eps_instance_list_number*—EPS instance list number.
Step 6	**showEpsState showEpsReq epsInstList***eps_instance_list_number* **Example:** `UCS(EpsPortType)# showEpsState showEpsReq epsInstList 30`	Adds CC/APS configuration request. • **showEpsReq—**Displays EPS request parameter. • **epsInstList—**Specifies the EPS instance list. • *eps_instance_list_number*—EPS instance list number. The valid value are from 1-100.
Step 7	**setEpsInstProperties review** **Example:** `UCS(EpsPortType)# setEpsInstProperties review`	Displays the configuration.

	Command or Action	Purpose
Step 8	**setEpsInstProperties commit** **Example:** `UCS(EpsPortType)# setEpsInstProperties commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to display EPS:

```
UCS(EpsPortType)# getEpsInstProperties epsRequest epsInst 30
UCS(EpsPortType)# getEpsInstance epsRequest epsInst 30
UCS(EpsPortType)# showEpsConfig showEpsReq epsInstList 30
UCS(EpsPortType)# showEpsState showEpsReq epsInstList 30
UCS(EpsPortType)# setEpsInstProperties review
UCS(EpsPortType)# setEpsInstProperties commit
UCS(EpsPortType)# exit
```

Clearing EPS Wait-To-Restore Timer

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.
Step 3	**clearEpsWtr clearEps epsInst***eps_instance_number* **Example:** `UCS(EpsPortType)# clearEpsWtr clearEps epsInst 30`	Sets EPS configuration. • **clearEps—**Specifies clear EPS WTR. • **epsInst—** Specifies the EPS instance. • *eps_instance_number*—EPS instance number.
Step 4	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to clear EPS:

```
UCS(EpsPortType)# clearEpsWtr clearEps epsInst 30
UCS(EpsPortType)# exit
```

Updating EPS

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

Step 3	**updateEpsInstance epsCommand** {**epsInst***epsInst_number* | **command** {**lockout** | **forced** | **manualp** | **manualw** | **exercise** | **freeze** | **localLockout**}}
	
	Example:
	`UCS(EpsPortType)# updateEpsInstance epsCommand epsInst 1`
	
	Example:
	`UCS(EpsPortType)# updateEpsInstance epsCommand epsInst 1`
	`UCS(EpsPortType)# updateEpsInstance epsCommand command exercise`
	`UCS(EpsPortType)# updateEpsInstance epsCommand command forced`
	`UCS(EpsPortType)# updateEpsInstance epsCommand command freeze`
	`UCS(EpsPortType)# updateEpsInstance epsCommand command localLockout`
	`UCS(EpsPortType)# updateEpsInstance epsCommand command lockout`
	`UCS(EpsPortType)# updateEpsInstance epsCommand command manualp`
	`UCS(EpsPortType)# updateEpsInstance epsCommand command manualw`
		Use only one of the following commands, as required: Sets EPS configuration.
		
		• **epsCommand—**Specifies the EPS command configuration.
		• **epsInst—** Specifies the EPS instance.
		• *epsInst_number*—EPS instance number.
		• **command—**Specifies the EPS commands.
		• **lockout—**Locks out of protection.
		• **forced—**Forces switching of normal traffic to protection.
		• **manualp—**Manually switches normal traffic to protection.
		• **manualw—**Manually switches normal traffic to working.
		• **exercise—**Specifies the exercise signal.
		• **freeze—**Specifies local freezing of EPS.
		• **localLockout—**Specifies local lockout of EPS.
		• **clear—**Clears EPS commands.

	Command or Action	Purpose
Step 4	**updateEpsInstance review** **Example:** `UCS(EpsPortType)# updateEpsInstance review`	Displays the configuration.
Step 5	**updateEpsInstance commit** **Example:** `UCS(EpsPortType)# updateEpsInstance commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to update EPS:

```
UCS(EpsPortType)# updateEpsInstance epsCommand epsInst 1
```

Use only one of the following commands, as required:

```
UCS(EpsPortType)# updateEpsInstance epsCommand command exercise
UCS(EpsPortType)# updateEpsInstance epsCommand command forced
UCS(EpsPortType)# updateEpsInstance epsCommand command freeze
UCS(EpsPortType)# updateEpsInstance epsCommand command localLockout
UCS(EpsPortType)# updateEpsInstance epsCommand command lockout
UCS(EpsPortType)# updateEpsInstance epsCommand command manualp
UCS(EpsPortType)# updateEpsInstance epsCommand command manualw
UCS(EpsPortType)# updateEpsInstance review
UCS(EpsPortType)# updateEpsInstance commit
UCS(EpsPortType)# exit
```

Deleting EPS

Before You Begin

- Architecture a1plus1 bidirectional
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

Step 3	**deleteEps deleteEpsConfig** {**epsInst***eps_instance_number* | **delete** {**eps** | **command** | **holdoff** | **revertive**}}
	
	Example:
	`UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30`
	`UCS(EpsPortType)# deleteEps deleteEpsConfig delete eps`
		Adds CC/APS configuration request.
		• **deleteEpsConfig—**Deletes EPS configuration.
		• **epsInst—**Specifies the EPS instance.
		• *ep_instance_number***—**EPS instance number.
		• **delete—**Deletes the configuration.
		• **eps—**Deletes EPS instance.
		• **command—**Deletes EPS commands.
		• **holdoff—**Clears hold off timer.
		• **revertive—**Disables revertive EPS.

Step 4	**deleteEps review** **Example:** `UCS(EpsPortType)# deleteEps review`	Displays the configuration.
Step 5	**deleteEps commit** **Example:** `UCS(EpsPortType)# deleteEps commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to delete EPS:

```
UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30
UCS(EpsPortType)# deleteEps deleteEpsConfig delete eps

UCS(EpsPortType)# deleteEps review
UCS(EpsPortType)# deleteEps commit
UCS(EpsPortType)# exit
```

Deleting EPS Command

Before You Begin

- Architecture a1plus1 bidirectional
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

Step 3	**deleteEps deleteEpsConfig** {**epsInst***eps_instance_number* | **delete** {**eps** | **command** | **holdoff** | **revertive**}}
	
	Example:
	`UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30`
	`UCS(EpsPortType)# deleteEps deleteEpsConfig delete command`
		Adds CC/APS configuration request.
		• **deleteEpsConfig—**Deletes EPS configuration.
		• **epsInst—**Specifies the EPS instance.
		• *ep_instance_number*—EPS instance number.
		• **delete—**Deletes the configuration.
		• **eps—**Deletes EPS instance.
		• **command—**Deletes EPS commands.
		• **holdoff—**Clears hold off timer.
		• **revertive—**Disables revertive EPS.

Step 4	**deleteEps review** **Example:** `UCS(EpsPortType)# deleteEps review`	Displays the configuration.
Step 5	**deleteEps commit** **Example:** `UCS(EpsPortType)# deleteEps commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode..

Configuration Example

The example shows how to delete EPS command:

```
UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30
UCS(EpsPortType)# deleteEps deleteEpsConfig delete command

UCS(EpsPortType)# deleteEps review
UCS(EpsPortType)# deleteEps commit
UCS(EpsPortType)# exit
```

Deleting EPS Hold Off Timer

Before You Begin

- Architecture a1plus1 bidirectional
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

Step 3	**deleteEps deleteEpsConfig** {**epsInst***eps_instance_number* | **delete** {**eps** | **command** | **holdoff** | **revertive**}}
	
	Example:
	`UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30`
	`UCS(EpsPortType)# deleteEps deleteEpsConfig delete holdoff`
		Adds CC/APS configuration request.
		• **deleteEpsConfig—**Deletes EPS configuration.
		• **epsInst—**Specifies the EPS instance.
		• *ep_instance_number*—EPS instance number.
		• **delete—**Deletes the configuration.
		• **eps—**Deletes EPS instance.
		• **command—**Deletes EPS commands.
		• **holdoff—**Clears hold off timer.
		• **revertive—**Disables revertive EPS.

	Command or Action	Purpose
Step 4	**deleteEps review** **Example:** `UCS(EpsPortType)# deleteEps review`	Displays the configuration.
Step 5	**deleteEps commit** **Example:** `UCS(EpsPortType)# deleteEps commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to delete EPS hold off timer:

```
UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30
UCS(EpsPortType)# deleteEps deleteEpsConfig delete holdoff

UCS(EpsPortType)# deleteEps review
UCS(EpsPortType)# deleteEps commit
UCS(EpsPortType)# exit
```

Deleting EPS Revertive Timer

Before You Begin

- Architecture a1plus1 bidirectional
- Domain port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**EpsPortType** **Example:** `UCS# EpsPortType`	Enters the EpsPortType mode.

	Command or Action	Purpose

Step 3	**deleteEps deleteEpsConfig** {**epsInst***eps_instance_number* | **delete** {**eps** | **command** | **holdoff** | **revertive**}}
	
	Example:
	`UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30`
	`UCS(EpsPortType)# deleteEps deleteEpsConfig delete revertive`
		Adds CC/APS configuration request.
		• **deleteEpsConfig—**Deletes EPS configuration.
		• **epsInst—**Specifies the EPS instance.
		• *ep_instance_number*—EPS instance number.
		• **delete—**Deletes the configuration.
		• **eps—**Deletes EPS instance.
		• **command—**Deletes EPS commands.
		• **holdoff—**Clears hold off timer.
		• **revertive—**Disables revertive EPS.

Step 4	**deleteEps review** **Example:** `UCS(EpsPortType)# deleteEps review`	Displays the configuration.
Step 5	**deleteEps commit** **Example:** `UCS(EpsPortType)# deleteEps commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(EpsPortType)# exit`	Exits the EpsPortType mode.

Configuration Example

The example shows how to delete EPS revertive timer:

```
UCS(EpsPortType)# deleteEps deleteEpsConfig epsInst 30
UCS(EpsPortType)# deleteEps deleteEpsConfig delete revertive

UCS(EpsPortType)# deleteEps review
UCS(EpsPortType)# deleteEps commit
UCS(EpsPortType)# exit
```

Verifying EPS

Use the following commands to verify the EPS status on the UCS controller.

- **showEpsConfig showEpsReq epsInstList 1**

This command displays the EPS configuration status on the NID. The following is a sample output from the command:

```
UCS(EpsPortType)# showEpsConfig showEpsReq epsInstList 1
UCS(EpsPortType)# showEpsConfig review
```

```
Commands in queue:
 showEpsConfig showEpsReq epsInstList 1
```

```
UCS(EpsPortType)# showEpsConfig commit
```

```
Clearing Socket 5
xpinfo->value : 1Clearing Socket 5
ShowEpsConfig_Output.epsInfo.epsInstance[0].epsInst = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.domain.t = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.domain.u.port =
'Port'
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.architecture.t =
1
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.architecture.u.a1plus1
 = '1plus1'
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.workFlow.t = 2
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.workFlow.u.portNo
 = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.protectFlow.t = 2
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.protectFlow.u.portNo
 = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.mepWork = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.mepProtect = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].config.mepAps = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.protectionType.t
 = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.protectionType.u.uni
 = 'unidirectional'
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.revertive.t =
 2
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.revertive.u.disable
 = 'Disable'
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.aps.t = 2
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.aps.u.disable
 = 'Disable'
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.wtrTime.t = 1
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.wtrTime.u.w10m
 = ''
ShowEpsConfig_Output.epsInfo.epsInstance[0].instConfig.holdoff = 100
ShowEpsConfig_Output.epsInfo.epsInstance[0].command.t = 2
ShowEpsConfig_Output.epsInfo.epsInstance[0].command.u.forced = 'forced'

 ShowEpsConfig Commit Success!!!
```

- **showEpsState showEpsReq epsInstList 1**

  This command displays the EPS status on the NID. The following is a sample output from the command:

```
UCS(EpsPortType)# showEpsState showEpsReq epsInstList 1
UCS(EpsPortType)# showEpsState review
```

```
Commands in queue:
 showEpsState showEpsReq epsInstList 1
```

```
UCS(EpsPortType)# showEpsState commit
```

```
Clearing Socket 5 Clearing Socket 5
ShowEpsState_Output.epsStateInfo.epsInst[0].epsInst = 1
ShowEpsState_Output.epsStateInfo.epsInst[0].protectionState = 'Disable'
ShowEpsState_Output.epsStateInfo.epsInst[0].wFlow = 'Ok'
ShowEpsState_Output.epsStateInfo.epsInst[0].pFlow = 'Ok'
ShowEpsState_Output.epsStateInfo.epsInst[0].transmitAps = 'LO'
ShowEpsState_Output.epsStateInfo.epsInst[0].receiveAps = 'LO'
ShowEpsState_Output.epsStateInfo.epsInst[0].architectureMismatch =
true
ShowEpsState_Output.epsStateInfo.epsInst[0].APSOnWorking = true
ShowEpsState_Output.epsStateInfo.epsInst[0].switchingIncomplete = true
ShowEpsState_Output.epsStateInfo.epsInst[0].noAPSReceived = true
ShowEpsState_Output.epsStateInfo.epsInst[0].txApsRe = 1
ShowEpsState_Output.epsStateInfo.epsInst[0].txApsBr = 2200564160
ShowEpsState_Output.epsStateInfo.epsInst[0].rxApsRe = 2200566368
ShowEpsState_Output.epsStateInfo.epsInst[0].rxApsBr = 2222748384

 ShowEpsState Commit Success!!!
```

CHAPTER 20

Configuring ERPS

This document describes the Ethernet Ring Protection Switching (ERPS) feature and configuration steps to implement protection switching mechanisms for Ethernet layer ring topologies.

Prerequisites for Configuring ERPS

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.

Restrictions for Configuring ERPS

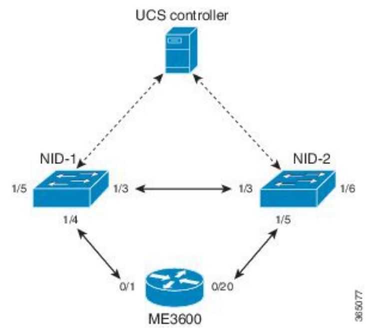
- Maintenance End Point (MEP) is not supported on Ethernet Virtual Connections (EVC) bridge domain.
- MEP domain for control VLAN is only on Port or VLAN.

Information About ERPS

The ITU-T G.8032 ERPS feature implements protection switching mechanisms for Ethernet layer ring topologies. This feature uses the G.8032 Ethernet Ring Protection (ERP) protocol, defined in ITU-T G.8032, to provide protection for Ethernet traffic in a ring topology, while ensuring that no loops are within the ring at the Ethernet layer. The loops are prevented by blocking traffic on either a predetermined link or a failed link.

The following figure shows the topology used for provisioning ERPS on NID-1 and NID-2 using the UCS Controller.

Figure 12: ERPS Topology



How to Provision ERPS

Creating VLAN on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**createVlanCommand createVlanReqvlan_list***vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 2000`	Creates VLAN list.

Step 4	**modifySwPort modifySWPortConfig interface***interface_id* | **mode** [**access Vlan***vlan_number*] | **trunk** {**allowed** | **native**}
	
	Example:
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 3`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2000`
		Modifies the switchport configuration.
		• **interface**—Selects the interface to be configured.
		• *Interface Id*—Specifies the interface ID.
		• **mode**—Specifies the mode of operation.
		• **access**—Sets mode to ACCESS unconditionally.

	Command or Action	Purpose
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 4` `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1` `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan-list 2000`	• **vlan—**Sets VLAN when interface is in access mode. • *vlan_number*—Specifies the VLAN number. • **trunk—**Sets mode to TRUNK unconditionally. • **allowed—**Sets allowed VLAN characteristics when interface is in trunk mode. • **native—**Sets native VLAN.
Step 5	**modifySwPort review** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort review`	Displays the configuration.
Step 6	**modifySwPort commit** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

The example shows how to create VLAN on NID-1:

```
UCS(config-controller-ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list
 2000

UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 3
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 2000

UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 4
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 2000

UCS(ProvisionPortVlanPortType)# modifySwPort review
UCS(ProvisionPortVlanPortType)# modifySwPort commit
UCS(ProvisionPortVlanPortType)# exit
```

Creating MEP on Port 1 of NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** `UCS# ProvisionMepPortType`	Enters the ProvisionMepPortType mode.

Step 3	**createMep createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level** *level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 100`
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName ERPS-1`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepId 100`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 3`
	`UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000`
		Creates MEP configuration.
		• **mepinstance—**Specifies the MEP instance number.
		• **mode—**Specifies the mode of the MEP instance.
		• **mep—**Specifies the maintenance entity end point.
		• **mip—**Specifies the maintenance entity intermediate point.
		• **direction—**Selects the direction of the MEP.
		• **up—**Specifies an Up MEP - monitoring egress OAM and traffic on residence port.
		• **down—**Specifies a Down MEP - monitoring ingress OAM and traffic on residence port.
		• **domain—**Selects the domain of the MEP.
		• **port—**Specifies a MEP in the Port Domain. Flow Instance is a Port.
		• **evc—**Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created.
		• **vlan—**Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created.
		• **flowId—**Specifies the flow related to the MEP.
		• **vid—**In case the MEP is a port Up-MEP or a EVC customer MIP the VID must be given.
		• **level—**Specifies the MEG level of the MEP.
		• *level_number*—MEG level number.
		• **residencePort—**Specifies the port monitored by MEP.
		• *port_number*—Residence port number.

	Command or Action	Purpose
		• **mepId—**Specifies MEP ID. • *id_number*—MEP ID number. • **megdomain—**Specifies the maintenance domain configuration. • **maName—**Specifies the ITU/IEEE MEG-ID (short MA name). • *ma_name*—Short MA name. • **megIdFormat—**Selects the MEG ID format. • **ituMeg—**Specifies the MEG-ID using ITU format (ICC - UMC). • **ituCcMeg—**Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC). • **ieee—**Specifies the MEG-ID (Short MA Name) using IEEE Character String format.

Step 4	**addPeerMepId commit** | **flush** | **peerMepConfig** {**macAddress** | **mepInstance** | **peerMepId**}
	
	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 100`
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 101`
		Adds peer MEP request.
		• **commit—**Commits addPeerMepId.
		• **flush—**Flushes all addPeerMepId commands from queue.
		• **peerMepConfig—**Adds peer mep request.
		• **macAddress—**Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception.
		• **mepInstance—**Specifies the mep instance number.
		• **peerMepId—**Specifies the peer MEP-ID.

Step 5	**addCcAps** {**commit** | **flush** | **mepFunctionalConfig** {**aps** {**enable** | **disable**} | **cc** {**enable** | **disable**} | **mepInstance** *mep_instance_number*} | **review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 100`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps octet 1`
		Adds CC/APS configuration request.
		• **commit—**Commits addCcAps.
		• **flush—**Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig—**Adds CC/APS configuration request.
		• **aps—**Specifies APS protocol.
		• **enable—**Enables APS.
		• **disbale—**Disables APS.
		• **cc—**Specifies continuity check.
		• **enable—**Enables CC.

	Command or Action	Purpose
		• **disbale**—Disables CC. • **mepInstance**—Specifies the mep instance number. • *mep_instance_number*—MEP instance number.
Step 6	**addCcAps review** **Example:** `UCS(ProvisionMepPortType)# addCcAps review`	Displays the configuration.
Step 7	**addCcAps commit** **Example:** `UCS(ProvisionMepPortType)# addCcAps commit`	Sends the configuration to NID.
Step 8	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Configuration Example

The example shows how to create MEP on port 1 of NID-1:

```
UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 100
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName ERPS-1
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 100
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 3
UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 100
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 101

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 100
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps
octet 1

UCS(ProvisionMepPortType)# addCcAps review
UCS(ProvisionMepPortType)# addCcAps commit
UCS(ProvisionMepPortType)# exit
```

Creating MEP on Port 2 of NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMepPortType** **Example:** `UCS# ProvisionMepPortType`	Enters the ProvisionMepPortType mode.

Step 3	**createMep createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level***level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 99`
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName W-N-V2000`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ieee name W-N-V2000`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepId 101`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 4`
	`UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000`
		Creates MEP configuration.
		• **mepinstance—**Specifies the MEP instance number.
		• **mode—**Specifies the mode of the MEP instance.
		• **mep—**Specifies the maintenance entity end point.
		• **mip—**Specifies the maintenance entity intermediate point.
		• **direction—**Selects the direction of the MEP.
		• **up—**Specifies an Up MEP - monitoring egress OAM and traffic on residence port.
		• **down—**Specifies a Down MEP - monitoring ingress OAM and traffic on residence port.
		• **domain—**Selects the domain of the MEP.
		• **port—**Specifies a MEP in the Port Domain. Flow Instance is a Port.
		• **evc—**Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created.
		• **vlan—**Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created.
		• **flowId—**Specifies the flow related to the MEP.
		• **vid—**In case the MEP is a port Up-MEP or a EVC customer MIP the VID must be given.
		• **level—**Specifies the MEG level of the MEP.
		• *level_number*—MEG level number.
		• **residencePort—**Specifies the port monitored by MEP.
		• *port_number*—Residence port number.

	Command or Action	Purpose
		• **mepId—**Specifies MEP ID. • *id_number*—MEP ID number. • **megdomain—**Specifies the maintenance domain configuration. • **maName—**Specifies the ITU/IEEE MEG-ID (short MA name). • *ma_name*—Short MA name. • **megIdFormat—**Selects the MEG ID format. • **ituMeg—**Specifies the MEG-ID using ITU format (ICC - UMC). • **ituCcMeg—**Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC). • **ieee—**Specifies the MEG-ID (Short MA Name) using IEEE Character String format.

Step 4	**addPeerMepId commit | flush | peerMepConfig** {**macAddress | mepInstance | peerMepId**}
	
	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 99`
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 102`
		Adds peer MEP request.
		• **commit—**Commits addPeerMepId.
		• **flush—**Flushes all addPeerMepId commands from queue.
		• **peerMepConfig—**Adds peer mep request.
		• **macAddress—**Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception.
		• **mepInstance—**Specifies the mep instance number.
		• **peerMepId—**Specifies the peer MEP-ID.

Step 5	**addCcAps** {**commit | flush | mepFunctionalConfig** {**aps** {**enable | disable**} **| cc** {**enable | disable**} **| mepInstance***mep_instance_number*} **| review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 99`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps octet 1`
		Adds CC/APS configuration request.
		• **commit—**Commits addCcAps.
		• **flush—**Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig—**Adds CC/APS configuration request.
		• **aps—**Specifies APS protocol.
		• **enable—**Enables APS.
		• **disbale—**Disables APS.
		• **cc—**Specifies continuity check.
		• **enable—**Enables CC.

	Command or Action	Purpose
		• **disbale—**Disables CC. • **mepInstance—**Specifies the mep instance number. • *mep_instance_number*—MEP instance number.
Step 6	**addCcAps review** **Example:** `UCS(ProvisionMepPortType)# addCcAps review`	Displays the configuration.
Step 7	**addCcAps commit** **Example:** `UCS(ProvisionMepPortType)# addCcAps commit`	Sends the configuration to NID.
Step 8	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Configuration Example

The example shows how to create MEP on port2 of NID-1:

```
UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 99
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName W-N-V2000
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ieee name W-N-V2000
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 101
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 4
UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 99
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 102

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 99
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps
octet 1

UCS(ProvisionMepPortType)# addCcAps review
UCS(ProvisionMepPortType)# addCcAps commit
UCS(ProvisionMepPortType)# exit
```

Configuring ERPS on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ErpsPortType** **Example:** `UCS# ErpsPortType`	Enters the ErpsPortType mode.

Step 3	**setErpsInstConfig erpsConfig** {**erpsInst***erpsInst_number* | **mep** {**port0** {**sf***sf_number* | **aps***aps_number*} | **port1** {**sf***sf_number* | **aps***aps_number*}} | **ringType** {**major** | **sub**}}
	
	Example:
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig erpsInst 1`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 aps 100`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 sf 100`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 aps 99`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 sf 99`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig port0 3`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig port1 4`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig ringType major`
		Sets ERPS configuration.
		• **erpsConfig—**Specifies the ERPS configuration.
		• **erpsInst—** Specifies the ERPS instance.
		• *erpsInst_number*—ERPS instance number
		• **mep—**Specifies the MEP configuration.
		• **port0—**Selects the ERPS port 0 interface.
		• **port1—**Selects the ERPS port 0 interface.
		• **sf—**Specifies signal fail MEP.
		• *sf_number*—Signal fail MEP number.
		• **aps—**Specifies the APS MEP.
		• *aps_number*— APS MEP number.
		• **ringType—**Specifies type of ring.
		• **major—**Specifies the major ring.
		• **sub—**Specifies the sub ring.

Step 4	**setErpsInstProperties erpsInstconfig** {**wtrTime***time_in_minutes* | **erpsInst***erp_instance_number* | **rplPort** {**port0** | **port1**} | **rplRole** {**owner** | **neighbour**} | **vlan** {**vlanList***vlan_list_number* | **add** | **remove** | **none**}}
	
	Example:
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig erpsInst 1`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig rplPort port0`
	`UCS(ErpsPortType)# setErpsInstProperties`
		Adds CC/APS configuration request.
		• **wtrTime—** Specifies the WTR time.
		• *time_in_minutes*—WTR time in minutes. Allowed range is 1, 5-12.
		• **erpsInst—**Specifies the ERPS instance.
		• *erp_instance_number*—ERPS instance number.
		• **rplPort—**Specifies the RPL port.
		• **port0—**Selects the ERPS port 0 interface.

	Command or Action	Purpose

	`erpsInstconfig rplRole owner`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig vlan vlanList 2-10`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1`
		• **port1—** Selects the ERPS port 1 interface.
		• **rplRole—**Specifies the RPL role.
		• **owner—**Specifies the RPL owner.
		• **neighbour—**Specifies the RPL neighbour.
		• **vlan—**Specifies the VLAN configuration.
		• **vlanList—**Specifies the VLAN list.
		• *vlan_list_number*— VLAN list number.
		• **add—**Adds to the set of included VLANs.
		• **remove—**Removes from the set of included VLANs.
		• **none—** Does not include any VLANs.

Step 5	**setErpsInstProperties review** **Example:** `UCS(ErpsPortType)# setErpsInstProperties review`	Displays the configuration.
Step 6	**setErpsInstProperties commit** **Example:** `UCS(ErpsPortType)# setErpsInstProperties commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(ErpsPortType)# exit`	Exits the ErpsPortType mode.

Configuration Example

The example shows how to configure ERPS on NID-1:

```
UCS(ErpsPortType)# setErpsInstConfig erpsConfig erpsInst 1
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 aps 100
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 sf 100
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 aps 99
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 sf 99
UCS(ErpsPortType)# setErpsInstConfig erpsConfig port0 3
UCS(ErpsPortType)# setErpsInstConfig erpsConfig port1 4
UCS(ErpsPortType)# setErpsInstConfig erpsConfig ringType major

UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig erpsInst 1
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig rplPort port0
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig rplRole owner
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig vlan vlanList 2-10
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1

UCS(ErpsPortType)# setErpsInstProperties review
UCS(ErpsPortType)# setErpsInstProperties commit
UCS(ErpsPortType)# exit
```

Creating VLAN on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionPortVlanPortType mode.
Step 3	**createVlanCommand createVlanReqvlan_list***vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 2000`	Creates VLAN list.

Step 4	**modifySwPort modifySWPortConfig interface***interface_id* | **mode** [**access Vlan***vlan_number*] | **trunk** {**allowed** | **native**}
	
	Example:
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 3`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2000`
	
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 5`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1`
	`UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2000`
		Modifies the switchport configuration.
		• **interface—**Selects the interface to be configured.
		• *Interface Id*—Specifies the interface ID.
		• **mode—**Specifies the mode of operation.
		• **access—**Sets mode to ACCESS unconditionally.
		• **vlan—**Sets VLAN when interface is in access mode.
		• *vlan_number*—Specifies the VLAN number.
		• **trunk—**Sets mode to TRUNK unconditionally.
		• **allowed—**Sets allowed VLAN characteristics when interface is in trunk mode.
		• **native—**Sets native VLAN.

Step 5	**modifySwPort review** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort review`	Displays the configuration.
Step 6	**modifySwPort commit** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort commit`	Sends the configuration to NID.

	Command or Action	Purpose
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

The example shows how to create VLAN on NID-2:

```
UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 2000

UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 3
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 2000

UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 5
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 1
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 2000

UCS(ProvisionPortVlanPortType)# modifySwPort review
UCS(ProvisionPortVlanPortType)# modifySwPort commit
UCS(ProvisionPortVlanPortType)# exit
```

Creating MEP on Port 1 of NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionMepPortType** **Example:** `UCS# ProvisionMepPortType`	Enters the ProvisionMepPortType mode.
Step 3	**createMep createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level***level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}} **Example:** `UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 100`	Creates MEP configuration. • **mepinstance—**Specifies the MEP instance number. • **mode—**Specifies the mode of the MEP instance. • **mep—**Specifies the maintenance entity end point. • **mip—**Specifies the maintenance entity intermediate point. • **direction—**Selects the direction of the MEP.

	Command or Action	Purpose

	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan`
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName ERPS-1`
	`UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepId 101`
	`UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP`
	`UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 3`
	`UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000`
		• **up—**Specifies an Up MEP - monitoring egress OAM and traffic on residence port.
		• **down—**Specifies a Down MEP - monitoring ingress OAM and traffic on residence port.
		• **domain—**Selects the domain of the MEP.
		• **port—**Specifies a MEP in the Port Domain. Flow Instance is a Port.
		• **evc—**Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created.
		• **vlan—**Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created.
		• **flowId—**Specifies the flow related to the MEP.
		• **vid—**In case the MEP is a port Up-MEP or a EVC customer MIP the VID must be given.
		• **level—**Specifies the MEG level of the MEP.
		• *level_number***—**MEG level number.
		• **residencePort—**Specifies the port monitored by MEP.
		• *port_number***—**Residence port number.
		• **mepId—**Specifies MEP ID.
		• *id_number***—**MEP ID number.
		• **megdomain—**Specifies the maintenance domain configuration.
		• **maName—**Specifies the ITU/IEEE MEG-ID (short MA name).
		• *ma_name***—**Short MA name.
		• **megIdFormat—**Selects the MEG ID format.
		• **ituMeg—**Specifies the MEG-ID using ITU format (ICC - UMC).
		• **ituCcMeg—**Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC).
		• **ieee—**Specifies the MEG-ID (Short MA Name) using IEEE Character String format.

Step 4	**addPeerMepId commit** | **flush** | **peerMepConfig** {**macAddress** | **mepInstance** | **peerMepId**}	Adds peer MEP request. • **commit—**Commits addPeerMepId.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 100`
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 100`
		• **flush**—Flushes all addPeerMepId commands from queue.
		• **peerMepConfig**—Adds peer mep request.
		• **macAddress**—Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception.
		• **mepInstance**—Specifies the mep instance number.
		• **peerMepId**—Specifies the peer MEP-ID.

Step 5	**addCcAps** {**commit** | **flush** | **mepFunctionalConfig** {**aps** {**enable** | **disable**} | **cc** {**enable** | **disable**} | **mepInstance***mep_instance_number*} | **review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 100`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps octet 1`
		Adds CC/APS configuration request.
		• **commit**—Commits addCcAps.
		• **flush**—Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig**—Adds CC/APS configuration request.
		• **aps**—Specifies APS protocol.
		• **enable**—Enables APS.
		• **disbale**—Disables APS.
		• **cc**—Specifies continuity check.
		• **enable**—Enables CC.
		• **disbale**—Disables CC.
		• **mepInstance**—Specifies the mep instance number.
		• *mep_instance_number*—MEP instance number.

Step 6	**addCcAps review** **Example:** `UCS(ProvisionMepPortType)# addCcAps review`	Displays the configuration.
Step 7	**addCcAps commit** **Example:** `UCS(ProvisionMepPortType)# addCcAps commit`	Sends the configuration to NID.
Step 8	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Configuration Example

The example shows how to create MEP on port 1 of NID-2:

```
UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 100
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName ERPS-1
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ituMeg
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 101
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 3
UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 100
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 100

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 100
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps
octet 1

UCS(ProvisionMepPortType)# addCcAps review
UCS(ProvisionMepPortType)# addCcAps commit
UCS(ProvisionMepPortType)# exit
```

Creating MEP on Port 2 of NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionMepPortType** **Example:** `UCS# ProvisionMepPortType`	Enters the ProvisionMepPortType mode.

Step 3	**createMep createMepConfig** {**mepinstance** | **mode** {**mep** | **mip**} | **direction** {**up** | **down**} | **domain** {**port** | **evc** | **vlan**} | **flowId** | **vid** | **level***level_number* | **residencePort** *port_number* | **mepId***id_number* | **megdomain** {**maName***ma_name* | **megIdFormat** {**ituMeg** | **ituCcMeg** | **ieee**}}}
	
	Example:
	`UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 99`
	`UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN`
	`UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan`
		Creates MEP configuration.
		• **mepinstance—**Specifies the MEP instance number.
		• **mode—**Specifies the mode of the MEP instance.
		• **mep—**Specifies the maintenance entity end point.
		• **mip—**Specifies the maintenance entity intermediate point.
		• **direction—**Selects the direction of the MEP.
		• **up—**Specifies an Up MEP - monitoring egress OAM and traffic on residence port.

	Command or Action	Purpose
	`UCS(ProvisionMepPortType)# createMep createMepConfig level 0` `UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName W-N-V2000` `UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ieee name W-N-V2000` `UCS(ProvisionMepPortType)# createMep createMepConfig mepId 103` `UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP` `UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 5` `UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000`	• **down**—Specifies a Down MEP - monitoring ingress OAM and traffic on residence port. • **domain**—Selects the domain of the MEP. • **port**—Specifies a MEP in the Port Domain. Flow Instance is a Port. • **evc**—Specifies a MEP in the EVC Domain. Flow Instance is a EVC. The EVC must be created. • **vlan**—Specifies a MEP in the VLAN Domain. Flow Instance is a VLAN. The VLAN must be created. • **flowId**—Specifies the flow related to the MEP. • **vid**—In case the MEP is a port Up-MEP or a EVC customer MIP the VID must be given. • **level**—Specifies the MEG level of the MEP. • *level_number*—MEG level number. • **residencePort**—Specifies the port monitored by MEP. • *port_number*—Residence port number. • **mepId**—Specifies MEP ID. • *id_number*—MEP ID number. • **megdomain**—Specifies the maintenance domain configuration. • **maName**—Specifies the ITU/IEEE MEG-ID (short MA name). • *ma_name*—Short MA name. • **megIdFormat**—Selects the MEG ID format. • **ituMeg**—Specifies the MEG-ID using ITU format (ICC - UMC). • **ituCcMeg**—Specifies the MEG-ID using ITU Country Code format (CC - ICC - UMC). • **ieee**—Specifies the MEG-ID (Short MA Name) using IEEE Character String format.
Step 4	**addPeerMepId commit** | **flush** | **peerMepConfig** {**macAddress** | **mepInstance** | **peerMepId**} **Example:** `UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 99`	Adds peer MEP request. • **commit**—Commits addPeerMepId. • **flush**—Flushes all addPeerMepId commands from queue. • **peerMepConfig**—Adds peer mep request.

	Command or Action	Purpose
	`UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 104`	• **macAddress—**Specifies the peer MAC. This is overwritten by any learned MAC - through CCM reception. • **mepInstance—**Specifies the mep instance number. • **peerMepId—**Specifies the peer MEP-ID.

Step 5	**addCcAps** {**commit** | **flush** | **mepFunctionalConfig** {**aps** {**enable** | **disable**} | **cc** {**enable** | **disable**} | **mepInstance***mep_instance_number*} | **review**}
	
	Example:
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 99`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7`
	`UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps octet 1`
		Adds CC/APS configuration request.
		• **commit—**Commits addCcAps.
		• **flush—**Flushes all addCcAps commands from queue.
		• **mepFunctionalConfig—**Adds CC/APS configuration request.
		• **aps—**Specifies APS protocol.
		• **enable—**Enables APS.
		• **disbale—**Disables APS.
		• **cc—**Specifies continuity check.
		• **enable—**Enables CC.
		• **disbale—**Disables CC.
		• **mepInstance—**Specifies the mep instance number.
		• *mep_instance_number*—MEP instance number.

Step 6	**addCcAps review** **Example:** `UCS(ProvisionMepPortType)# addCcAps review`	Displays the configuration.
Step 7	**addCcAps commit** **Example:** `UCS(ProvisionMepPortType)# addCcAps commit`	Sends the configuration to NID.
Step 8	**exit** **Example:** `UCS(ProvisionMepPortType)# exit`	Exits the ProvisionMepPortType mode.

Configuration Example

The example shows how to create MEP on port 2 of NID-2:

```
UCS(ProvisionMepPortType)# createMep createMepConfig mepInstance 99
UCS(ProvisionMepPortType)# createMep createMepConfig direction DOWN
UCS(ProvisionMepPortType)# createMep createMepConfig domain vlan
UCS(ProvisionMepPortType)# createMep createMepConfig level 0
```

```
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain maName W-N-V2000
UCS(ProvisionMepPortType)# createMep createMepConfig megDomain megIdFormat ieee name W-N-V2000
UCS(ProvisionMepPortType)# createMep createMepConfig mepId 103
UCS(ProvisionMepPortType)# createMep createMepConfig mode MEP
UCS(ProvisionMepPortType)# createMep createMepConfig residencePort 5
UCS(ProvisionMepPortType)# createMep createMepConfig flow 2000

UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig mepInstance 99
UCS(ProvisionMepPortType)# addPeerMepId peerMepConfig peerMepId 104

UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig mepInstance 99
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig cc enable frameRate fr1s
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable mode multi
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable priority 7
UCS(ProvisionMepPortType)# addCcAps mepFunctionalConfig aps enable switchingProtocol raps
octet 1

UCS(ProvisionMepPortType)# addCcAps review
UCS(ProvisionMepPortType)# addCcAps commit
UCS(ProvisionMepPortType)# exit
```

Configuring ERPS on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ErpsPortType** **Example:** `UCS# ErpsPortType`	Enters the ErpsPortType mode.

Step 3	**setErpsInstConfig erpsConfig** {**erpsInst***erpsInst_number* | **mep** {**port0** {**sf***sf_number* | **aps***aps_number*} | **port1** {**sf***sf_number* | **aps** *aps_number*}} | **ringType** {**major** | **sub**}}
	
	Example:
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig erpsInst 1`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 aps 100`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 sf 100`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 aps 99`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 sf 99`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig port0 3`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig port1 5`
	`UCS(ErpsPortType)# setErpsInstConfig erpsConfig ringType major`
		Sets ERPS configuration.
		• **erpsConfig—**Specifies the ERPS configuration.
		• **erpsInst—** Specifies the ERPS instance.
		• *erpsInst_number*—ERPS instance number
		• **mep—**Specifies the MEP configuration.
		• **port0—**Selects the ERPS port 0 interface.
		• **port1—**Selects the ERPS port 0 interface.
		• **sf—**Specifies signal fail MEP.
		• *sf_number*—Signal fail MEP number.
		• **aps—**Specifies the APS MEP.
		• *aps_number*— APS MEP number.
		• **ringType—**Specifies type of ring.

	Command or Action	Purpose
		• **major**—Specifies the major ring. • **sub**—Specifies the sub ring.

Step 4	**setErpsInstProperties erpsInstconfig** {**wtrTime***time_in_minutes* | **erpsInst***erp_instance_number* | **rplPort** {**port0** | **port1**} | **rplRole** {**owner** | **neighbour**} | **vlan** {**vlanList***vlan_list_number* | **add** | **remove** | **none**}}
	
	Example:
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig erpsInst 1`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig rplPort port0`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig rplRole neighbour`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig vlan vlanList 2-10`
	`UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1`
		Sets ERPS instance.
		• **wtrTime**— Specifies the WTR time.
		• *time_in_minutes*—WTR time in minutes. Allowed range is 1, 5-12.
		• **erpsInst**—Specifies the ERPS instance.
		• *erp_instance_number*—ERPS instance number.
		• **rplPort**—Specifies the RPL port.
		• **port0**—Selects the ERPS port 0 interface.
		• **port1**— Selects the ERPS port 1 interface.
		• **rplRole**—Specifies the RPL role.
		• **owner**—Specifies the RPL owner.
		• **neighbour**—Specifies the RPL neighbour.
		• **vlan**—Specifies the VLAN configuration.
		• **vlanList**—Specifies the VLAN list.
		• *vlan_list_number*— VLAN list number.
		• **add**—Adds to the set of included VLANs.
		• **remove**—Removes from the set of included VLANs.
		• **none**— Does not include any VLANs.

Step 5	**setErpsInstProperties review** **Example:** `UCS(ErpsPortType)# setErpsInstProperties review`	Displays the configuration.
Step 6	**setErpsInstProperties commit** **Example:** `UCS(ErpsPortType)# setErpsInstProperties commit`	Sends the configuration to NID.
Step 7	**exit** **Example:** `UCS(ErpsPortType)# exit`	Exits the ErpsPortType mode.

Configuration Example

The example shows how to configure ERPS on NID-2:

```
UCS(ErpsPortType)# setErpsInstConfig erpsConfig erpsInst 1
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 aps 100
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port0 sf 100
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 aps 99
UCS(ErpsPortType)# setErpsInstConfig erpsConfig mep port1 sf 99
UCS(ErpsPortType)# setErpsInstConfig erpsConfig port0 3
UCS(ErpsPortType)# setErpsInstConfig erpsConfig port1 5
UCS(ErpsPortType)# setErpsInstConfig erpsConfig ringType major

UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig erpsInst 1
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig rplPort port0
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig rplRole neighbour
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig vlan vlanList 2-10
UCS(ErpsPortType)# setErpsInstProperties erpsInstconfig wtrTime 1

UCS(ErpsPortType)# setErpsInstProperties review
UCS(ErpsPortType)# setErpsInstProperties commit
UCS(ErpsPortType)# exit
```

Configuring ERPS on the UCS Controller

To configure ERPS on the UCS Controller, complete the following steps.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `UCS> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `UCS# configure terminal`	Enters global configuration mode.
Step 3	**ethernet ring g8032** *ring-name* **Example:** `UCS(config)# ethernet ring g8032 ring1`	Specifies the Ethernet ring and enters Ethernet ring port configuration mode.
Step 4	**port0 interface** *type number* **Example:** `UCS(config-erp-ring)# port0 interface fastethernet 0/1/0`	Connects port0 of the local node of the interface to the Ethernet ring and enters Ethernet ring protection mode.

	Command or Action	Purpose
Step 5	**monitor service instance** *instance-id* **Example:** `UCS(config-erp-ring-port)# monitor service instance 1`	Assigns the Ethernet service instance to monitor the ring port (port0) and detect ring failures.
Step 6	**exit** **Example:** `UCS(config-erp-ring-port)# exit`	Exits Ethernet ring port configuration mode.
Step 7	**port1** {**interface***type number* | **none**} **Example:** `UCS(config-erp-ring)# port1 interface fastethernet 0/1/1`	Connects port1 of the local node of the interface to the Ethernet ring and enters Ethernet ring protection mode.
Step 8	**monitor service instance** *instance-id* **Example:** `UCS(config-erp-ring-port)# monitor service instance 2`	Assigns the Ethernet service instance to monitor the ring port (port1) and detect ring failures. • The interface (to which port1 is attached) must be a subinterface of the main interface.
Step 9	**exit** **Example:** `UCS(config-erp-ring-port)# exit`	Exits Ethernet ring port configuration mode.
Step 10	**exclusion-list vlan-ids** *vlan-id* **Example:** `UCS(config-erp-ring)# exclusion-list vlan-ids 2`	Specifies VLANs that are unprotected by the Ethernet ring protection mechanism.
Step 11	**open-ring** **Example:** `UCS(config-erp-ring)# open-ring`	Specifies the Ethernet ring as an open ring.
Step 12	**instance** *instance-id* **Example:** `UCS(config-erp-ring)# instance 1`	Configures the Ethernet ring instance and enters Ethernet ring instance configuration mode.

	Command or Action	Purpose
Step 13	**description** *descriptive-name* **Example:** `UCS(config-erp-inst)# description cisco_customer_instance`	Specifies a descriptive name for the Ethernet ring instance.
Step 14	**profile** *profile-name* **Example:** `UCS(config-erp-inst)# profile profile1`	Specifies the profile associated with the Ethernet ring instance.
Step 15	**rpl** {**port0** | **port1**} {**owner** | **neighbor** | **next-neighbor** } **Example:** `UCS(config-erp-inst)# rpl port0 neighbor`	Specifies the Ethernet ring port on the local node as the RPL owner, neighbor, or next neighbor.
Step 16	**inclusion-list vlan-ids** *vlan-id* **Example:** `UCS(config-erp-inst)# inclusion-list vlan-ids 11`	Specifies VLANs that are protected by the Ethernet ring protection mechanism.
Step 17	**aps-channel** **Example:** `UCS(config-erp-inst)# aps-channel`	Enters Ethernet ring instance aps-channel configuration mode.
Step 18	**level** *level-value* **Example:** `UCS(config-erp-inst-aps)# level 5`	Specifies the Automatic Protection Switching (APS) message level for the node on the Ethernet ring. • All nodes in the Ethernet ring must be configured with the same level.
Step 19	**port0** **service instance** *instance-id* **Example:** `UCS(config-erp-inst-aps)# port0 service instance 100`	Associates APS channel information with port0.
Step 20	**port1 service instance** {*instance-id* | **none** } **Example:** `UCS(config-erp-inst-aps)# port1 service instance 100`	Associates APS channel information with port1.

	Command or Action	Purpose
Step 21	**end** **Example:** `UCS(config-erp-inst-aps)# end`	Returns to user EXEC mode.

Configuration Example

The example shows how to configure ERPS on the UCS Controller:

```
!
ethernet cfm domain W-N-V2000 level 0
service W-N-V2000 evc evc2000 vlan 2000 direction down
  continuity-check
  continuity-check interval 1s
  efd notify g8032
!

!
interface GigabitEthernet0/1
switchport trunk allowed vlan none
switchport mode trunk
!
service instance 2000 ethernet evc2000
  encapsulation dot1q 2000
  bridge-domain 2000
  cfm mep domain W-N-V2000 mpid 102
   rmep mpid 101
!

!
interface GigabitEthernet0/20
switchport trunk allowed vlan none
switchport mode trunk
!
service instance 2000 ethernet evc2000
  encapsulation dot1q 2000
  bridge-domain 2000
  cfm mep domain W-N-V2000 mpid 104
   rmep mpid 103
!

!
ethernet ring g8032 profile 1
timer wtr 1
!
ethernet ring g8032 1
port0 interface GigabitEthernet0/1
port1 interface GigabitEthernet0/20
instance 1
  profile 1
  inclusion-list vlan-ids 2-10,2000
  aps-channel
   level 0
   port0 service instance 2000
   port1 service instance 2000
  !
!
```

Verifying ERPS

Use the following command to verify the ERPS status on the UCS controller.

- **showErpsConfig showErpsReq erpsInstList 1**

  This command displays the ERPS status. The following is a sample output from the command:

```
UCS(ErpsPortType)# showErpsConfig showErpsReq erpsInstList 1
UCS(ErpsPortType)# showErpsConfig review
```

```
showErpsConfig reviewCommands in queue:
        showErpsConfig showErpsReq erpsInstList 1
```

```
UCS(ErpsPortType)# showErpsConfig commit
```

```
Stat = 0ShowErpsConfig_Output.erpsInfo.erpsInstance[0].grpId = 1
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.ringType.t = 1
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.ringType.u.major
 = 'major'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.virtualConnection.t
 = 2
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.virtualConnection.u.disable
 = 'Disable'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.interconnect.t
 = 2
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.interconnect.u.disable
 = 'Disable'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.instance = 0
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.port0 = 3
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.port1 = 4
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.mep.port0.sf =
 100
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.mep.port0.aps
= 100
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.mep.port1.sf =
 99
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].config.mep.port1.aps
= 99
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.guardTime
= 500
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.wtrTime =
1
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.revertive.t
 = 1
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.revertive.u.enable
 = 'Enable'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.version.t
= 2
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.version.u.v2
 = 'V2'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.topologyChangePropagate.t
 = 2
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.topologyChangePropagate.u.disable
 = 'Disable'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.holdoff =
0
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.rplRole.t
```

```
= 1
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.rplRole.u.owner
 = 'owner'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.rplPort.t
= 1
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.rplPort.u.port0
 = 'port0'
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.vlan.t = 1
ShowErpsConfig_Output.erpsInfo.erpsInstance[0].instConfig.vlan.u.vlanList
 =
'2,3,4,5,6,7,8,9,10,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,1023,1024,1022,1021,1013,1012'


 ShowErpsConfig Commit Success!!!
```

- **showErpsStats erpsShowStateReq erpsInst 1**

  This command displays the ERPS status on the NID. The following is a sample output from the command:

```
UCS(EpsPortType)# showErpsStats erpsShowStateReq erpsInst 1
UCS(EpsPortType)# showErpsStats erpsShowStateReq show brief
UCS(EpsPortType)# showErpsstats commit
```

```
ShowErpsStats_Output.erpsState.erpsInst[0].grpId = 1
ShowErpsStats_Output.erpsState.erpsInst[0].show.t = 1
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.grpId = 1
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.type = 'Maj'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.version = '2'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.port0 =
'GigabitEthernet 1/3'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.port0Lnk =
'U'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.port0Blk =
'B'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.majGrp = ''
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rplRole =
'Ownr'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rplPort =
'Port0'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rplBlk = 'Y'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.fsmState =
'IDLE'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rApsTx = 'Y'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rApsPort0Rx
= ' '
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.fop = 'N'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.port1 =
'GigabitEthernet 1/4'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.port1Lnk =
'U'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.port1Blk =
'U'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.revertive =
'Rev'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.ringType =
'-'
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rplRole_1 =
''
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rplPort_1 =
```

```
''
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rplBlk_1 = ''
ShowErpsStats_Output.erpsState.erpsInst[0].show.u.brief.rApsPort1Rx
= ' '

ShowErpsStats Commit Success!!!
```

CHAPTER 21

Configuring L2CP

This document describes the Layer 2 Control Protocol (L2CP) feature and configuration steps to implement L2CP.

Prerequisites for Configuring L2CP

- NID must be added to the controller.
- NID must be accessible from the controller.
- NID must have an IP address.

Restrictions for Configuring L2CP

- When committing multiple lists, the list in previous commit is not retained. Example: a peer list 16-18 in a previous commit is replaced by a new commit of peer list 21.

  To retain multiple lists, you must specify the lists in a single commit. Example: peer list 16-18,21.
- Any L2CP processing configured using forward/peer/discard modes applies to all EVCs on the port.
- Provisioning L2CP in tunnel mode is not supported. You cannot prevent core switches from processing frame as a L2CP frame in a service provider network.
- To delete a previously configured discardList, you must configure **discard discardList** command with **no** before you commit the command. Otherwise, by default the previously configured value is retained.

  Example:

```
setL2CPPortConfig l2cpPortConfiguration portNumber 6
setL2CPPortConfig l2cpPortConfiguration discard discardList no
```

Information About L2CP

L2CP addresses the requirement for a bidirectional, IP- based protocol that operates across a number of access and aggregation network technologies such as Ethernet. The L2CP message exchange conveys status and control information between access devices and one or more other devices that require the information for executing local functions.

L2CP handling is required for edge switches providing Ethernet Virtual Connections (EVCs) in a service provider network.

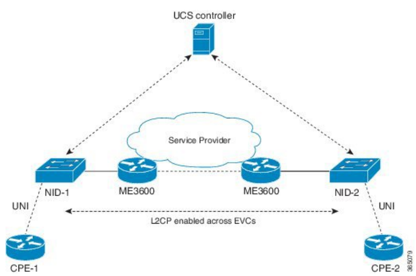
In this case, UCS Controller and NID are used to establish an EVC between UNI ports of two Customer Premise Equipment (CPE).

The following options are available to provision L2CP on NID:

- **Forward—**The L2CP frame is forwarded to the network port like other layer 2 frames in the EVC.
- **Peer—**The L2CP frame is processed by a local protocol entity and is not forwarded.
- **Discard—**The L2CP frame is discarded.

The following figure shows the topology used for provisioning L2CP on NIDs using the UCS Controller.

Figure 13: L2CP Topology



Configuring L2CP Using a UCS Controller

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionL2CPPortType** **Example:** `UCS# ProvisionL2CPPortType`	Enters the ProvisionL2CPPortType mode.

	Command or Action	Purpose
Step 3	**ProvisionL2CPPortType** {**default** | **exit** | **getL2CPPortConfig** | **no** | **setL2CPPortConfig**} **Example:** `UCS# ProvisionL2CPPortType`	Sub-command options. • **default—**Sets a command to its defaults. • **exit—**Exits from ProvisionL2CPPortType sub configuration mode. • **getL2CPPortConfig—**Retrieves current L2CP configuration request. • **no—**Negates a command or set its defaults. • **setL2CPPortConfig—**Configures L2CP forward on EVCs on this port.
Step 4	**getL2CPPortConfig l2cpPort***l2cpPort_number* **Example:** `UCS(ProvisionL2CPPortType)# getL2CPPortConfig l2cpPort 5`	Retrieves initial or default L2CP configuration. • **l2cpPort—**Specifies L2CP port configuration request. • *l2cpPort_number*—L2CP port number. The valid range is from 1 to 6.

Step 5	**setL2CPPortConfig l2cpPortConfiguration** {**enabled** | **portNumber** *portNumber* | **discard** {**discardList***discardList_range*} | **forward** {**forwardList***forwardList_range*} | **peer** {**peerList***peerList_range*}}
	
	Example:
	`UCS(ProvisionL2CPPortType)# setL2CPPortConfig l2cpPortConfiguration portNumber 5`
	`UCS(ProvisionL2CPPortType)# setL2CPPortConfig l2cpPortConfiguration enabled enable`
		Sets up L2CP forward/peer/discard configuration request on all EVCs on this port.
		• **enabled—**Specifies L2CP configuration enabled/disabled on this port.
		• **portNumber—**Specifies port number to configure L2CP.
		• *portNumber*—Port number to configure L2CP. The valid range is from 1 to 6.
		• **discard—**Discards L2CP frames.
		• **discardList—** Selects BPDU addresses and GARP addresses.
		• *discardList_range*— BPDU addresses (0-15) and GARP addresses (16-31).
		• **forward—** Allows forwarding of L2CP frames.
		• **forwardList—**Selects BPDU addresses and GARP addresses.
		• *forwardList_range*— BPDU addresses (0-15) and GARP addresses (16-31).
		• **peer—**Redirects L2CP frames to local protocol entity.
		• **peerList—**Selects BPDU addresses and GARP addresses.
		• *peerList_range*— BPDU addresses (0-15) and GARP addresses (16-31).

	Command or Action	Purpose
Step 6	**setL2CPPortConfig review** **Example:** `UCS(L2CPPortType)# setL2CPPortConfig review`	(Optional) Displays the configuration.
Step 7	**setL2CPPortConfig flush** **Example:** `UCS(L2CPPortType)# setL2CPPortConfig flush`	(Optional) Flushes the configuration.
Step 8	**setL2CPPortConfig commit** **Example:** `UCS(L2CPPortType)# setL2CPPortConfig commit`	Sends the configuration to NID.
Step 9	**getL2CPPortConfig l2cpPort** *l2cpPort_number* **Example:** `UCS(ProvisionL2CPPortType)# getL2CPPortConfig l2cpPort 5`	Retrieves current L2CP configuration for a specified port. • **l2cpPort—**Specifies L2CP port configuration request. • *l2cpPort_number*—L2CP port number. The valid range is from 1 to 6.
Step 10	**getL2CPPortConfig review** **Example:** `UCS(ProvisionL2CPPortType)# getL2CPPortConfig review`	(Optional) Displays the configuration.
Step 11	**getL2CPPortConfig flush** **Example:** `UCS(ProvisionL2CPPortType)# getL2CPPortConfig flush`	(Optional) Flushes the configuration.
Step 12	**getL2CPPortConfig commit** **Example:** `UCS(ProvisionL2CPPortType)# getL2CPPortConfig commit`	Sends the configuration to NID.
Step 13	**exit** **Example:** `UCS(ProvisionL2CPPortType)# exit`	Exits the ProvisionL2CPPortType mode.

Configuration Example

- The example shows how to enable L2CP on a port:

```
UCS# ProvisionL2CPPortType ?
  <cr>

UCS(ProvisionL2CPPortType)#?
ProvisionL2CPPortType sub-mode commands:
```

```
  default            Set a command to its defaults
  exit               Exit from ProvisionL2CPPortType sub configuration mode
  getL2CPPortConfig  Get current L2CP configuration request
  no                 Negate a command or set its defaults
  setL2CPPortConfig  Configure L2CP forward on EVCs on this port

UCS(ProvisionL2CPPortType)#setL2CPPortConfig ?
  commit                 commit setL2CPPortConfig
  flush                  flush all setL2CPPortConfig commands from queue
  l2cpPortConfiguration  Configure L2CP forward on EVCs on this port
  review                 review setL2CPPortConfig commands

UCS(ProvisionL2CPPortType)#setL2CPPortConfig l2
UCS(ProvisionL2CPPortType)#$pPortConfiguration ?
  discard     Discard L2CP frames
  enabled     L2CP configuration enabled/disabled on this port
  forward     Allow forwarding of L2CP frames
  peer        Redirect L2CP frames to local protocol entity
  portNumber  Port number to configure L2CP

UCS(ProvisionL2CPPortType)#$guration portNumber 3
UCS(ProvisionL2CPPortType)#$guration enabled enable

UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable

Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
UCS(ProvisionL2CPPortType)#setL2CPPortConfig commit
 SetL2CPPortConfig Commit Success!!!
```

When SetL2CPPortConfig operation is executed on a NID from a UCS Controller, initially GetL2CPPortConfig fetches the current configuration. This is followed by SetL2CPPortConfig to set the new L2CP configuration. The following is a sample output on the NID.

```
#
Decoding of Request message was successful
Decoded record:
GetL2CPPortConfig_Output.l2cpPortConfiguration.enabled = true
GetL2CPPortConfig_Output.l2cpPortConfiguration.portNumber = 3
GetL2CPPortConfig_Output.l2cpPortConfiguration.discard.discardList =
 ''
GetL2CPPortConfig_Output.l2cpPortConfiguration.forward.forwardList =
 '16-31'
GetL2CPPortConfig_Output.l2cpPortConfiguration.peer.peerList = '0-15'
GetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
Decoding of Request message was successful
Decoded record:
SetL2CPPortConfig_Input.l2cpPortConfiguration.enabled = true
SetL2CPPortConfig_Input.l2cpPortConfiguration.portNumber = 3
SetL2CPPortConfig_Input.l2cpPortConfiguration.discard.discardList =
''
```

```
SetL2CPPortConfig_Input.l2cpPortConfiguration.forward.forwardList =
'16-31'
SetL2CPPortConfig_Input.l2cpPortConfiguration.peer.peerList = '0-15'
Encoding of Response message was successful
Encoded record:
SetL2CPPortConfig_Output.l2cpPortConfigResponse = 0
SetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
SetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
SetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
SetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
SetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
SetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

- The examples shows how to enable L2CP Forward on a port.

```
UCS# ProvisionL2CPPortType ?
  <cr>

UCS(ProvisionL2CPPortType)#?
ProvisionL2CPPortType sub-mode commands:
  default            Set a command to its defaults
  exit               Exit from ProvisionL2CPPortType sub configuration mode
  getL2CPPortConfig  Get current L2CP configuration request
  no                 Negate a command or set its defaults
  setL2CPPortConfig  Configure L2CP forward on EVCs on this port

UCS(ProvisionL2CPPortType)#setL2CPPortConfig ?
  commit                 commit setL2CPPortConfig
  flush                  flush all setL2CPPortConfig commands from queue
  l2cpPortConfiguration  Configure L2CP forward on EVCs on this port
  review                 review setL2CPPortConfig commands

UCS(ProvisionL2CPPortType)#setL2CPPortConfig l2
UCS(ProvisionL2CPPortType)#$pPortConfiguration ?
  discard     Discard L2CP frames
  enabled     L2CP configuration enabled/disabled on this port
  forward     Allow forwarding of L2CP frames
  peer        Redirect L2CP frames to local protocol entity
  portNumber  Port number to configure L2CP

UCS(ProvisionL2CPPortType)#$guration portNumber 3
UCS(ProvisionL2CPPortType)#$guration enabled enable
UCS(ProvisionL2CPPortType)#$guration forward for
UCS(ProvisionL2CPPortType)#$orward forwardList 1-14
UCS(ProvisionL2CPPortType)#
UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration forward forwardList 1-14
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
UCS(ProvisionL2CPPortType)#setL2CPPortConfig commit
 SetL2CPPortConfig Commit Success!!!
```

The following is a sample output on the NID.

```
#
Decoding of Request message was successful
Decoded record:
GetL2CPPortConfig_Input.l2cpPort = 3
Encoding of Response message was successful
Encoded record:
GetL2CPPortConfig_Output.l2cpPortConfiguration.enabled = true
```

```
GetL2CPPortConfig_Output.l2cpPortConfiguration.portNumber = 3
GetL2CPPortConfig_Output.l2cpPortConfiguration.discard.discardList =
 ''
GetL2CPPortConfig_Output.l2cpPortConfiguration.forward.forwardList =
 '16-31'
GetL2CPPortConfig_Output.l2cpPortConfiguration.peer.peerList = '0-15'
GetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
Decoding of Request message was successful
Decoded record:
SetL2CPPortConfig_Input.l2cpPortConfiguration.enabled = true
SetL2CPPortConfig_Input.l2cpPortConfiguration.portNumber = 3
SetL2CPPortConfig_Input.l2cpPortConfiguration.discard.discardList =
''
SetL2CPPortConfig_Input.l2cpPortConfiguration.forward.forwardList =
'1-14'
SetL2CPPortConfig_Input.l2cpPortConfiguration.peer.peerList = '0-15'
Encoding of Response message was successful
Encoded record:
SetL2CPPortConfig_Output.l2cpPortConfigResponse = 0
SetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
SetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
SetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
SetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
SetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
SetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

```
#show running-config interface GigabitEthernet 1/3
```

```
Building configuration...
interface GigabitEthernet 1/3
 switchport hybrid allowed vlan 1
 switchport hybrid acceptable-frame-type untagged
 switchport hybrid ingress-filtering
 switchport hybrid port-type unaware
 switchport mode trunk
 lldp med type end-point
 qos dscp-remark rewrite
 evc l2cp forward 1-14
```

- The examples shows how to enable L2CP Forward, Peer, Discard on a port.

```
UCS# ProvisionL2CPPortType ?
  <cr>

UCS(ProvisionL2CPPortType)#?
```

```
ProvisionL2CPPortType sub-mode commands:
  default            Set a command to its defaults
  exit               Exit from ProvisionL2CPPortType sub configuration mode
  getL2CPPortConfig  Get current L2CP configuration request
  no                 Negate a command or set its defaults
  setL2CPPortConfig  Configure L2CP forward on EVCs on this port

UCS(ProvisionL2CPPortType)#setL2CPPortConfig ?
  commit                 commit setL2CPPortConfig
  flush                  flush all setL2CPPortConfig commands from queue
  l2cpPortConfiguration  Configure L2CP forward on EVCs on this port
  review                 review setL2CPPortConfig commands

UCS(ProvisionL2CPPortType)#setL2CPPortConfig l2
UCS(ProvisionL2CPPortType)#$pPortConfiguration ?
  discard     Discard L2CP frames
  enabled     L2CP configuration enabled/disabled on this port
  forward     Allow forwarding of L2CP frames
  peer        Redirect L2CP frames to local protocol entity
  portNumber  Port number to configure L2CP

UCS(ProvisionL2CPPortType)#$guration portNumber 3
UCS(ProvisionL2CPPortType)#$guration enabled enable
UCS(ProvisionL2CPPortType)#$guration forward for
UCS(ProvisionL2CPPortType)#$orward forwardList 1-14
UCS(ProvisionL2CPPortType)#$peer peerList 16-20
UCS(ProvisionL2CPPortType)#$guration discard di
UCS(ProvisionL2CPPortType)#$iscard discardList 19
UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration peer peerList 16-20
        setL2CPPortConfig l2cpPortConfiguration discard discardList 19
        setL2CPPortConfig l2cpPortConfiguration forward forwardList 1-14

UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration peer peerList 16-20
        setL2CPPortConfig l2cpPortConfiguration discard discardList 19
        setL2CPPortConfig l2cpPortConfiguration forward forwardList 1-14
UCS(ProvisionL2CPPortType)#setL2CPPortConfig commit
 SetL2CPPortConfig Commit Success!!!
```

The following is a sample output on the NID.

```
# Decoding of Request message was successful
Decoded record:
GetL2CPPortConfig_Input.l2cpPort = 3
Encoding of Response message was successful
Encoded record:
GetL2CPPortConfig_Output.l2cpPortConfiguration.enabled = true
GetL2CPPortConfig_Output.l2cpPortConfiguration.portNumber = 3
GetL2CPPortConfig_Output.l2cpPortConfiguration.discard.discardList =
 ''
GetL2CPPortConfig_Output.l2cpPortConfiguration.forward.forwardList =
 '1-14'
GetL2CPPortConfig_Output.l2cpPortConfiguration.peer.peerList = '0-15'
GetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetL2CPPortConfig_Output.xmlns:soapenc =
```

```
"http://schemas.xmlsoap.org/soap/encoding/"
GetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
Decoding of Request message was successful
Decoded record:
SetL2CPPortConfig_Input.l2cpPortConfiguration.enabled = true
SetL2CPPortConfig_Input.l2cpPortConfiguration.portNumber = 3
SetL2CPPortConfig_Input.l2cpPortConfiguration.discard.discardList =
'19'
SetL2CPPortConfig_Input.l2cpPortConfiguration.forward.forwardList =
'1-14'
SetL2CPPortConfig_Input.l2cpPortConfiguration.peer.peerList = '16-20'
Encoding of Response message was successful
Encoded record:
SetL2CPPortConfig_Output.l2cpPortConfigResponse = 0
SetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
SetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
SetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
SetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
SetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
SetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

```
# show running-config interface GigabitEthernet 1/3
```

```
Building configuration...
interface GigabitEthernet 1/3
 switchport hybrid allowed vlan 1
 switchport hybrid acceptable-frame-type untagged
 switchport hybrid ingress-filtering
 switchport hybrid port-type unaware
 switchport mode trunk
 lldp med type end-point
 qos dscp-remark rewrite
 evc l2cp peer 16-18,20 forward 1-14 discard 19
!
end
```

- The examples shows how to disable or reset L2CP Discard on a port.

```
UCS# ProvisionL2CPPortType ?
  <cr>

UCS(ProvisionL2CPPortType)#?
ProvisionL2CPPortType sub-mode commands:
  default            Set a command to its defaults
  exit               Exit from ProvisionL2CPPortType sub configuration mode
  getL2CPPortConfig  Get current L2CP configuration request
  no                 Negate a command or set its defaults
  setL2CPPortConfig  Configure L2CP forward on EVCs on this port

UCS(ProvisionL2CPPortType)#setL2CPPortConfig ?
  commit                 commit setL2CPPortConfig
  flush                  flush all setL2CPPortConfig commands from queue
  l2cpPortConfiguration  Configure L2CP forward on EVCs on this port
  review                 review setL2CPPortConfig commands

UCS(ProvisionL2CPPortType)#setL2CPPortConfig l2
UCS(ProvisionL2CPPortType)#$pPortConfiguration ?
```

```
  discard     Discard L2CP frames
  enabled     L2CP configuration enabled/disabled on this port
  forward     Allow forwarding of L2CP frames
  peer        Redirect L2CP frames to local protocol entity
  portNumber  Port number to configure L2CP

UCS(ProvisionL2CPPortType)#$guration portNumber 3
UCS(ProvisionL2CPPortType)#$guration enabled enable
UCS(ProvisionL2CPPortType)#$guration discard di
UCS(ProvisionL2CPPortType)#$iscard discardList no
UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration peer peerList 16-20
        setL2CPPortConfig l2cpPortConfiguration discard discardList 19
        setL2CPPortConfig l2cpPortConfiguration forward forwardList 1-14

UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration discard discardList no

UCS(ProvisionL2CPPortType)#setL2CPPortConfig commit
 SetL2CPPortConfig Commit Success!!!
```

The following is a sample output on the NID.

```
# Decoding of Request message was successful
Decoded record:
GetL2CPPortConfig_Input.l2cpPort = 3
Encoding of Response message was successful
Encoded record:
GetL2CPPortConfig_Output.l2cpPortConfiguration.enabled = true
GetL2CPPortConfig_Output.l2cpPortConfiguration.portNumber = 3
GetL2CPPortConfig_Output.l2cpPortConfiguration.discard.discardList =
 '19'
GetL2CPPortConfig_Output.l2cpPortConfiguration.forward.forwardList =
 '1-14'
GetL2CPPortConfig_Output.l2cpPortConfiguration.peer.peerList = '16-20'
GetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
Decoding of Request message was successful
Decoded record:
SetL2CPPortConfig_Input.l2cpPortConfiguration.enabled = true
SetL2CPPortConfig_Input.l2cpPortConfiguration.portNumber = 3
SetL2CPPortConfig_Input.l2cpPortConfiguration.discard.discardList =
'no'
SetL2CPPortConfig_Input.l2cpPortConfiguration.forward.forwardList =
'1-14'
SetL2CPPortConfig_Input.l2cpPortConfiguration.peer.peerList = '16-20'
Encoding of Response message was successful
Encoded record:
SetL2CPPortConfig_Output.l2cpPortConfigResponse = 0
```

```
SetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
SetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
SetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
SetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
SetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
SetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

```
# show running-config interface GigabitEthernet 1/3
```

```
Building configuration...
interface GigabitEthernet 1/3
 switchport hybrid allowed vlan 1
 switchport hybrid acceptable-frame-type untagged
 switchport hybrid ingress-filtering
 switchport hybrid port-type unaware
 switchport mode trunk
 lldp med type end-point
 qos dscp-remark rewrite
 evc l2cp peer 16-20 forward 1-14
```

- The examples shows how to retrieve current configuration on a NID.

```
UCS(ProvisionL2CPPortType)#getL2CPPortConfig ?
  commit    commit getL2CPPortConfig
  flush     flush all getL2CPPortConfig commands from queue
  l2cpPort  Get current L2CP configuration request
  review    review getL2CPPortConfig commands

UCS(ProvisionL2CPPortType)#getL2CPPortConfig l2cpPort ?
  <1-6>  Get L2CP Port Configuration Request

UCS(ProvisionL2CPPortType)#$Config l2cpPort 3 ?
  <cr>
UCS(ProvisionL2CPPortType)#getL2CPPortConfig review
Commands in queue:
        getL2CPPortConfig l2cpPort 3
UCS(ProvisionL2CPPortType)#
UCS(ProvisionL2CPPortType)#getL2CPPortConfig commit
GetL2CPPortConfig_Output.l2cpPortConfiguration.enabled = true
GetL2CPPortConfig_Output.l2cpPortConfiguration.portNumber = 3
GetL2CPPortConfig_Output.l2cpPortConfiguration.discard.discardList = 'no'
GetL2CPPortConfig_Output.l2cpPortConfiguration.forward.forwardList = '1-14'
GetL2CPPortConfig_Output.l2cpPortConfiguration.peer.peerList = '16-20'

GetL2CPPortConfig Commit Success!!!
```

The following is a sample output on the NID.

```
# Decoding of Request message was successful
Decoded record:
GetL2CPPortConfig_Input.l2cpPort = 3
Encoding of Response message was successful
Encoded record:
GetL2CPPortConfig_Output.l2cpPortConfiguration.enabled = true
GetL2CPPortConfig_Output.l2cpPortConfiguration.portNumber = 3
GetL2CPPortConfig_Output.l2cpPortConfiguration.discard.discardList =
 'no'
GetL2CPPortConfig_Output.l2cpPortConfiguration.forward.forwardList =
 '1-14'
```

```
GetL2CPPortConfig_Output.l2cpPortConfiguration.peer.peerList = '16-20'
GetL2CPPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetL2CPPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetL2CPPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetL2CPPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetL2CPPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetL2CPPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

- The examples shows how to flush L2CP configuration on a port.

```
UCS# ProvisionL2CPPortType ?
  <cr>

UCS(ProvisionL2CPPortType)#?
ProvisionL2CPPortType sub-mode commands:
  default            Set a command to its defaults
  exit               Exit from ProvisionL2CPPortType sub configuration mode
  getL2CPPortConfig  Get current L2CP configuration request
  no                 Negate a command or set its defaults
  setL2CPPortConfig  Configure L2CP forward on EVCs on this port

UCS(ProvisionL2CPPortType)#setL2CPPortConfig ?
  commit                 commit setL2CPPortConfig
  flush                  flush all setL2CPPortConfig commands from queue
  l2cpPortConfiguration  Configure L2CP forward on EVCs on this port
  review                 review setL2CPPortConfig commands

UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration forward forwardList 1-14
        setL2CPPortConfig l2cpPortConfiguration peer peerList 16-20
UCS(ProvisionL2CPPortType)#setL2CPPortConfig flush
UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
 No commands in queue
```

- The examples shows how to remove specific configuration CLI from a UCS controller.

```
UCS# ProvisionL2CPPortType ?
  <cr>

UCS(ProvisionL2CPPortType)#?
ProvisionL2CPPortType sub-mode commands:
  default            Set a command to its defaults
  exit               Exit from ProvisionL2CPPortType sub configuration mode
  getL2CPPortConfig  Get current L2CP configuration request
  no                 Negate a command or set its defaults
  setL2CPPortConfig  Configure L2CP forward on EVCs on this port

UCS(ProvisionL2CPPortType)#setL2CPPortConfig ?
  commit                 commit setL2CPPortConfig
  flush                  flush all setL2CPPortConfig commands from queue
  l2cpPortConfiguration  Configure L2CP forward on EVCs on this port
  review                 review setL2CPPortConfig commands

UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration forward forwardList 1-14
        setL2CPPortConfig l2cpPortConfiguration peer peerList 16-20

UCS(ProvisionL2CPPortType)#no setL2CPPortConfig l2cpPor$
UCS(ProvisionL2CPPortType)#$n forward forwardList 1-14
```

```
UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration peer peerList 16-20
UCS(ProvisionL2CPPortType)#setL2CPPortConfig review
Commands in queue:
        setL2CPPortConfig l2cpPortConfiguration portNumber 3
        setL2CPPortConfig l2cpPortConfiguration enabled enable
        setL2CPPortConfig l2cpPortConfiguration peer peerList 16-20
UCS(ProvisionL2CPPortType)#setL2CPPortConfig commit
 SetL2CPPortConfig Commit Success!!!
```

CHAPTER 22

Configuring MAC Security

This document describes the MAC security feature and configuration steps to implement MAC security.

Prerequisites for Configuring MAC Security

Information About MAC Security

You can use the MAC security feature to restrict input to an interface by limiting and identifying MAC addresses of the devices that are allowed to access the port. When you assign secure MAC addresses to a secure port, the port does not forward packets with source addresses outside the group of defined addresses. If you limit the number of secure MAC addresses to one and assign a single secure MAC address, the device attached to that port is assured the full bandwidth of the port.

How to Provision MAC Security

Configuring Port Security

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMacTableSecurityType** **Example:** `UCS# ProvisionMacTableSecurityType`	Enters the ProvisionMacTableSecurityType mode.

Step 3	**portSecurityGlobalConfig portSecurityGlobalConfigReq** {**mode** {**disable** | **enable**} | **agingTime** {**time** *time* | **disable**}}
	
	Example:
	`UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig portSecurityGlobalConfigReq agingTime time 60`
	`UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig portSecurityGlobalConfigReq mode enable`
		Port Security Global Configuration.
		• **portSecurityGlobalConfigReq—**Specifies port security global configuration.
		• **mode—**Option to enable/disable port security.
		• **disable—**Disables port security globally.
		• **enable—**Enables port security globally.
		• **agingTime—**Enables or disables port security aging.
		• **time—**Enables and sets time.
		• *time*—Time. The valid range is from 10 to 10000000 seconds.
		• **disable—**Disables aging.

Step 4	**portSecurityGlobalConfig review** **Example:** `UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig review`	Displays the configuration.
Step 5	**portSecurityGlobalConfig commit** **Example:** `UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig commit`	Sends the configuration to NID.
Step 6	**portSecurityInterfaceConfig portSecurityInterfaceConfigReq** {**interface_id** *interface_id* | **port_security** {**mode** {**disable** | **enable**} |**mac_limit**	Interface mode Port Security Configuration. • **interface_id—**Specifies the interface ID.

	Command or Action	Purpose

	{**maximum***max_mac_number* | **disable**} | **violation** {**protect** | **shutdown** | **traponly** | **trap-shutdown** | **disable**}}}
	
	Example:
	`UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq interface_id 2`
	`UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq port_security mode enable`
	
	`UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq interface_id 2`
	`UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq port_security mac_limit maximum 100`
	
	`UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq interface_id 2`
	`UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq port_security violation shutdown`
		• *interface_id*—Interface ID. The valid range is from 1 to 6.
		• **port_security—**Configures port security.
		• **mode—**Specifies the mode for port security.
		• **disable—**Disables port security.
		• **enable—**Enables port security.
		• **mac_limit—**Specifies MAC address learning limit.
		• **maximum—**Specifies the maximum number of MAC addresses.
		• *max_mac_number*—Maximum number of MAC addresses. The valid range is from 1 to 1024.
		• **disable—**Removes the MAC limit.
		• **violation—**Specifies the action when exceeding the limit.
		• **protect—**Specifies no action.
		• **shutdown—**Shuts down the port.
		• **traponly—**Sends an SNMP trap.
		• **trap-shutdown—**Sends an SNMP trap and shuts down the port.
		• **disable—**Disables violation type.

Step 7	**portSecurityInterfaceConfig review** **Example:** `UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig review`	Displays the configuration.
Step 8	**portSecurityInterfaceConfig commit** **Example:** `UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(ProvisionMacTableSecurityType)# exit`	Exits the ProvisionMacTableSecurityType mode.

Configuration Example

The example shows how to configure port security:

```
UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig portSecurityGlobalConfigReq
agingTime time 60
UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig portSecurityGlobalConfigReq
mode enable
UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig review
UCS(ProvisionMacTableSecurityType)# portSecurityGlobalConfig commit

UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq
 interface_id 2
UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq
 port_security mode enable
UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq
 interface_id 2
UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq
 port_security mac_limit maximum 100
UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq
 interface_id 2
UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig portSecurityInterfaceConfigReq
 port_security violation shutdown

UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig review
UCS(ProvisionMacTableSecurityType)# portSecurityInterfaceConfig commit
UCS(ProvisionMacTableSecurityType)# exit
```

Configuring MAC Security

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMacTableSecurityType** **Example:** `UCS# ProvisionMacTableSecurityType`	Enters the ProvisionMacTableSecurityType mode.
Step 3	**setMacGlobalConfig setMacGlobalConfigReq** {**macAgingTime** {**setAgingTime** *aging_time* | **disable**} | **staticMacEntry** | **learning**} **Example:** `UCS(ProvisionMacTableSecurityType)# setMacGlobalConfig setMacGlobalConfigReq macAgingTime setAgingTime 100`	Global Configuration for MAC address table. • **macAgingTime—**Configures MAC aging time. • **setAgingTime—**Specifies aging time. • *aging_time*—Aging time in seconds. Valid range is from 10 to 1000000. • **disable—**Disables MAC aging. • **staticMacEntry—**Specifies static MAC address. • **learning—**Specifies MAC learning on VLAN.

	Command or Action	Purpose
Step 4	**setMacGlobalConfig review** **Example:** `UCS(ProvisionMacTableSecurityType)# setMacGlobalConfig review`	Displays the configuration.
Step 5	**setMacGlobalConfig commit** **Example:** `UCS(ProvisionMacTableSecurityType)# setMacGlobalConfig commit`	Sends the configuration to NID.

Step 6	**setMacInterfaceConfig setMacInterfaceConfigReq** {**interface_id** interface-id **|** **mode** {**learning** {**enable** | **disable**} | **secure** {**enable** | **disable**}}}
	
	Example:
	`UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig setMacInterfaceConfigReq interface-id 3`
	`UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig setMacInterfaceConfigReq mode learning enable`
		Interface mode for MAC configuration.
		• **interface_id—**Specifies the interface ID.
		• *interface_id*—Interface ID.
		• **mode—**Specifies the learning mode.
		• **learning—**Specifies port default learning mode.
		• **enable—**Enables MAC learning.
		• **disable—**Disables MAC learning.
		• **secure—**Specifies port secure learning mode.
		• **enable—**Enables secure MAC learning.
		• **disable—**Disables secure MAC learning.

Step 7	**setMacInterfaceConfig review** **Example:** `UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig review`	Displays the configuration.
Step 8	**setMacInterfaceConfig commit** **Example:** `UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig commit`	Sends the configuration to NID.
Step 9	**exit** **Example:** `UCS(ProvisionMacTableSecurityType)# exit`	Exits the ProvisionMacTableSecurityType mode.

Configuration Example

The example shows how to configure MAC table learning:

```
UCS(ProvisionMacTableSecurityType)# setMacGlobalConfig setMacGlobalConfigReq macAgingTime
setAgingTime 100
```

```
UCS(ProvisionMacTableSecurityType)# setMacGlobalConfig review
UCS(ProvisionMacTableSecurityType)# setMacGlobalConfig commit

UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig setMacInterfaceConfigReq
interface_id 3
UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig setMacInterfaceConfigReq mode
learning enable
UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig review
UCS(ProvisionMacTableSecurityType)# setMacInterfaceConfig commit

UCS(ProvisionMacTableSecurityType)# exit
```

Clearing MAC Address Table

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionMacTableSecurityType** **Example:** `UCS# ProvisionMacTableSecurityType`	Enters the ProvisionMacTableSecurityType mode.
Step 3	**clearMacTable clearMacTableReq clearAll** **Example:** `UCS(ProvisionMacTableSecurityType)# clearMacTable clearMacTableReq clearAll`	Clears MAC address table. • **clearAll—**Clears all entries.
Step 4	**clearMacTable review** **Example:** `UCS(ProvisionMacTableSecurityType)# clearMacTable review`	Displays the configuration.
Step 5	**clearMacTable commit** **Example:** `UCS(ProvisionMacTableSecurityType)# clearMacTable commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(ProvisionMacTableSecurityType)# exit`	Exits the ProvisionMacTableSecurityType mode.

Configuration Example

The example shows how to configure port security:

```
UCS(ProvisionMacTableSecurityType)# clearMacTable clearMacTableReq clearAll
UCS(ProvisionMacTableSecurityType)# clearMacTable review
UCS(ProvisionMacTableSecurityType)# clearMacTable commit
UCS(ProvisionMacTableSecurityType)# exit
```

Verifying MAC Security

Use the following command to verify the MAC security status on the UCS controller.

- **showMacTableLearningReq mode**

  This command displays the MAC table learning status. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)# showmacTablelearning showMacTableLearningReq mode
UCS(ProvisionMacTableSecurityType)# showmacTablelearning review
```

```
Commands in queue:
showMacTableLearning showMacTableLearningReq mode
```

```
UCS(ProvisionMacTableSecurityType)# showmacTablelearning commit
```

```
Clearing Socket 4
ShowMacTableLearning_Output.showMacTableLearingResp.status[0].interface_
 = 1
ShowMacTableLearning_Output.showMacTableLearingResp.status[0].mode =
 'Auto'
ShowMacTableLearning_Output.showMacTableLearingResp.status[1].interface_
 = 2
ShowMacTableLearning_Output.showMacTableLearingResp.status[1].mode =
 'Auto'
ShowMacTableLearning_Output.showMacTableLearingResp.status[2].interface_
 = 3
ShowMacTableLearning_Output.showMacTableLearingResp.status[2].mode =
 'Auto'
ShowMacTableLearning_Output.showMacTableLearingResp.status[3].interface_
 = 4
ShowMacTableLearning_Output.showMacTableLearingResp.status[3].mode =
 'Auto'
ShowMacTableLearning_Output.showMacTableLearingResp.status[4].interface_
 = 5
ShowMacTableLearning_Output.showMacTableLearingResp.status[4].mode =
 'Auto'
ShowMacTableLearning_Output.showMacTableLearingResp.status[5].interface_
 = 6
ShowMacTableLearning_Output.showMacTableLearingResp.status[5].mode =
 'Auto'
 ShowMacTableLearning Commit Success!!!
```

- **showMacTableEntriesReq all**

  This command displays the list of all MAC entries. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)# showMacTableEntries showMacTableEntriesReq all
UCS(ProvisionMacTableSecurityType)# showmacTableentries review
```

```
Commands in queue:
 showMacTableEntries showMacTableEntriesReq all
```

UCS(ProvisionMacTableSecurityType)# **showmacTableentries commit**

```
Clearing Socket 4
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].mac_address
 = '00:00:0c:07:ac:03'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].mac_address
 = '00:09:e8:74:36:c5'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].mac_address
 = '00:14:1b:ec:18:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].mac_address
 = '00:19:a9:a2:9e:80'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].mac_address
 = '00:1c:b0:f5:b4:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].mac_address
 = '00:1c:b1:9a:00:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].vlan_id
 = 1
```

```
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].mac_address
 = '00:1c:b1:f9:d0:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].mac_address
 = '00:1c:b1:fa:48:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].mac_address
 = '00:3a:99:fd:4b:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].ports
 = ' CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].mac_address
 = '18:9c:5d:a7:f4:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].mac_address
 = '33:33:00:00:00:01'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].mac_address
 = '33:33:00:00:00:02'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].mac_address
 = '33:33:ff:fd:4b:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].type
 = 'Static '
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].mac_address
 = 'ff:ff:ff:ff:ff:ff'
```

```
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].ports
 = 'GigabitEthernet 1/1-6 CPU'
 ShowMacTableEntries Commit Success!!!
```

- **showMacTableEntriesReq interface_id 2**

  This command displays the list of all MAC entries for a given interface. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)#showMacTableEntries showMacTableEntriesReq
interface_id 2
UCS(ProvisionMacTableSecurityType)# showmacTableentries review
```

```
Commands in queue:
 showMacTableEntries showMacTableEntriesReq interface_id 2
```

```
UCS(ProvisionMacTableSecurityType)# showmactableentries commit
```

```
Clearing Socket 4
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].mac_address
 = '33:33:00:00:00:01'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].mac_address
 = '33:33:00:00:00:02'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].mac_address
 = '33:33:ff:fd:4b:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].mac_address
 = 'ff:ff:ff:ff:ff:ff'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].ports
 = 'GigabitEthernet 1/1-6 CPU'

 ShowMacTableEntries Commit Success!!!
```

- **showMacTableEntriesReq mac_address ff:ff:ff:ff:ff:ff**

This command displays the list of all MAC entries for a given MAC address. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)# showMacTableEntries showMacTableEntriesReq
mac-address ff:ff:ff:ff:ff:ff
UCS(ProvisionMacTableSecurityType)# showmacTableentries review
```

```
Commands in queue:
 showMacTableEntries showMacTableEntriesReq mac_address
ff:ff:ff:ff:ff:ff
```

```
UCS(ProvisionMacTableSecurityType)# showmactableentries commit
```

```
Clearing Socket 4
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].mac_address
 = 'ff:ff:ff:ff:ff:ff'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].ports
 = 'GigabitEthernet 1/1-6 CPU'

 ShowMacTableEntries Commit Success!!!
```

- **showMacTableEntriesReq static**

This command displays all the static MAC entries. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)# showMacTableEntries showMacTableEntriesReq static
UCS(ProvisionMacTableSecurityType)# showmacTableentries review
```

```
Commands in queue:
 showMacTableEntries showMacTableEntriesReq static
```

```
UCS(ProvisionMacTableSecurityType)# showmactableentries commit
```

```
Clearing Socket 4
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].mac_address
 = '00:00:0c:07:ac:03'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].mac_address
 = '00:09:e8:74:36:c5'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].mac_address
 = '00:14:1b:ec:18:00'
```

```
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].mac_address
 = '00:19:a9:a2:9e:80'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].mac_address
 = '00:1c:b0:f5:b4:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].mac_address
 = '00:1c:b1:9a:00:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].mac_address
 = '00:1c:b1:f9:d0:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].mac_address
 = '00:1c:b1:fa:48:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].mac_address
 = '00:3a:99:fd:4b:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].ports
 = ' CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].mac_address
 = '18:9c:5d:a7:f4:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].ports
 = 'GigabitEthernet 1/1 CPU'
```

```
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].mac_address
 = '33:33:00:00:00:01'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].mac_address
 = '33:33:00:00:00:02'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].mac_address
 = '33:33:ff:fd:4b:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].mac_address
 = 'ff:ff:ff:ff:ff:ff'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].ports
 = 'GigabitEthernet 1/1-6 CPU'
 ShowMacTableEntries Commit Success!!!
```

- **showMacTableEntriesReq vlan_id 1**

  This command displays all the MAC entries for a specified VLAN ID. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)# showMacTableEntries showMacTableEntriesReq vlan-id
 1
UCS(ProvisionMacTableSecurityType)# showmacTableentries review
```

```
Commands in queue:
  showMacTableEntries showMacTableEntriesReq vlan_id 1
```

```
UCS(ProvisionMacTableSecurityType)# showMacTableEntries commit
```

```
Clearing Socket 4
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].mac_address
 = '00:00:0c:07:ac:03'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[0].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].vlan_id
```

```
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].mac_address
 = '00:09:e8:74:36:c5'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[1].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].mac_address
 = '00:14:1b:ec:18:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[2].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].mac_address
 = '00:19:a9:a2:9e:80'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[3].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].mac_address
 = '00:1c:b0:f5:b4:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[4].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].mac_address
 = '00:1c:b1:9a:00:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[5].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].mac_address
 = '00:1c:b1:f9:d0:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[6].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].mac_address
 = '00:1c:b1:fa:48:00'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[7].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].mac_address
```

```
 = '00:3a:99:fd:4b:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[8].ports
 = ' CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].mac_address
 = '18:9c:5d:a7:f4:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[9].ports
 = 'GigabitEthernet 1/1 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].mac_address
 = '33:33:00:00:00:01'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[10].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].mac_address
 = '33:33:00:00:00:02'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[11].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].vlan0_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].mac_address
 = '33:33:ff:fd:4b:1c'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[12].ports
 = 'GigabitEthernet 1/1-6 CPU'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].type
 = 'Static'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].vlan_id
 = 1
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].mac_address
 = 'ff:ff:ff:ff:ff:ff'
ShowMacTableEntries_Output.showMacTableEntriesResp.macEntries[13].ports
 = 'GigabitEthernet 1/1-6 CPU'
 ShowMacTableEntries Commit Success!!!
```

- **showMacAddressCountReq count**

  This command displays count of MAC addresses learnt per interface. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)# showMacTableCount showMacAddressCountReq count
UCS(ProvisionMacTableSecurityType)# showmacTablecount review
```

```
Commands in queue:
 showMacTableCount showMacAddressCountReq count
```

```
UCS(ProvisionMacTableSecurityType)# showmacTablecount commit
```

```
Clearing Socket 4
ShowMacTableCount_Output.showMacAddressCountResp.macCount[0].Interface_id
 = 1
ShowMacTableCount_Output.showMacAddressCountResp.macCount[0].count =
 9
ShowMacTableCount_Output.showMacAddressCountResp.macCount[1].Interface_id
 = 2
ShowMacTableCount_Output.showMacAddressCountResp.macCount[1].count =
 0
ShowMacTableCount_Output.showMacAddressCountResp.macCount[2].Interface_id
 = 3
ShowMacTableCount_Output.showMacAddressCountResp.macCount[2].count =
 0
ShowMacTableCount_Output.showMacAddressCountResp.macCount[3].Interface_id
 = 4
ShowMacTableCount_Output.showMacAddressCountResp.macCount[3].count =
 0
ShowMacTableCount_Output.showMacAddressCountResp.macCount[4].Interface_id
 = 5
ShowMacTableCount_Output.showMacAddressCountResp.macCount[4].count =
 0
ShowMacTableCount_Output.showMacAddressCountResp.macCount[5].Interface_id
 = 6
ShowMacTableCount_Output.showMacAddressCountResp.macCount[5].count =
 0
 ShowMacTableCount Commit Success!!!
```

- **showMacTableAgingReq agingTimeValue**

  This command displays the MAC aging time. The following is a sample output from the command:

```
UCS(ProvisionMacTableSecurityType)# showMacTableAgingTime showMacTableAgingReq
agingTimeValue
UCS(ProvisionMacTableSecurityType)# showmactableagingTime review
```

```
Commands in queue:
 showMacTableAgingTime showMacTableAgingReq agingTimeValue
```

```
UCS(ProvisionMacTableSecurityType)# showmactableagingTime commit
```

```
Clearing Socket 4
ShowMacTableAgingTime_Output.showMacTableAgingResp.macAgingTime = 'MAC
 Age Time: 300'

 ShowMacTableAgingTime Commit Success!!!
```

CHAPTER 23

Configuring NTP

The Network Time Protocol (NTP) synchronizes the time of day among a set of distributed time servers and clients so that you can correlate events when you receive system logs and other time-specific events from multiple network devices. NTP uses the User Datagram Protocol (UDP) as its transport protocol. All NTP communications use Coordinated Universal Time (UTC).

Prerequisites for Configuring NTP

Restrictions for Configuring NTP

- Maximum number of servers supported is 5.

Information About NTP

NTP Timestamping Synchronization

Y.1731 Delay Mesaurment uses the timestamps in the packets to calculate the delay between two end-points/systems. The Cisco ASR 900 router supports PHY assisted timestamping for Y.1731 one-way Delay Measurment (1 DM) and two-way Delay Measurement (2 DM) OAM packets. 1 DM requires both MEP end-points clock to be synchronized. When PTP is used, DM packets is timestamped with PTP time in the PHY's.

NTP is a client-server protocol running over IP/UDP and used to synchronize the network devices clock to a common clock source/reference clock. NTP client process clock information from NTP server and updates the system clock periodically. The system clock can be used to update timestamp information in the delay measurement packets.

NTP timestamping feature provides an alternative option to do hardware assisted timestamping with NTP time for Y.1731 1 DM packets., To perform hardware assisted timestamping for Y.1731 1 DM packets, the system clock (TOD) must be synchronized to the hardware clock used in the PHY's. To configure NTP on the router, use the platform time-source ntp command. For information, see Cisco IOS Interface and Hardware Component Command Reference.

How to Configure NTP

Provisioning the UCS Controller to Configure NTP

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**NtpPortType** **Example:** `UCS# NtpPortType`	Enters NTP provisioning mode.

Step 3	**NtpPortType** {**default** | **deleteNtpConfig** | **exit** | **getNtpConfig** | **no** | **setNtpConfig**}
	
	Example:
	`UCS(NtpPortType)# ?`
	`NtpPortType sub-mode commands:`
	`  default           Set a command to its defaults`
	`  deleteNtpConfig  delete NTP config request`
	`  exit              Exit from NtpPortType sub configuration`
	`mode`
	`  getNtpConfig     get ntp properties request`
	`  no                Negate a command or set its defaults`
	`  setNtpConfig     Set Ntp Server Details`
		Displays the supported configurations for NTP.

Step 4	**exit** **Example:** `UCS(NtpPortType)# exit`	Exits the NtpPortType mode.

Configuration Example

The following example shows the supported NTP configuration:

```
UCS(NtpPortType)# ?
NtpPortType sub-mode commands:
  default           Set a command to its defaults
  deleteNtpConfig   delete NTP config request
  exit              Exit from NtpPortType sub configuration mode
  getNtpConfig      get ntp properties request
  no                Negate a command or set its defaults
  setNtpConfig      Set Ntp Server Details
```

Configuring NTP on the UCS Controller

Before You Begin

- Ensure that the NID is reachable for the provided NTP server.
- Set the time zone for synchronization with the NTP server. See Configuring the System Clock.
- Perform the steps to provision NTP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**setNtpConfig** {**commit** | **flush** | **ntpConfig** | **review**}
	
	Example:
	`UCS(NtpPortType)# setNtpConfig ?`
	`  commit            commit deleteNtpConfig`
	`  flush             flush all deleteNtpConfig commands`
	`from queue`
	`  ntpConfig         Set Ntp Server Details`
	`  review            review deleteNtpConfig commands`
		Sets NTP configuration
		• **commit**—Sends the NTP configuration to NID.
		• **flush**—Flushes all NTP configuration from the queue.
		• **ntpConfig**—Sets the NTP server configuration on the UCS controller.
		• **review**—Displays the configuration on the UCS controller.

Step 2	**setNtpConfig ntpConfig** {**hostinfo** {**hostname** *host_name*} | **ipv4address** *IPv4_address* | **ipv6address** *IPv6_address*} | **ntpmode** {**enable** | **number** *server_number*}}
	
	Example:
	`UCS(NtpPortType)# setNtpConfig hostinfo hostname host1`
	
	`UCS(NtpPortType)# setNtpConfig ipv4address 192.34.7.8`
	`UCS(NtpPortType)# setNtpConfig ipv6address`
	`2001:DB8:0:ABCD::1`
	`UCS(NtpPortType)# setNtpConfig ntpmode enable`
	`UCS(NtpPortType)# setNtpConfig ntpmode number 5`
		Configures NTP.
		• **hostinfo**—Sets the host information such as host name, IPv4 address and IPv6 address on the UCS controller.
		• **ntpmode**—Enables or disables the NTP mode on the UCS controller.
		• **number** *server_number*—Sets the NTP server details. The valid range is from 1 to 5.

	Command or Action	Purpose

Step 3	**setNtpconfig review**
	
	Example:
	`UCS(NtpPortType)# setNtpconfig review`
	
	`Commands in queue:`
	`setNtpConfig ntpConfig hostInfo hostName host1`
	`setNtpConfig ntpConfig hostInfo ipv4Address 192.34.7.8`
	`setNtpConfig ntpConfig ntpMode enable`
	`setNtpConfig ntpConfig number 5`
	`setNtpConfig ntpConfig ntpMode enable`
		Displays the NTP configuration on the UCS controller.

Step 4	**setNtpconfigcommit** **Example:** `UCS(NtpPortType)# setNtpconfig commit`	Sends the NTP configuration to the NID.
Step 5	**exit** **Example:** `UCS(NtpPortType)# exit`	Exits the NtpPortType mode.

Configuration Example

The example shows how to configure NTP on the UCS controller:

```
UCS(NtpPortType)# setNtpConfig hostinfo hostname host1
UCS(NtpPortType)# setNtpConfig ipv4address 192.34.7.8
UCS(NtpPortType)# setNtpConfig ipv6address 2001:DB8:0:ABCD::1
UCS(NtpPortType)# setNtpConfig ntpmode enable
UCS(NtpPortType)# setNtpConfig ntpmode number 5
UCS(NtpPortType)# setNtpconfig review

Commands in queue:
        setNtpConfig ntpConfig hostInfo hostName host1
        setNtpConfig ntpConfig hostInfo ipv4Address 192.34.7.8
        setNtpConfig ntpConfig ntpMode enable
        setNtpConfig ntpConfig number 5
        setNtpConfig ntpConfig ntpMode enable

UCS(NtpPortType)# setNtpconfig commit
UCS(NtpPortType)# exit
```

Configuring NTP with Default Configuration

You can set the default NTP configuration on the UCS controller.

Before You Begin

- Perform the steps to provision NTP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**default**{**getNtpConfig** | **setNtpConfig** | **deleteNtpConfig** | **exit** }
	
	Example:
	`UCS(NtpPortType)# default ?`
	
	`deleteNtpConfig  delete NTP config request`
	`exit             Exit from NtpPortType sub configuration mode`
	`getNtpConfig     get ntp properties request`
	`setNtpConfig     Set Ntp Server Details`
		Sets the default NTP configuration.
		• **getNtpConfig—**View the configuration on the UCS controller.
		• **setNtpConfig—**Sets the configuration on the UCS controller.
		• **deleteNtpConfig—**Deletes the configuration from the UCS controller.
		• **exit—**Exits from NtpPortType configuration mode.

Step 2	**exit** **Example:** `UCS(NtpPortType)# exit`	Exits the NtpPortType mode.

Viewing the NTP Configuration

Before You Begin

- Perform the steps to provision NTP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getNtpConfig** {**commit** | **flush** | **ntpStatusRequest** *ntp_status*| **review**}
	
	Example:
	`UCS(NtpPortType)#getNtpConfig ntpStatusRequest 1`
	`UCS(NtpPortType)#getNtpConfig review`
	`UCS(NtpPortType)#getNtpConfig commit`
		• **ntpStatusRequest—**Request NTP configuration properties.
		• **commit—**Sends the NTP configuration to NID.
		• **flush—**Flushes all NTP configuration from the queue.
		• **review—**Displays the configuration.

Step 2	**exit** **Example:** `UCS(NtpPortType)# exit`	Exits the NtpPortType mode.

Configuration Example

The example shows how to view the configuration:

```
UCS(NtpPortType)# getNtpConfig ntpStatusRequest 1
UCS(NtpPortType)# getNtpConfig review

Commands in queue:
        getNtpConfig ntpStatusRequest 1
        getNtpConfig ntpStatusRequest 2
        getNtpConfig ntpStatusRequest 3

UCS(NtpPortType)# getNtpConfig commmit
UCS(NtpPortType)# end
```

Deleting the NTP Configuration

Before You Begin

- Perform the steps to provision NTP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**deleteNtpConfig** {**commit** | **flush** | **ntpDeleteConfig** | **review**}
	
	Example:
	`UCS(NtpPortType)# deleteNtpConfig ?`
	`  commit            commit deleteNtpConfig`
	`  flush             flush all deleteNtpConfig`
	`commands from queue`
	`  ntpDeleteConfig  delete NTP config request`
	`  review           review deleteNtpConfig commands`
		Removes the NTP configuration.
		• **commit—**Sends the NTP configuration to NID.
		• **flush—**Flushes all NTP configuration from the queue.
		• **ntpDeleteConfig—**Deletes the NTP configuration request on the UCS controller.
		• **review—**Displays the configuration on the UCS controller.

Step 2	**ntpDeleteConfig** {**ntpEnable** | **ntpServerNo***server_num*}
	
	Example:
	`UCS(NtpPortType)# deleteNtpConfig ntpDeleteConfig ntpEnable`
	`UCS(NtpPortType)# deleteNtpConfig ntpDeleteConfig ntpServer 1`
		Removes NTP configuration.
		• **ntpEnable—**Disables the NTP configuration.
		• **ntpServerNo—**Disables the NTP server.
		• *server_num*—Specifies the NTP server. The valid range is from 1 to 5.

Step 3	**ntpDeleteConfig review** **Example:** `UCS(NtpPortType)# deleteNtpConfig review`	Displays the NTP configuration.
Step 4	**ntpDeleteConfig commit** **Example:** `UCS(NtpPortType)# deleteNtpConfig commit`	Sends the NTP configuration to the NID.

	Command or Action	Purpose
Step 5	**exit** **Example:** `UCS(NtpPortType)# exit`	Exits the NtpPortType mode.

Configuration Example

The following example shows how to delete the NTP configuration:

```
UCS(NtpPortType)# deleteNtpConfig ntpDeleteConfig ntpEnable
UCS(NtpPortType)# deleteNtpConfig ntpDeleteConfig ntpServer 1
UCS(NtpPortType)# deleteNtpConfig review
Commands in queue:
        deleteNtpConfig ntpDeleteConfig ntpEnable
        deleteNtpConfig ntpDeleteConfig ntpServerNo 2
UCS(NtpPortType)# deleteNtpConfig commit
DeleteNtpConfig Commit Success!!!
UCS(NtpPortType)# deleteNtpConfig exit
```

Verifying NTP

Use these commands to verify the NTP status on the UCS controller.

- **show ntp status**

  This command displays the NTP status on the NID. The following is a sample output from the command:

```
UCS# show ntp status

NTP Mode : disabled
Idx   Server IP host address (a.b.c.d) or a host name string
---   ------------------------------------------------------
1
2
3
4
5
```

CHAPTER 24

Configuring Storm Control

A traffic storm occurs when packets flood the LAN, creating excessive traffic and degrading network performance. The traffic broadcast and multicast suppression (or storm control) feature prevents LAN ports from being disrupted by a broadcast, multicast and unicast traffic storm on physical interfaces.

Restrictions for Configuring Strorm Control

- Storm control cannot be configured per port. It is configured globally on all ports

Information on Storm Control

A broadcast storm occurs when huge amount of broadcast, multicast, or unknown unicast packets flood the LAN, creating excessive traffic and degrading network performance. Errors in the protocol-stack implementation or in the network configuration can also cause a storm. The mechanism to prevent and control such events is known as storm control or broadcast suppression.

Broadcast and Multicast Suppression monitors incoming traffic levels periodically, and compares traffic level with configured storm control policer level or rate. The traffic storm control threshold level is measured based on the traffic rate in bits (or kilobits) per second at which broadcast, multicast, unicast packets are received.

Storm control prevents traffic on a LAN from being disrupted by a broadcast, multicast, or unicast storm on a port. Storm control is applicable for physical interfaces and is used to restrict the unicast, broadcast and multicast ingress traffic on the Layer2 interfaces.

How to Configure Storm Control

Provisioning the UCS Controller to Configure Storm Control

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionStormControl** **Example:** `UCS#`**`ProvisionStormControl`**	Enters Storm control provisioning mode.
Step 3	**ProvisionStormControl** {**getStormControlGlobal** | **setStormControlGlobal** | **showStormControl** |**no** | **exit**} **Example:** `UCS(ProvisionStormControl)#` **`?`**	Displays the supported configurations for storm control. • **getStormControlGlobal—**View the configuration, see Retrieving the Storm Control Configuration. • **setStormControlGlobal—**Configures storm control, see Configuring Storm Control on the. • **showStormControlGlobal—**Displays the configuration, see Displaying the Storm Control Configuration. • **no—**Negates the configuration, see Negating Storm Control Configuration and Restoring Defaults. • **exit—**Exits the configuration.
Step 4	**exit** **Example:** `UCS(ProvisionStormControl)#` **`exit`**	Exits the ProvisionStormControl mode.

Configuration Example

The following example shows the supported storm control configuration:

```
UCS(ProvisionStormControl)# ?
ProvisionStormControl sub-mode commands:
  exit                   Exit from ProvisionStormControl sub configuration mode
  getStormControlGlobal  Storm Control Global Configuration Get Request
  no                     Negate a command or set its defaults
  setStormControlGlobal  Storm Control Global Configuration Set Request
  showStormControl       Display Storm Control Policer properties
```

Configuring Storm Control with Default Configuration

Before You Begin

SUMMARY STEPS

1. **default** {**getStormControlGlobal** | **setStormControlGlobal** | **showStormControl** | **exit**}
2. **exit**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**default** {**getStormControlGlobal** | **setStormControlGlobal** | **showStormControl** | **exit**}
	
	Example:
	`Switch(ProvisionStormControl)# default ?`
	`  exit                  Exit from ProvisionStormControl`
	` sub configuration mode`
	`  getStormControlGlobal  Storm Control Global`
	`Configuration Get Request`
	`  setStormControlGlobal  Storm Control Global`
	`Configuration Set Request`
	`  showStormControl       Display Storm Control Policer`
	`properties`
		Configures default storm control.
		• **getStormControlGlobal—**Views the configuration on the.
		• **setStormControlGlobal—**Sets the configuration on the .
		• **showStormControl—**Displays the configuration from the .
		• **exit—**Exits from ProvisionStormControl mode.

Step 2	**exit** **Example:** `Switch(ProvisionStormControl)# exit`	

Retrieving the Storm Control Configuration

Before You Begin

- Perform the steps to provision storm control on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**getStormControlGlobal** {**commit** | **flush** | **getStormControlGlobalRequest** | **review**}	Retrieve the storm control configuration. • **getStormControlGlobalRequest—**Request storm control configuration properties.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionStormControl)#getStormControlGlobal getStormControlGlobalRequest`
	`UCS(ProvisionStormControl)#getStormControlGlobal review`
	`UCS(ProvisionStormControl)#getStormControlGlobal commit`
		• **commit**—Sends the storm control configuration to NID.
		• **flush**—Flushes all storm control configuration from the queue.
		• **review**—Displays the configuration.

Step 2	**exit** **Example:** `UCS(ProvisionStormControl)# exit`	Exits the ProvisionStormControl mode.

Configuration Example

The example shows how to retrieve the configuration on the UCS controller:

```
UCS(ProvisionStormControl)#getStormControlGlobal getStormControlGlobalRequest
UCS(ProvisionStormControl)#getStormControlGlobal review
Commands in queue:
        getStormControlGlobal getStormControlGlobalRequest
        getStormControlGlobal getStormControlGlobalRequest

UCS(ProvisionStormControl)# getStormControlGlobal commmit
UCS(ProvisionStormControl)# end
```

Displaying the Storm Control Configuration

Before You Begin

- Perform the steps to provision storm control on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**showStormControl** {**commit** | **flush** | **showStormControlReq** | **review**}
	Example:
	`UCS(ProvisionStormControl)#showStormControl showStormControlReq`
	`UCS(ProvisionStormControl)#showStormControl review`
	`UCS(ProvisionStormControl)#showStormControl commit`
		Displays the storm control configuration.
		• **showStormControlReq**—Displays storm control policer properties.
		• **commit**—Sends the show storm control configuration to NID.
		• **flush**—Flushes all show storm control configuration from the queue.
		• **review**—Displays the show storm configuration.

	Command or Action	Purpose
Step 2	**exit** **Example:** `UCS(ProvisionStormControl)# exit`	Exits the ProvisionStormControl mode.

Configuration Example

The example shows how to view the configuration:

```
UCS(ProvisionStormControl)#showStormControl showStormControlReq
UCS(ProvisionStormControl)#showStormControl review

Commands in queue:
        showStormControl showStormControlReq
UCS(ProvisionStormControl)#showStormControl commit

ShowStormControl_Output.stormControlGlobalConfiguration.broadcast.bc_enabled = true
ShowStormControl_Output.stormControlGlobalConfiguration.broadcast.level = 1024000
ShowStormControl_Output.stormControlGlobalConfiguration.broadcast.mode.t = 2
ShowStormControl_Output.stormControlGlobalConfiguration.broadcast.mode.u.kbps = 'kbps'
ShowStormControl_Output.stormControlGlobalConfiguration.multicast.mc_enabled = true
ShowStormControl_Output.stormControlGlobalConfiguration.multicast.level = 512000
ShowStormControl_Output.stormControlGlobalConfiguration.multicast.mode.t = 2
ShowStormControl_Output.stormControlGlobalConfiguration.multicast.mode.u.kbps = 'kbps'
ShowStormControl_Output.stormControlGlobalConfiguration.unicast.uc_enabled = true
ShowStormControl_Output.stormControlGlobalConfiguration.unicast.level = 1000
ShowStormControl_Output.stormControlGlobalConfiguration.unicast.mode.t = 2
ShowStormControl_Output.stormControlGlobalConfiguration.unicast.mode.u.kbps = 'kbps'
ShowStormControl Commit Success!!!

UCS(ProvisionStormControl)# exit
```

Negating Storm Control Configuration and Restoring Defaults

Note Following are the default values for storm control:

- broadcast
  - bc_enabled = false
  - level = 1
  - mode = bps
- multicast
  - mc_enabled = false
  - level = 1
  - mode = bps
- unicast
  - uc_enabled = false
  - level = 1
  - mode = bps

Before You Begin

- Perform the steps to provision storm control on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**no** {**getStormControlGlobal** | **setStormControlGlobal** | **showStormControl** | **exit**}
	
	Example:
	`UCS(ProvisionStormControl)# no ?`
	`  exit                 Exit from ProvisionStormControl sub configuration mode`
	`  getStormControlGlobal  Storm Control Global Configuration Get Request`
	`  setStormControlGlobal  Storm Control Global Configuration Set Request`
	`  showStormControl       Display Storm Control Policer properties`
		Negates storm control configurations.
		• **getStormControlGlobal—**View the configuration.
		• **setStormControlGlobal—**Sets the configuration.
		• **showStormControl—**Displays the configuration.
		• **exit—**Exits from ProvisionStormControl mode.

	Command or Action	Purpose
Step 2	**exit** **Example:** `UCS(ProvisionStormControl)# exit`	Exits the ProvisionStormControl mode.

Configuration Example

The following example is a sample output for negation:

```
UCS(ProvisionStormControl)#showStormControl review
Commands in queue:
showStormControl showStormControlReq

UCS(ProvisionStormControl)#no showStormControl showStormControlReq
UCS(ProvisionStormControl)#showStormControl review
No commands in queue

UCS(ProvisionStormControl)#setStormControlGlobal review
Commands in queue:
setStormControlGlobal stormControlGlobalConfiguration broadcast bc_enabled enable
setStormControlGlobal stormControlGlobalConfiguration broadcast level 64
setStormControlGlobal stormControlGlobalConfiguration broadcast level 128

UCS(ProvisionStormControl)#no setStormControlGlobal stormControlGlobalConfiguration broadcast
 level 64
UCS(ProvisionStormControl)#stormControlGlobal review
Commands in queue:
setStormControlGlobal stormControlGlobalConfiguration broadcast bc_enabled enable
setStormControlGlobal stormControlGlobalConfiguration broadcast level 128
```

Deleting the NTP Configuration

Before You Begin

- Perform the steps to provision NTP on the UCS controller.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**deleteNtpConfig** {**commit** | **flush** | **ntpDeleteConfig** | **review**}
	
	Example:
	`UCS(NtpPortType)# deleteNtpConfig ?`
	`  commit            commit deleteNtpConfig`
	`  flush             flush all deleteNtpConfig commands from queue`
	`  ntpDeleteConfig   delete NTP config request`
	`  review            review deleteNtpConfig commands`
		Removes the storm control configuration.
		• **commit—**Sends the NTP configuration to NID.
		• **flush—**Flushes all NTP configuration from the queue.
		• **ntpDeleteConfig—**Deletes the NTP configuration request on the UCS controller.
		• **review—**Displays the configuration.

	Command or Action	Purpose

Step 2	**ntpDeleteConfig** { **ntpEnable** | **ntpServerNo***server_num* }
	
	Example:
	`UCS(NtpPortType)# deleteNtpConfig ntpDeleteConfig ntpEnable`
	`UCS(NtpPortType)# deleteNtpConfig ntpDeleteConfig ntpServer 1`
		• **ntpEnable—**Disables the NTP configuration.
		• **ntpServerNo—**Disables the NTP server.
		• *server_num***—**Specifies the NTP server. The valid range is from 1 to 5.

Step 3	**exit** **Example:** `UCS(NtpPortType)# exit`	Exits the NtpPortType mode.

CHAPTER 25

Configuring Syslog

This document describes the Syslog feature and configuration steps to implement Syslog.

Prerequisites for Configuring Syslog

Information About Syslog

Syslog is a method to collect messages from devices to a server running a syslog daemon. A syslog service simply accepts messages, and stores them in files or prints them according to a simple configuration file. This form of logging is the best available for Cisco devices because it can provide protected long-term storage for logs. This is useful both in routine troubleshooting and in incident handling.

Enabling Syslog

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**sysLog** **Example:** `UCS# sysLog`	Enters the sysLog mode.

Step 3	**setSyslogProperties sysLogConf** {**valid** | **logServer** {**valid** | **host** {**hostname** | **ipv4address**}} | **level** {**info** | **error** | **warning**}} {**enable** | **disable**}
	
	Example:
	`UCS(sysLog)# setSyslogProperties sysLogConf valid enable`
	`UCS(sysLog)# setSyslogProperties sysLogConf level info`
	`UCS(sysLog)# setSyslogProperties sysLogConf logServer valid enable`
	`UCS(sysLog)# setSyslogProperties sysLogConf logServer host ipv4address 10.78.101.221`
		Configures syslog properties.
		• **sysLogConf—**Sets syslog properties.
		• **valid—**Enables or disables logging.
		• **logServer—**Specifies the logging server.
		• **valid—**Enables or disables logging server.
		• **host—**Specifies log server name or hostname.
		• **hostname—**Specifies domain name of the logging server.
		• **ipv4address—**Specifies IPv4 address of the logging server.
		• **level—**Specifies the log level.
		• **info—** Specifies the level information.
		• **error—** Specifies the level error.
		• **warning—** Specifies the level warning.
		• **enable—** Enables configuration of syslog properties.
		• **disable—** Disables configuration of syslog properties.

Step 4	**setSyslogProperties review** **Example:** `UCS(sysLog)# setSyslogProperties review`	Displays the configuration.
Step 5	**setSyslogProperties commit** **Example:** `UCS(sysLog)# setSyslogProperties commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(sysLog)# exit`	Exits the sysLog mode.

Configuration Example

The example shows how to enable syslog:

```
UCS(sysLog)# setSyslogProperties sysLogConf valid enable
UCS(sysLog)# setSyslogProperties sysLogConf level info
UCS(sysLog)# setSyslogProperties sysLogConf logServer valid enable
UCS(sysLog)# setSyslogProperties sysLogConf logServer host ipv4address 10.78.101.221

UCS(sysLog)# setSyslogProperties review
UCS(sysLog)# setSyslogProperties commit
UCS(sysLog)# exit
```

Clearing Syslog

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**sysLog** **Example:** `UCS# sysLog`	Enters the sysLog mode.
Step 3	**clearSysLog clearLogLevelConf** {**error** | **info** | **warning**} {**enable** | **disable**} **Example:** `UCS(sysLog)# clearSysLog clearLogLevelConf info enable`	Clears system log information. • **clearLogLevelConf**—Clears log information. • **error**—Specifies level error. • **info**—Specifies level information. • **warning**—Specifies level warning. • **enable**—Enables the clearing of log information. • **disable**—Disables the clearing of log information.
Step 4	**clearSysLog review** **Example:** `UCS(sysLog)# clearSysLog review`	Displays the configuration.
Step 5	**clearSysLog commit** **Example:** `UCS(sysLog)# clearSysLog commit`	Sends the configuration to NID.
Step 6	**clearSysLog clearLogLevelConf** {**error** | **info** | **warning**} {**enable** | **disable**}	Clears system log level errors. • **clearLogLevelConf**—Clears log information.

	Command or Action	Purpose
	Example: `UCS(sysLog)# clearSysLog clearLogLevelConf error enable`	• **error—**Specifies level error. • **info—**Specifies level information. • **warning—**Specifies level warning. • **enable—**Enables the clearing of log information. • **disable—**Disables the clearing of log information.
Step 7	**clearSysLog review** **Example:** `UCS(sysLog)# clearSysLog review`	Displays the configuration.
Step 8	**clearSysLog commit** **Example:** `UCS(sysLog)# clearSysLog commit`	Sends the configuration to NID.
Step 9	**clearSysLog clearLogLevelConf** {**error** | **info** | **warning**} {**enable** | **disable**} **Example:** `UCS(sysLog)# clearSysLog clearLogLevelConf warning enable`	Clears system log level warnings. • **clearLogLevelConf—**Clears log information. • **error—**Specifies level error. • **info—**Specifies level information. • **warning—**Specifies level warning. • **enable—**Enables the clearing of log information. • **disable—**Disables the clearing of log information.
Step 10	**clearSysLog review** **Example:** `UCS(sysLog)# clearSysLog review`	Displays the configuration.
Step 11	**clearSysLog commit** **Example:** `UCS(sysLog)# clearSysLog commit`	Sends the configuration to NID.
Step 12	**exit** **Example:** `UCS(sysLog)# exit`	Exits the sysLog mode.

Configuration Example

The example shows how to clear syslog:

```
UCS(sysLog)# clearSysLog clearLogLevelConf info
UCS(sysLog)# clearSysLog review
UCS(sysLog)# clearSysLog commit
UCS(sysLog)# clearSysLog clearLogLevelConf error
UCS(sysLog)# clearSysLog review
UCS(sysLog)# clearSysLog commit
UCS(sysLog)# clearSysLog clearLogLevelConf warning
UCS(sysLog)# clearSysLog review
UCS(sysLog)# clearSysLog commit
UCS(sysLog)# exit
```

Verifying Syslog

Use the following command to verify the syslog status on the UCS controller.

- **showLogLevelConf**

  This command displays the syslog configuration status on the NID. The following is a sample output from the command:

```
UCS(sysLog)# showSysLog showLogLevelConf
UCS(sysLog)# showSysLog review
```

```
Commands in queue:
 showSysLog showLogLevelConf
```

```
UCS(sysLog)# showSysLog commit
```

```
Clearing Socket 0 Clearing Socket 0
ShowSysLog_Output.showLogLevelResponse.hostMode = true
ShowSysLog_Output.showLogLevelResponse.hostAddress = '10.78.101.221'
ShowSysLog_Output.showLogLevelResponse.logLevel = 'info'
ShowSysLog_Output.showLogLevelResponse.noOfLogEntries.InfoCounter =
40
ShowSysLog_Output.showLogLevelResponse.noOfLogEntries.warningCounter
 = 0
ShowSysLog_Output.showLogLevelResponse.noOfLogEntries.errorCounter =
 0

 ShowSysLog Commit Success!!!
```

CHAPTER 26

Configuring SPAN

This document describes the Switched Port Analyzer (SPAN) feature and configuration steps to implement SPAN.

Prerequisites for Configuring SPAN

- You must enable SPAN globally to support the desired SPAN configuration.
- NID must have an IP address.
- You must select a SPAN source from the following options:
  - Interface—one or more source interfaces.
  - VLAN— one or more source VLANs.
  - CPU— to monitor CPU traffic.

Restrictions for Configuring SPAN

- You cannot configure a port as both a source and destination port.
- VLAN SPAN monitors only the traffic that leaves or enters Layer 2 ports in the VLAN.
- SPAN sources interface and VLAN cannot exit together.

Information About SPAN

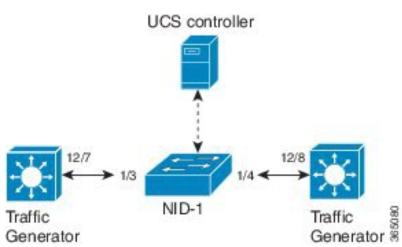
Switched Port Analyzer (SPAN) feature, sometimes called port mirroring or port monitoring, selects network traffic for analysis by a network analyzer. The SPAN feature is local when the monitored ports are all located on the same switch as the destination port. A local SPAN session is an association of a destination port with source ports. You can monitor incoming or outgoing traffic on a series or range of ports.

SPAN is used to monitor traffic within the switch. Traffic source can be from:

- Single or multiple ports
- Single or multiple VLANs
- Source CPU

Destination can be an interface on the same switch. The following figure shows the topology used for provisioning SPAN on a NID using a UPE NID Controller.

Figure 14: SPAN Topology



How to Provision SPAN

Enabling SPAN Globally to Start a Monitoring Session

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**span** **Example:** `UCS# span`	Enters the SPAN mode.
Step 3	**setSpanGlobalConfReq** {**enable** | **disbale**}	Enters SPAN global configuration mode.

	Command or Action	Purpose
	Example: `UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable`	Sub-command options. • **enable—**Enables SPAN globally. • **disbale—**Disables SPAN globally.
Step 4	**setSpanGlobalConf review** **Example:** `UCS(SPAN)# setSpanGlobalConf review`	(Optional) Displays the configuration.
Step 5	**setSpanGlobalConf commit** **Example:** `UCS(SPAN)# setSpanGlobalConf commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(SPAN)# exit`	Exits the Opens a new session for NID 1. mode.

Configuration Example

- The example shows how to enable SPAN globally:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
```

Configuring SPAN Source Interface

Before You Begin

Perform the steps to enable SPAN globally.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setSpanSrcConfRequest** {**source** {**cpu** {**rx** | **tx** | **both**} | {**vlan** *vlan_list*} | **interface** {*intf_range* | **traffic-type** {**rx** | **tx** | **both**}} **Example:** `UCS(SPAN)# setSpanSrcConf commitsetSpanSrcConf setSpanSrcConfRequest source interface intf_range 1-2`	Configures SPAN source interface. • **source—**Mirrors source interface or VLAN. • **cpu—**Mirrors source CPU. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic.

	Command or Action	Purpose
		• **both—**Mirrors received and transmitted traffic. • **vlan—**Mirrors source VLAN. • *vlan_list*—Mirrors source VLAN. • **interface—** Mirrors source interface and traffic type. • *intf_range*—Mirrors an interface number or a range from 1 to 6. • **traffic-type—**Mirrors traffic type. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic. • **both—**Mirrors received and transmitted traffic.
Step 2	**setSpanSrcConfRequest** {**source** {**cpu** {**rx** | **tx** | **both**} | {**vlan** vlan_list} | **interface** {*intf_range* | **traffic-type** {**rx** | **tx** | **both**}} **Example:** `UCS(SPAN)# setSpanSrcConf commitsetSpanSrcConf setSpanSrcConfRequest source interface traffic-type both`	Configures SPAN source traffic type as both, receive and transmit. • **source—**Mirrors source interface or VLAN. • **cpu—**Mirrors source CPU. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic. • **both—**Mirrors received and transmitted traffic. • **vlan—**Mirrors source VLAN. • *vlan_list*—Mirrors source VLAN. • **interface—** Mirrors source interface and traffic type. • *intf_range*—Mirrors an interface number or a range from 1 to 6. • **traffic-type—**Mirrors traffic type. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic. • **both—**Mirrors received and transmitted traffic.
Step 3	**setSpanSrcConf review** **Example:** `UCS(SPAN)# setSpanSrcConf review`	(Optional) Displays the configuration.
Step 4	**setSpanGlobalConf commit** **Example:** `UCS(SPAN)# setSpanSrcConf commit`	Sends the configuration to NID.

	Command or Action	Purpose
Step 5	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure SPAN on an interface range:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
UCS(SPAN)# setSpanSrcConf commitsetSpanSrcConf setSpanSrcConfRequest source interface
 intf_range 1-2
UCS(SPAN)# setSpanSrcConf commitsetSpanSrcConf setSpanSrcConfRequest source interface
 traffic-type both
UCS(SPAN)# setSpanSrcConf review
UCS(SPAN)# setSpanSrcConf commit
UCS(SPAN)# exit
```

Configuring SPAN Source CPU

Before You Begin

Perform the steps to enable SPAN globally.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setSpanSrcConfRequest** {**source** {**cpu** {**rx** | **tx** | **both**} | {**vlan** *vlan_list*} | **interface** {*intf_range* | **traffic-type** {**rx** | **tx** | **both**}} **Example:** `UCS(SPAN)# setSpanSrcConf setSpanSrcConfRequest source cpu both`	Configures SPAN source CPU. • **source—**Mirrors source interface or VLAN. • **cpu—**Mirrors source CPU. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic. • **both—**Mirrors received and transmitted traffic. • **vlan—**Mirrors source VLAN. • *vlan_list*—Mirrors source VLAN. • **interface—** Mirrors source interface and traffic type.

	Command or Action	Purpose
		• *intf_range*—Mirrors an interface number or a range from 1 to 6. • **traffic-type—**Mirrors traffic type. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic. • **both—**Mirrors received and transmitted traffic.
Step 2	**setSpanSrcConf review** **Example:** `UCS(SPAN)# setSpanSrcConf review`	(Optional) Displays the configuration.
Step 3	**setSpanGlobalConf commit** **Example:** `UCS(SPAN)# setSpanSrcConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure SPAN on an interface range:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
UCS(SPAN)# setSpanSrcConf setSpanSrcConfRequest source cpu both
UCS(SPAN)# setSpanSrcConf review
UCS(SPAN)# setSpanSrcConf commit
UCS(SPAN)# exit
```

Configuring SPAN Source VLAN

Before You Begin

Perform the steps to enable SPAN globally.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setSpanSrcConfRequest** {**source** {**cpu** {**rx** | **tx** | **both**} | {**vlan** *vlan_list*} | **interface** {*intf_range* | **traffic-type** {**rx** | **tx** | **both**}} **Example:** `UCS(SPAN)# setSpanSrcConf setSpanSrcConfRequest source vlan vlan_list 100`	Configures SPAN source VLAN. • **source—**Mirrors source interface or VLAN. • **cpu—**Mirrors source CPU. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic. • **both—**Mirrors received and transmitted traffic. • **vlan—**Mirrors source VLAN. • *vlan_list*—Mirrors source VLAN. • **interface—** Mirrors source interface and traffic type. • *intf_range*—Mirrors an interface number or a range from 1 to 6. • **traffic-type—**Mirrors traffic type. • **rx—**Mirrors received traffic. • **tx—**Mirrors transmitted traffic. • **both—**Mirrors received and transmitted traffic.
Step 2	**setSpanSrcConf review** **Example:** `UCS(SPAN)# setSpanSrcConf review`	(Optional) Displays the configuration.
Step 3	**setSpanGlobalConf commit** **Example:** `UCS(SPAN)# setSpanSrcConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure SPAN on an interface range:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
```

```
UCS(SPAN)# setSpanSrcConf setSpanSrcConfRequest source vlan vlan_list 100
UCS(SPAN)# setSpanSrcConf review
UCS(SPAN)# setSpanSrcConf commit
UCS(SPAN)# exit
```

Configuring SPAN Destination

Before You Begin

Perform the steps to enable SPAN globally.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setSpanDestConfRequest destination intf_id** **Example:** `UCS(SPAN)# setSpanDestConf setSpanDestConfRequest destination intf_id 4`	Configures SPAN destination. • **destination—**Mirrors destination interface. • **intf_id—**Specifies single port ID range from 1 to 6.
Step 2	**setSpanDestConf review** **Example:** `UCS(SPAN)# setSpanDestConf review`	(Optional) Displays the configuration.
Step 3	**setSpanDestConf commit** **Example:** `UCS(SPAN)# setSpanDestConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure SPAN destination:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
UCS(SPAN)# setSpanDestConf setSpanDestConfRequest destination intf_id 4
UCS(SPAN)# setSpanDestConf review
UCS(SPAN)# setSpanDestConf commit
UCS(SPAN)# exit
```

Deleting SPAN Source Configuration

Before You Begin

Perform the steps to enable SPAN globally.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**delSpanSrcConfRequest** {**source** {**cpu** {**rx** | **tx** | **both**} | {**vlan** *vlan_list*} | **interface** {*intf_range* | **traffic-type** {**rx** | **tx** | **both**}} **Example:** `UCS(SPAN)# delSpanSrcConf delSpanSrcConfRequest source cpu rx`	Deletes SPAN source configuration. • **source**—Removes mirror of source interface or VLAN. • **cpu**—Removes mirror of source CPU. • **rx**—Removes mirror of received traffic. • **tx**—Removes mirror of transmitted traffic. • **both**—Removes mirror of received and transmitted traffic. • **vlan**—Removes mirror of source VLAN. • *vlan_list*—Removes mirror of source VLAN. • **interface**— Removes mirror of source interface and traffic type. • *intf_range*—Removes mirror of interface number or a range from 1 to 6. • **traffic-type**—Removes mirror of traffic type. • **rx**—Removes mirror of received traffic. • **tx**—Removes mirror of transmitted traffic. • **both**—Removes mirror of received and transmitted traffic.
Step 2	**delSpanSrcConf review** **Example:** `UCS(SPAN)# delSpanSrcConf review`	(Optional) Displays the configuration.
Step 3	**delSpanSrcConf commit** **Example:** `UCS(SPAN)# delSpanSrcConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure SPAN on an interface range:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
UCS(SPAN)# delSpanSrcConf delSpanSrcConfRequest source cpu rx
UCS(SPAN)# delSpanSrcConf review
UCS(SPAN)# delSpanSrcConf commit
UCS(SPAN)# exit
```

Deleting SPAN Destination Configuration

Before You Begin

Perform the steps to enable SPAN globally.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**delSpanDestConfRequest destination intf_id** **Example:** `UCS(SPAN)# delSpanDstConf delSpanDstConfRequest detination intf_id 4`	Deletes SPAN destination configuration. • **destination—**Removes mirror of destination interface. • **intf_id—**Specifies single port ID range from 1 to 6.
Step 2	**delSpanDstConf review** **Example:** `UCS(SPAN)# delSpanDstConf review`	(Optional) Displays the configuration.
Step 3	**delSpanDstConf commit** **Example:** `UCS(SPAN)# delSpanDstConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure SPAN destination:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
```

```
UCS(SPAN)# exit
UCS(SPAN)# delSpanDstConf delSpanDstConfRequest detination intf_id 4
UCS(SPAN)# delSpanDstConf review
UCS(SPAN)# delSpanDstConf commit
UCS(SPAN)# exit
```

Verifying Diagnostics POST

Use the following commands to verify the diagnostics test status.

- **showDiagResults showDiagTestResults**

  The following is a sample output from the command:

```
UCS(Diagnostics)#  showDiagResults showDiagTestResults
UCS(Diagnostics)#  showDiagResults review
```

```
Commands in queue:
                   showDiagResults showDiagTestResults
```

```
UCS(Diagnostics)#  showDiagResults commit
```

```
ShowDiagResults_Output.diagTestResults.testresult[0] = 'External Port
 Loopback Test =>'
ShowDiagResults_Output.diagTestResults.testresult[1] = 'Passed'
ShowDiagResults_Output.diagTestResults.testresult[2] = 'Sync-E
Reference Source Clock Test =>'
ShowDiagResults_Output.diagTestResults.testresult[3] = 'Passed'
ShowDiagResults_Output.diagTestResults.testresult[4] = 'PTP One PPS
Test =>'
ShowDiagResults_Output.diagTestResults.testresult[5] = 'Passed'
 ShowDiagResults Commit Success!!!
```

Additional References

Related Documents

Related Topic	Document Title
Cisco ME 3800x and ME 3600x Switches Software Configuration Guide, Cisco IOS Release 15.4(1)S	http://www.cisco.com/c/en/us/td/docs/switches/metro/me3600x_3800x/software/release/15-4_1_S/configuration/guide/3800x3600xscg.html

MIBs

MIB	MIBs Link
MIBs Supporting Cisco IOS	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

CHAPTER 27

Configuring RSPAN

This document describes the Remote Switched Port Analyzer (RSPAN) feature and configuration steps to implement RSPAN.

Prerequisites for Configuring RSPAN

- You must enable SPAN globally to support the desired SPAN configuration.
- NID must have an IP address.
- You must select a SPAN source from the following options:
    - Interface—one or more source interfaces.
    - VLAN— one or more source VLANs.
    - CPU— to monitor CPU traffic.

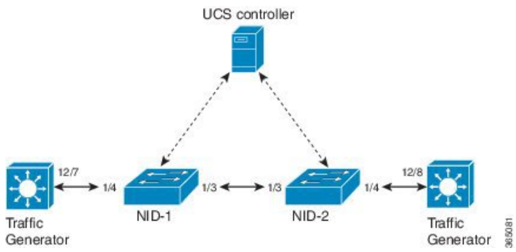
Restrictions for Configuring RSPAN

- You cannot configure a port as both a source and destination port.
- VLAN SPAN monitors only the traffic that leaves or enters Layer 2 ports in the VLAN.
- SPAN sources interface and VLAN cannot exit together.

Information About RSPAN

Remote Switched Port Analyzer (RSPAN) is an advanced feature that requires a special VLAN to carry the traffic that is monitored by SPAN between switches. RSPAN is useful when source ports are not located on the same switch as the destination port. The following figure shows the topology used for provisioning RSPAN on two NIDs using the UCS Controller.

Figure 15: RSPAN Topology



How to Provision RSPAN

Enabling SPAN Globally to Start a Monitoring Session

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**span** **Example:** `UCS# span`	Enters the SPAN mode.
Step 3	**setSpanGlobalConfReq** {**enable** | **disbale**} **Example:** `UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable`	Enters SPAN global configuration mode. Sub-command options. • **enable**—Enables SPAN globally. • **disbale**—Disables SPAN globally.
Step 4	**setSpanGlobalConf review** **Example:** `UCS(SPAN)# setSpanGlobalConf review`	(Optional) Displays the configuration.

	Command or Action	Purpose
Step 5	**setSpanGlobalConf commit** **Example:** `UCS(SPAN)# setSpanGlobalConf commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(SPAN)# exit`	Exits the Opens a new session for NID 1. mode.

Configuration Example

- The example shows how to enable SPAN globally:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
```

Configuring SPAN Source Interface on NID-1

Before You Begin

Perform the steps to enable SPAN globally. See Enabling SPAN Globally to Start a Monitoring Session.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setSpanSrcConfRequest** {**source** {**cpu** {**rx** | **tx** | **both**} | {**vlan** vlan_list} | **interface** {*intf_range* | **traffic-type** {**rx** | **tx** | **both**} **Example:** `UCS(SPAN)# setSpanSrcConf commitsetSpanSrcConf setSpanSrcConfRequest source interface intf_range 1/4`	Configures SPAN source interface. • **source**—Mirrors source interface or VLAN. • **cpu**—Mirrors source CPU. • **rx**—Mirrors received traffic. • **tx**—Mirrors transmitted traffic. • **both**—Mirrors received and transmitted traffic. • **vlan**—Mirrors source VLAN. • *vlan_list*—Mirrors source VLAN. • **interface**— Mirrors source interface and traffic type. • *intf_range*—Mirrors an interface number or a range from 1 to 6. • **traffic-type**—Mirrors traffic type.

	Command or Action	Purpose
		• **rx**—Mirrors received traffic. • **tx**—Mirrors transmitted traffic. • **both**—Mirrors received and transmitted traffic.
Step 2	**setSpanSrcConf review** **Example:** `UCS(SPAN)# setSpanSrcConf review`	(Optional) Displays the configuration.
Step 3	**setSpanGlobalConf commit** **Example:** `UCS(SPAN)# setSpanSrcConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure SPAN source on NID-1:

```
UCS(SPAN)# setSpanSrcConf commitsetSpanSrcConf setSpanSrcConfRequest source interface
 intf_range 1/4
UCS(SPAN)# setSpanSrcConf review
UCS(SPAN)# setSpanSrcConf commit
UCS(SPAN)# exit
```

Configuring Destination VLAN on NID-1

Before You Begin

Perform the steps to configure SPAN source on NID-1. See Configuring SPAN Source Interface on NID-1.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setrSpandestConf setRSpanDestConfRequest remote** *vlan_id* **Example:** `UCS(SPAN)# setrSpandestConf setRSpanDestConfRequest remote vlan_id 500`	Configures destination VLAN. • **remote**—Mirrors remote destination. • *vlan_id*— Remote mirror destination VLAN number.

	Command or Action	Purpose
Step 2	**setrSpandestConf review** **Example:** `UCS(SPAN)# setrSpandestConf review`	(Optional) Displays the configuration.
Step 3	**setrSpandestConf commit** **Example:** `UCS(SPAN)# setrSpandestConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure destination VLAN on NID-1:

```
UCS(SPAN)# setrSpandestConf setRSpanDestConfRequest remote vlan_id 500
UCS(SPAN)# setrSpandestConf review
UCS(SPAN)# setrSpandestConf commit
UCS(SPAN)# exit
```

Configuring Source VLAN on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**span** **Example:** `UCS# span`	Enters the SPAN mode.
Step 3	**setrSpansrcConf setRSpanSrcConfRequest remote** *vlan_id* **Example:** `UCS(SPAN)# setrSpansrcConf setRSpanSrcConfRequest remote vlan_id 500`	Configures RSPAN source. • **remote—**Mirrors remote source. • *vlan_id*— Remote mirror source VLAN number.

	Command or Action	Purpose
Step 4	**setrSpansrcConf review** **Example:** `UCS(SPAN)# setrSpansrcConf review`	(Optional) Displays the configuration.
Step 5	**setrSpansrcConf commit** **Example:** `UCS(SPAN)# setrSpansrcConf commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure source VLAN on NID-2:

```
UCS# span
UCS(SPAN)# setSpanGlobalConf setSpanGlobalConfReq enable
UCS(SPAN)# setSpanGlobalConf review
UCS(SPAN)# setSpanGlobalConf commit
UCS(SPAN)# exit
UCS(SPAN)# setrSpansrcConf setRSpanSrcConfRequest remote vlan_id 500
UCS(SPAN)# setrSpansrcConf review
UCS(SPAN)# setrSpansrcConf commit
UCS(SPAN)# exit
```

Configuring Destination Interface on NID-2

Before You Begin

Perform the steps to configure source VLAN on NID-2. See Configuring Source VLAN on NID-2.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**setSpanDestConf setSpanDestConfRequest destination** *intf_id* **Example:** `UCS(SPAN)# setSpanDestConf setSpanDestConfRequest destination intf_id 5`	Configures destination interface. • **destination—**Mirrors destination interface. • *intf_id*— Single port ID from 1 to 6.
Step 2	**setSpanDestConf review** **Example:** `UCS(SPAN)# setSpanDestConf review`	(Optional) Displays the configuration.

	Command or Action	Purpose
Step 3	**setSpanDestConf commit** **Example:** `UCS(SPAN)# setSpanDestConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to configure destination VLAN on NID-1:

```
UCS(SPAN)# setSpanDestConf setSpanDestConfRequest destination intf_id 5
UCS(SPAN)# setSpanDestConf review
UCS(SPAN)# setSpanDestConf commit
UCS(SPAN)# exit
```

Deleting RSPAN Source Configuration on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**delRSpanSrcConfRequest remote** *vlan_id* **Example:** `UCS(SPAN)# delRSpanSrcConf delRSpanSrcConfRequest remote vlan_id 500`	Deletes RSPAN source configuration. • **remote**—Removes remote mirror source. • *vlan_id*— Removes remote mirror source VLAN number.
Step 2	**delRSpanSrcConf review** **Example:** `UCS(SPAN)# delRSpanSrcConf review`	(Optional) Displays the configuration.
Step 3	**delRSpanSrcConf commit** **Example:** `UCS(SPAN)# delRSpanSrcConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to delete RSPAN source configuration on NID-2:

```
UCS(SPAN)# delRSpanSrcConf delRSpanSrcConfRequest remote vlan_id 500
UCS(SPAN)# delRSpanSrcConf review
UCS(SPAN)# delRSpanSrcConf commit
UCS(SPAN)# exit
```

Deleting RSPAN Destination Configuration on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**delRSpanDstConfRequest remote** *vlan_id* **Example:** `UCS(SPAN)# delRSpanDstConf delRSpanDstConfRequest remote vlan_id 500`	Deletes RSPAN destination configuration. • **remote—**Removes remote mirror destination. • *vlan_id***—**Removes remote mirror destination VLAN number.
Step 2	**delSpanDstConf review** **Example:** `UCS(SPAN)# delRSpanDstConf review`	(Optional) Displays the configuration.
Step 3	**delSpanDstConf commit** **Example:** `UCS(SPAN)# delRSpanDstConf commit`	Sends the configuration to NID.
Step 4	**exit** **Example:** `UCS(SPAN)# exit`	Exits the SPAN mode.

Configuration Example

- The example shows how to delete RSPAN destination configuration on NID-1:

```
UCS(SPAN)# delRSpanDstConf delRSpanDstConfRequest remote vlan_id 500
UCS(SPAN)# delRSpanDstConf review
UCS(SPAN)# delRSpanDstConf commit
UCS(SPAN)# exit
```

Verifying RSPAN

Use the following commands to verify the RSPAN status on the UCS controller.

- **showSpanConfig showSpanConfigReq**

This command displays the SPAN configuration status on the NID, when source interface is 1/4 and traffic type is both. The following is a sample output from the command:

```
UCS(SPAN)# showSpanConfig showSpanConfigReq
UCS(SPAN)# showSpanConfig review
```

```
Commands in queue:
    showSpanConfig showSpanConfigReq
```

```
UCS(SPAN)# showSpanConfig commit
```

```
ShowSpanConfig_Output.showSpanConfigResp.span_config[0] = 'Session:
1, Mode: Disabled'
ShowSpanConfig_Output.showSpanConfigResp.span_config[1] = 'Type: Remote
 Source Session'
ShowSpanConfig_Output.showSpanConfigResp.span_config[2] = 'Dest RMIRROR
 VLAN: 500'
ShowSpanConfig_Output.showSpanConfigResp.span_config[3] = 'Source
VLAN(s): '
ShowSpanConfig_Output.showSpanConfigResp.span_config[4] = 'Source
port(s): 1/5'
ShowSpanConfig_Output.showSpanConfigResp.span_config[5] = 'Traffic
Type: '
ShowSpanConfig_Output.showSpanConfigResp.span_config[6] = 'rx : 1/5'
ShowSpanConfig_Output.showSpanConfigResp.span_config[7] = 'Destination
 Ports: 1/4'
ShowSpanConfig Commit Success!!!
```

Additional References

Related Documents

Related Topic	Document Title
Cisco ME 3800x and ME 3600x Switches Software Configuration Guide, Cisco IOS Release 15.4(1)S	http://www.cisco.com/c/en/us/td/docs/switches/metro/me3600x_3800x/software/release/15-4_1_S/configuration/guide/3800x3600xscg.html

MIBs

MIB	MIBs Link
MIBs Supporting Cisco IOS	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

CHAPTER 28

Configuring RFC 2544

This document describes the RFC 2544 feature and configuration steps to implement RFC 2544.

Prerequisites for Configuring RFC 2544

- You must disable:
    - Link Layer Discovery Protocol (LLDP) transmit and receive on source port.
    - Loop protection on destination port or Spanning Tree Protocol (STP) on destination and source port.
- You must create:
    - Traffic test loop on destination port.
    - RFC 2544 profile with source port specified.
- There should be no traffic coming in or out of the ports.
- NID must have an IP address.

Restrictions for Configuring RFC 2544

- Ethernet Virtual Circuit (EVC) Maintenance End Points (MEP) is not supported.

•

Information About RFC 2544

RFC 2544 defines a number of tests that can be used to describe the performance characteristics of a network interconnect devices. These tests certify that a Service Level Agreement (SLA) between a customer and a service provider is met.

You can perform RFC 2544 benchmark tests on Carrier Ethernet switch platforms running ME 1200 software without the need for any external test equipment.

The RFC 2544 benchmarking can be run on a Metro Ethernet and offers a variety of diagnosis, such as:

- Throughput—Measures the maximum rate at which none of the offered frames are dropped on the device.
- Back-to-back—Measures the buffering capacity of a device.
- Frame loss—Measures the performance of a network device in an overloaded state.
- Latency—Measures the round-trip time taken by a test frame to travel through a network device or across the network and back to the test port.

In addition, the ME 1200 software includes a test suite tool that allows creating, saving, and executing test profiles and capturing and reporting results. The Local Node acts as a frame generator and checker.

Note For RFC 2544 to function properly, the Remote Node must support looping of particular frames.

The RFC 2544 benchmarking can be done either on the Port MEP or Virtual Local Area Network (VLAN) MEP. The following figure shows the topology used for provisioning RFC 2544 on two NIDs using the UCS Controller.

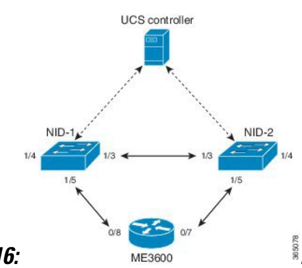


Figure 16: RFC 2544 Topology

Before executing RFC 2544 test, you must prepare a test profile. The RFC 2544 test profile contains all the parameters associated with one test, where *one test* may be a combination of one or more sub-tests (Throughput, Latency, and Frame Loss, Back-to-Back).

Common and sub-test specific parameters in a test profile are listed below:

- **Common Parameters**
  - Profile Name— Name of each profile. Name can be up to 32 characters. Default name is New profile.
  - Profile Description—A text description up to 128 characters associated with the profile. Default description is blank.

- MEG Level— Maintenance Entity Group (MEG) level on which the RFC 2544 test is run. Default MEG level is 7.
- Egress Port—Egress port of the switch on which the RFC 2544 test frames are generated and checked.
- Sequence Number Check—Checks generated frame sequence number. Default is Disabled.
- Dwell Time—Number of seconds to wait after each trial for the system to settle before reading statistics from the hardware. Default is 2 seconds.
- Type—Selects between two types of traffic: and VLAN-based . With VLAN-based , a configurable VLAN tag is inserted in the generated test frames.
- VLAN ID—Specifies the VLAN ID if VLAN-based is configured.
- PCP—Specifies the PCP value if VLAN-based is configured.
- DEI—Specifies the DEI value if VLAN-based is configured.
- DMAC—Specifies the DMAC of the generated frames for both Port-based and VLAN-based .
- Frame Size—Specifies the frame size each test must be repeated with, such as 64,128,256,512, 1024,1280,1518,2000, and 9600 bytes. Default frame size is all but 9600.
- Sub-Tests To Run—Specifies the sub-tests to be run in the profile (Throughput, Latency, Frame Loss, Back-to-Back). Default sub-tests to run is Throughput and Latency.

- **Throughput Test Parameters**
  - Trial Duration—Duration of a trial run in seconds. Valid range is from 1 to 1800 seconds. Default trial duration is 60 seconds.
  - Minimum and Maximum Rate—Specifies the maximum and minimum search rates.
  - Rate Step—Specifies the granularity of search within the minimum and maximum rates define above. All three input parameters are specified in % of the egress port's actual link speed and must be in the range from 1 to 1000% with a granularity of 1%. Default rate step is Minimum: 800% of link speed, Maximum: 1000% of link speed, and Step size: 20% of link speed.
  - Allowed Frame Loss—Specifies the allowable frame loss. Valid value is in range is from 0 to 100% with a granularity of 1%. Default allowable frame loss is 0.
- **Latency Test Parameters**
  - Trial Duration—Duration of a trial run in seconds. Valid range is from 10 to 1800 seconds. Default trial duration is 120 seconds.
  - Delay Measurement Interval—Specifies the number of seconds between each delay measurement. Valid range is from 1 to 60 seconds in steps of 1 second. Default delay measurement interval is 10 seconds.
  - Allowed Frame Loss—Specifies the pass criterion of an allowable frame loss. Valid range is from 0 to 10% with a granularity of 0.1%. Default allowed frame loss is 0.
- **Frame Loss Test Parameters**

  - Trial Duration—Duration of a trial run in seconds. Valid range is from 1 to 1800 seconds. Default trial duration is 60 seconds.
  - Minimum and Maximum Rate—Specifies the maximum and minimum search rates.
  - Rate Step—Specifies the granularity of search within the minimum and maximum rates define above. All three input parameters must be specified in % of the egress port's actual link speed and must be in the range from 1 to 1000% with a granularity of 1%. Default rate step is Minimum: 800%.

- **Back-to-Back Test Parameters**
  - Trial duration—Specifies the duration of a burst. Valid range is from 100 to 10000 milliseconds. Default trial duration is 2000 milliseconds.
  - Trial Count—Specifies the number of times the trial is executed. Valid range is from 1 to 100. Default trial count is 50. Up to 16 profiles can be created and saved in the switch flash memory.

RFC 2544 Test Report

On executing a RFC 2544 test profile, RFC 2544 test report is generated. The RFC 2544 test report is in clear text format and contains all the input parameters defined by the associated test profile and the measurement results. The RFC 2544 test report can be used to certify if an SLA is met.

The last 10 RFC 2544 test reports are stored in the Flash memory of the .

How to Provision RFC 2544

Disabling LLDP Port on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**setLldpportconfig lldpPortConfiguration** {lldp_receive_enable {**disable** | **enable**} | lldp_transmit_enable {**disable** | **enable**} | port_number *port_number*}	Sets the Link Layer Discovery Protocol (LLDP) port configuration. • **lldp_receive_enable**—Whether LLDP receive is enabled or disabled.

	Command or Action	Purpose

	Example:
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration port_number 3`
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_receive_enable disable`
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_transmit_enable disable`
		• **lldp_transmit_enable**—Whether LLDP transmit is enabled or disabled.
		• **port_number**—The target interface number. The valid values are from 1 to 6.

Step 4	**setLldpPortConfig review** **Example:** `UCS(ProvisionLldpPortType)# setLldpPortConfig review`	Displays the setLldpPortConfig configuration.
Step 5	**setLldpPortConfig commit** **Example:** `UCS(ProvisionLldpPortType)# setLldpPortConfig commit`	Sends the setLldpConfig configuration to the ME 1200 NID.
Step 6	**exit** **Example:** `UCS(ProvisionLldpPortType)# exit`	Exits the ProvisionLldpPortType mode.

Configuration Example

The example shows how to disable LLDP port on NID-1:

```
UCS# ProvisionLldpPortType
UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration port_number 3
UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_receive_enable
disable
UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_transmit_enable
disable
UCS(ProvisionLldpPortType)# setLldpPortConfig review
UCS(ProvisionLldpPortType)# setLldpPortConfig commit
UCS(ProvisionLldpPortType)# exit
```

Creating Layer 2 VLANs on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**createVlanCommand createVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 2-4095`	Creates the VLAN list. The valid values are from 1 to 4095.
Step 4	**createVlanCommand review** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand review`	Displays the createVlanCommand configuration.
Step 5	**createVlanCommand commit** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand commit`	Sends the createVlanCommand configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

The example shows how to create Layer 2 VLANs on NID-1:

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 2-4095
UCS(ProvisionPortVlanPortType)# createVlanCommand review
UCS(ProvisionPortVlanPortType)# createVlanCommand commit
UCS(ProvisionPortVlanPortType)# exit
```

Assigning VLANs to Ports on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**modifySwPort modifySWPortConfig mode access vlan** *vlan_id* **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 3`	Sets the mode to ACCESS, and assigns a VLAN.
Step 4	**modifySwPort modifySWPortConfig mode trunk** {**allowed vlan** {**add** {**all** | **vlan_list** *vlan_list* } | **remove** {**all** | **vlan_list** *vlan_list* }} | {**native vlan** *vlan_list* } **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2-4095`	Sets the mode to TRUNK. • **allowed—**Sets the allowed VLAN characteristics when interface is in trunk mode. • **add—**Adds either all VLANs or specified VLANs to the current list. • **remove—**Removes either all VLANs or specified VLANs from the current list. • *vlan_d*—Specifies the VLAN ID. The valid values are from 0 to 4095.
Step 5	**modifySwPort review** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort review`	Displays the modifySwPort configuration.
Step 6	**modifySwPort commit** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort commit`	Sends the modifySwPort configuration to the UCS controller.
Step 7	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

The example shows how to assign VLANs to ports on NID-1:

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interaface 3
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 3
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 2-4095
UCS(ProvisionPortVlanPortType)# modifySwPort review
UCS(ProvisionPortVlanPortType)# modifySwPort commit
UCS(ProvisionPortVlanPortType)# exit
```

Disabling Spanning-Tree Protocol on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**setStpglobalConfig stpGlobalConfig** {**edge** {**bpdu-filter** | **bpdu-guard**} {**enable** | **disable**} | **mode** {**mstp** | **rstp** | **stp**} {**enable** | **disable**} | **mst** {**forward-time** *Fwdtime* | **instance** *instance* {**active** {**enable** | **disable**} | **priority** *Prio* | **vlan** *WORD*} | **max-age** *Maxage* | **max-hops** *Maxhops* | **name** *Name* | **revision** *Revision* } | **port-number** *Port number* {**enable** | **disable**} | **recovery** *Interval* | **transmit** *hold-count* } **Example:** `UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig port-number 3 disable` **Note** If the spanning-tree mode is STP or RSTP, and if the priority for the software needs to be changed, you can change using mst instance 0 and priority.	Configures the spanning-tree global configuration. • **stpGlobalConfig**—Sets the spanning-tree global configuration. • **edge**—Configures the edge ports. ◦ **bpdu-filter**—Enables or disables the BPDU filter (stop BPDU tx/rx). ◦ **bpdu-guard**—Enables or disables the BPDU guard. • **mode**—Configures the STP protocol mode. ◦ **mstp**—Enables or disables the Multiple Spanning Tree (802.1s). ◦ **rstp**—Enables or disables the Rapid Spanning Tree (802.1w) ◦ **stp**—Enables or disables the Spanning Tree (802.1D). • **mst**—Configures the STP bridge instance. ◦ *Fwdtime*—Forward time. The range is from 4 to 30 seconds. ◦ *instance*—Instance. The range is from 0 to 7 where CIST=0, MST2=1 and so on. ◦ **active**—Enables or disables the instance. ◦ *Prio* —Specifies the priority. The range is from 0 to 61440 seconds. The range should be given in the sets of (0, 4096, 8192...) and so on. ◦ *WORD*—VLAN range. ◦ *Maxage*—Maximum age. The range is from 6 to 40 seconds.

	Command or Action	Purpose
		◦ *Maxhops*—Maximum hops. The range is from 6 to 40 hop counts. ◦ *Name*—Name of the bridge. You can use 32 characters to define. ◦ *Revision*—Revision. The range is from 0-65535 revisions. • **port-number**—Configures the port number in the range from 1 to 6. ◦ *Port number*—Port number. The range is from 1 to 6. ◦ **disable**—Disables the port-number. ◦ **enable**—Enables the port-number. • **recovery**—Configures the error recovery timeout. ◦ *Interval*—Interval. The range is from 30-86400 seconds. • **transmit**—Configures the BPDUs to transmit. ◦ *hold-count*—Maximum number of transmit BPDUs per second. The range is from 1 to 10 seconds.
Step 4	**setStpGlobalConfig review** **Example:** `UCS(ProvisionStpPortType)# setStpGlobalConfig review`	Displays the setStpGlobalConfig.
Step 5	**setStpGlobalConfig commit** **Example:** `UCS(ProvisionStpPortType)# setStpGlobalConfig commit`	Sends the setStpGlobalConfig configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

The example shows how to disable Spanning-Tree Protocol on NID-1:

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig port-number 3 disable
UCS(ProvisionStpPortType)# setStpGlobalConfig review
UCS(ProvisionStpPortType)# setStpGlobalConfig commit
UCS(ProvisionStpPortType)# exit
```

Disabling LLDP Port on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.

Step 3	**setLldpportconfig lldpPortConfiguration** {lldp_receive_enable {**disable** | **enable**} | lldp_transmit_enable {**disable** | **enable**} | port_number *port_number*}
	
	Example:
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration port_number 3`
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_receive_enable disable`
	`UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_transmit_enable disable`
		Sets the Link Layer Discovery Protocol (LLDP) port configuration.
		• **lldp_receive_enable—**Whether LLDP receive is enabled or disabled.
		• **lldp_transmit_enable—**Whether LLDP transmit is enabled or disabled.
		• **port_number—**The target interface number. The valid values are from 1 to 6.

Step 4	**setLldpPortConfig review** **Example:** `UCS(ProvisionLldpPortType)# setLldpPortConfig review`	Displays the setLldpPortConfig configuration.
Step 5	**setLldpPortConfig commit** **Example:** `UCS(ProvisionLldpPortType)# setLldpPortConfig commit`	Sends the setLldpConfig configuration to the UCS controlller.
Step 6	**exit** **Example:** `UCS(ProvisionLldpPortType)# exit`	Exits the ProvisionLldpPortType mode.

Configuration Example

The example shows how to disable LLDP port on NID-2:

```
UCS# ProvisionLldpPortType
UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration port_number 3
UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_receive_enable
disable
UCS(ProvisionLldpPortType)# setLldpPortConfig lldpPortConfiguration lldp_transmit_enable
disable
UCS(ProvisionLldpPortType)# setLldpPortConfig review
```

```
UCS(ProvisionLldpPortType)# setLldpPortConfig commit
UCS(ProvisionLldpPortType)# exit
```

Creating Layer 2 VLANs on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**createVlanCommand createVlanReq vlan_list** *vlan_list* **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 2-4095`	Creates the VLAN list. The valid values are from 1 to 4095.
Step 4	**createVlanCommand review** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand review`	Displays the createVlanCommand configuration.
Step 5	**createVlanCommand commit** **Example:** `UCS(ProvisionPortVlanPortType)# createVlanCommand commit`	Sends the createVlanCommand configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

The example shows how to create Layer 2 VLANs on NID-2:

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# createVlanCommand createVlanReq vlan_list 2-4095
UCS(ProvisionPortVlanPortType)# createVlanCommand review
UCS(ProvisionPortVlanPortType)# createVlanCommand commit
UCS(ProvisionPortVlanPortType)# exit
```

Assigning VLANs to Ports on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionPortVlanPortType** **Example:** `UCS# ProvisionPortVlanPortType`	Enters the ProvisionLldpPortType mode.
Step 3	**modifySwPort modifySWPortConfig interface** *interface_id* **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interface 3`	Modifies the switchport configuration on the defined interface.
Step 4	**modifySwPort modifySWPortConfig mode access vlan** *vlan_id* **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 3`	Sets the mode to ACCESS, and assigns a VLAN.
Step 5	**modifySwPort modifySWPortConfig mode trunk** {**allowed vlan** {**add** {**all** | **vlan_list** *vlan_list* } | **remove** {**all** | **vlan_list** *vlan_list* }} | {**native vlan** *vlan_list* } **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add vlan_list 2-4095`	Sets the mode to TRUNK. • **allowed—**Sets the allowed VLAN characteristics when interface is in trunk mode. • **add—**Adds either all VLANs or specified VLANs to the current list. • **remove—**Removes either all VLANs or specified VLANs from the current list. • *vlan_d*—Specifies the VLAN ID. The valid values are from 0 to 4095.
Step 6	**modifySwPort review** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort review`	Displays the modifySwPort configuration.
Step 7	**modifySwPort commit** **Example:** `UCS(ProvisionPortVlanPortType)# modifySwPort commit`	Sends the modifySwPort configuration to the UCS controller.

	Command or Action	Purpose
Step 8	**exit** **Example:** `UCS(ProvisionPortVlanPortType)# exit`	Exits the ProvisionPortVlanPortType mode.

Configuration Example

The example shows how to assign VLANs to ports on NID-2:

```
UCS# ProvisionPortVlanPortType
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig interaface 3
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk native vlan 3
UCS(ProvisionPortVlanPortType)# modifySwPort modifySWPortConfig mode trunk allowed vlan add
 vlan_list 2-4095
UCS(ProvisionPortVlanPortType)# modifySwPort review
UCS(ProvisionPortVlanPortType)# modifySwPort commit
UCS(ProvisionPortVlanPortType)# exit
```

Disabling Spanning-Tree Protocol on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**ProvisionLldpPortType** **Example:** `UCS# ProvisionLldpPortType`	Enters the ProvisionLldpPortType mode.

Step 3	**setStpglobalConfig stpGlobalConfig** {**edge** {**bpdu-filter** | **bpdu-guard**} {**enable** | **disable**} | **mode** {**mstp** | **rstp** | **stp**} {**enable** | **disable**} | **mst** {**forward-time** *Fwdtime* | **instance** *instance* {**active** {**enable** | **disable**} | **priority** *Prio* | **vlan** *WORD*} | **max-age** *Maxage* | **max-hops** *Maxhops* | **name** *Name* | **revision** *Revision* } | **port-number** *Port number* {**enable** | **disable**} | **recovery** *Interval* | **transmit** *hold-count* }
	
	Example:
	`UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig port-number 3 disable`
		Configures the spanning-tree global configuration.
		• **stpGlobalConfig—**Sets the spanning-tree global configuration.
		• **edge—**Configures the edge ports.
		  ◦ **bpdu-filter—**Enables or disables the BPDU filter (stop BPDU tx/rx).
		  ◦ **bpdu-guard—**Enables or disables the BPDU guard.
		• **mode—**Configures the STP protocol mode.
		  ◦ **mstp—**Enables or disables the Multiple Spanning Tree (802.1s).

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td></td><td>Note If the spanning-tree mode is STP or RSTP, and if the priority for the software needs to be changed, you can change using mst instance 0 and priority.</td><td>

rstp—Enables or disables the Rapid Spanning Tree (802.1w)
stp—Enables or disables the Spanning Tree (802.1D).

mst—Configures the STP bridge instance.

Fwdtime—Forward time. The range is from 4 to 30 seconds.
instance—Instance. The range is from 0 to 7 where CIST=0, MST2=1 and so on.

active—Enables or disables the instance.
Prio —Specifies the priority. The range is from 0 to 61440 seconds. The range should be given in the sets of (0, 4096, 8192...) and so on.
WORD—VLAN range.

Maxage—Maximum age. The range is from 6 to 40 seconds.
Maxhops—Maximum hops. The range is from 6 to 40 hop counts.
Name—Name of the bridge. You can use 32 characters to define.
Revision—Revision. The range is from 0-65535 revisions.

port-number—Configures the port number in the range from 1 to 6.

Port number—Port number. The range is from 1 to 6.
disable—Disables the port-number.
enable—Enables the port-number.

recovery—Configures the error recovery timeout.

Interval—Interval. The range is from 30-86400 seconds.

transmit—Configures the BPDUs to transmit.

hold-count—Maximum number of transmit BPDUs per second. The range is from 1 to 10 seconds.

</td></tr>
<tr><td>Step 4</td><td>setStpGlobalConfig review

Example:
<code>UCS(ProvisionStpPortType)#
setStpGlobalConfig review</code></td><td>Displays the setStpGlobalConfig.</td></tr>
</table>

	Command or Action	Purpose
Step 5	**setStpGlobalConfig commit** **Example:** `UCS(ProvisionStpPortType)# setStpGlobalConfig commit`	Sends the setStpGlobalConfig configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(ProvisionStpPortType)# exit`	Exits the ProvisionStpPortType mode.

Configuration Example

The example shows how to disable Spanning-Tree Protocol on NID-2:

```
UCS# ProvisionStpPortType
UCS(ProvisionStpPortType)# setStpGlobalConfig stpGlobalConfig port-number 3 disable
UCS(ProvisionStpPortType)# setStpGlobalConfig review
UCS(ProvisionStpPortType)# setStpGlobalConfig commit
UCS(ProvisionStpPortType)# exit
```

Creating Port MEP Profile on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.

Step 3	**setRfc2544Profile Rfc2544Profile** {**profileName** | **description** | **megLevel** | **egressPort** | **seqNoCheck** {**enable** | **disable**} | **dwellTime** | **mepType** {**portDownMep** | **vlanDownMep**} | **vlanId** | **pcp** | **dei** | **dMac**}
	
	Example:
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile profileName profile1`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile description profile1`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile egressPort 3`
		Creates Port MEP profile.
		• **profileName—**Specifies RFC 2544 profile name.
		• **description—**Adds a description to profile.
		Note We recommend that you add a description for the profile.
		• **megLevel—**Sets profile MEG level used in TST PDUs.
		• **egressPort—**Sets the egress interface on which PDUs are transmitted.

	Command or Action	Purpose

	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile megLevel 5`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile mepType portDownMep`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile seqNoCheck disable`
		• **seqNoCheck—**Enables sequence number checking of looped TST PDUs.
		◦ **enable—**Enables sequence number.
		◦ **disable—**Disables sequence number.
		• **dwellTime—**Controls the number of seconds that the execution pauses after each trial, before reading counters and status from hardware.
		• **mepType—**Specifies MEP type port. MEP or VLAN MEP.
		◦ **portDownMep—**Creates a port down MEP.
		◦ **vlanDownMep—**Creates a VLAN down MEP. All PDUs are then transmitted with a VLAN tag.
		• **vlanId—**Specifies VLAN ID incase of VLAN down MEP.
		• **pcp—**Specifies PCP value used in the VLAN tag incase of VLAN MEP.
		• **dei—**Specifies DEI value used in the VLAN tag incase vlan mep.
		• **dMac—**Specifies destination MAC address used in generation of the Y.1731 TST and 1DM frames.

Step 4	**setRfc2544Profile review** **Example:** `UCS(RFC2544PortType)# setRfc2544Profile review`	Displays the setRfc2544Profile.
Step 5	**setRfc2544Profile commit** **Example:** `UCS(RFC2544PortType)# setRfc2544Profile commit`	Sends the setRfc2544Profile configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to create Port MEP profile on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile profileName profile1
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile description profile1
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile egressPort 3
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile megLevel 5
```

```
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile mepType portDownMep
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile seqNoCheck disable
UCS(RFC2544PortType)# setRfc2544Profile review
UCS(RFC2544PortType)# setrfc2544profile commit
UCS(RFC2544PortType)# exit
```

Creating Traffic Test Loop on Destination Port on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType configuration mode.

Step 3	**setTrafficTestLoop trafficTestLoopConfig** {**instNum** | **adminState** {**enable** | **disable**} | **custVID** | **name** | **type** {**macLoop** | **oamLoop**} | **interface** | **direction** {**facility** | **terminal**} | **domain** {**evc** | **port** | **vlan**} | **flowId** | **level**}
	
	Example:
	`UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig interface 3`
	`UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig type macLoop`
	`UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig direction facility`
	
	`UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig domain port`
	`UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig adminState enable`
	
	`UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig instNum 1`
		Creates traffic test loop on destination port on NID-2.
		• **instNum—**Specifies the traffic-test-loop instance number.
		• **adminState—**Specifies the administrative state.
		  ◦ **enable—**Creates a loop if all required resources are available and operational state is up.
		  ◦ **disable—**Deletes the loop and operational state is down.
		• **custVID—**Only relevant for OAM-loop in EVC domain. Loops C-tagged customer frames with this specified VID in the EVC.
		• **name—**Specifies the traffic-test-loop name.
		• **type—**Specifies the type of the traffic-test-loop. Currently only MAC loop is supported.
		  ◦ **macLoop—**All frames in the flow are looped with MAC swap.
		  ◦ **oamLoop—**Y.1731 OAM aware and is looping the following:
		    ◦ Loopback Messages (LBM) and Loopback Replies (LBR)
		    ◦ Delay Measurement Message (DMM) and Delay Measurement Reply (DMR)
		• **interface—**Specifies the residence port of the traffic-test-loop.
		• **direction—**Specifies the direction of the traffic-test-loop.

	Command or Action	Purpose
		◦ **facility—**Specifies that this traffic-test-loop is pointing to the port. Looping is done from ingress to egress. ◦ **terminal—**Specifies that this traffic-test-loop is pointing to the forwarding plane. Looping is done from egress to ingress. **Note** The **terminal** option is not supported. • **domain—**The domain of the traffic-test-loop. ◦ **evc—**This traffic-test-loop is in the EVC domain. ◦ **port—**This traffic-test-loop is in the Port domain. ◦ **vlan—**This traffic-test-loop is in the VLAN domain. **Note** Only **port** domain is supported. • **flowId—**Specifies the EVC domain instance ID or VID in VLAN domain. • **level—**Specifies the Y.1731 OAM level of the traffic-test-loop. This is relevant only for OAM looping type traffic-test-loop.
Step 4	**setTrafficTestLoop review** **Example:** `UCS(RFC2544PortType)# setTrafficTestLoop review`	Displays the setTrafficTestLoop configuration.
Step 5	**setTrafficTestLoop commit** **Example:** `UCS(RFC2544PortType)# setTrafficTestLoop commit`	Sends the setTrafficTestLoop configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to create traffic test loop on destination port on NID-2:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig interface 3
UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig type macLoop
UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig direction facility
UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig domain port
UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig adminState enable
UCS(RFC2544PortType)# setTrafficTestLoop trafficTestLoopConfig instNum 1
UCS(RFC2544PortType)# setTrafficTestLoop review
```

```
UCS(RFC2544PortType)# setTrafficTestLoop commit
UCS(RFC2544PortType)# exit
```

Disabling Loop Protection on Destination Port on NID-2

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 2`	Opens a new session for NID 2.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType configuration mode.
Step 3	**deleteTrafficTestLoop deleteLoopConfig** {**trafficLoop** *instNum* |**loopPotect** *interface*} **Example:** `UCS(RFC2544PortType)# deleteTrafficTestLoop deleteLoopConfig loopPotect interface 3`	Disables loop protection on destination port on NID-2. • **deleteLoopConfig—**Deletes traffic test loop configuration. • **trafficLoop—**Deletes traffic test loop configuration. ◦ *instNum*—Specifies the traffic-test-loop instance number. • **loopPotect—**Deletes loop protection at port level. ◦ *interface*—Specifies the residence port of the traffic-test-loop.
Step 4	**deleteTrafficTestLoop review** **Example:** `UCS(RFC2544PortType)# deleteTrafficTestLoop review`	Displays the deleteTrafficTestLoop configuration.
Step 5	**deleteTrafficTestLoop commit** **Example:** `UCS(RFC2544PortType)# deleteTrafficTestLoop commit`	Sends the deleteTrafficTestLoop configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to disable loop protection on destination port on NID-2:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# deleteTrafficTestLoop deleteLoopConfig loopPotect interface 3
UCS(RFC2544PortType)# deleteTrafficTestLoop review
UCS(RFC2544PortType)# deleteTrafficTestLoop commit
UCS(RFC2544PortType)# exit
```

Setting RFC 2544 Reporting Parameters on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.

Step 3	**setReportParams rfc2544Reports reportAction** {**delete** *reportName* | **save** {*reportName* | *tftpPath*} | **start** {*reportName* | *profileName* | *description*} | **stop** *reportName* | **rename** {*oldName* | *newName*}}
	
	Example:
	`UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start profileName profile1`
	`UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start reportName profile1`
	`UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start description profile1`
		Sets RFC 2544 reporting parameters.
		• **reportAction—**Specifies action to be performed on the report.
		◦ **delete—**Deletes the existing report.
		◦ *reportName***—**Specifies name of the report.
		◦ **save—**Saves the existing report.
		◦ *reportName***—**Specifies the name of existing report.
		◦ *tftpPath***—**Specifies TFTP server URL tftp://server[:port]/path-to-file.
		◦ **start—**Starts profile execution
		◦ *reportName***—**Specifies unique name of the resulting report.
		◦ *profileName***—**Specifies name of the profile to execute.
		◦ *description***—**(Optional) Provides a description of the report.
		Note We recommend that you add a description for the report.

	Command or Action	Purpose
		◦ **stop**—Stops the report. ◦ *reportName*—Specifies name of the report to be stopped. ◦ **rename**—Renames the existing report. ◦ *oldName*—Specifies name of the old profile. ◦ *newName*—Specifies name of the new profile.
Step 4	**setReportParams review** **Example:** `UCS(RFC2544PortType)# setReportParams review`	Displays the setReportParams configuration.
Step 5	**setReportParams commit** **Example:** `UCS(RFC2544PortType)# setReportParams commit`	Sends the setReportParams configuration to the ME 1200 NID.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to set the RFC 2544 reporting parameters on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start profileName profile1
UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start reportName profile1
UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start description profile1
UCS(RFC2544PortType)# setReportParams review
UCS(RFC2544PortType)# setReportParams commit
UCS(RFC2544PortType)# exit
```

Displaying RFC 2544 Profile and Report on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.
Step 3	**showRfc2544 showRequest show** {**profiles** | **report**} **Example:** `UCS(RFC2544PortType)# showRfc2544 showRequest show profiles`	Displays RFC 2544 profile. • **show—**Displays profile or report. ◦ **profiles—**Displays profile information. ◦ **report—**Displays report information.
Step 4	**showRfc2544 review** **Example:** `UCS(RFC2544PortType)# showRfc2544 review`	Displays the showRfc2544 configuration.
Step 5	**showRfc2544 commit** **Example:** `UCS(RFC2544PortType)# showRfc2544 commit`	Sends the setRfc2544Profile configuration to the UCS controller.
Step 6	**showRfc2544 showRequest show** {**profiles** | **report**} **Example:** `UCS(RFC2544PortType)# showRfc2544 showRequest show report`	Displays RFC 2544 profile. • **show—**Displays profile or report. ◦ **profiles—**Displays profile information. ◦ **report—**Displays report information.
Step 7	**showRfc2544 review** **Example:** `UCS(RFC2544PortType)# showRfc2544 review`	Displays the showRfc2544 configuration.
Step 8	**showRfc2544 commit** **Example:** `UCS(RFC2544PortType)# showRfc2544 commit`	Sends the setRfc2544Profile configuration to the UCS controller.
Step 9	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to display RFC 2544 profile and report on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# showRfc2544 showRequest show profiles
```

```
UCS(RFC2544PortType)# showRfc2544 review
UCS(RFC2544PortType)# showRfc2544 commit
```

```
ShowRfc2544_Output.showResponse.t = 1
ShowRfc2544_Output.showResponse.u.profile[0].profileName = 'profile1'
ShowRfc2544_Output.showResponse.u.profile[0].description = 'profile1'

 ShowRfc2544 Commit Success!!!
```

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# showRfc2544 showRequest show report
UCS(RFC2544PortType)# showRfc2544 review
UCS(RFC2544PortType)# showRfc2544 commit
```

```
ShowRfc2544_Output.showResponse.t = 2
ShowRfc2544_Output.showResponse.u.report[0].reportName = 'Report1'
ShowRfc2544_Output.showResponse.u.report[0].created =
'1970-01-04T07:29:25+00:00'
ShowRfc2544_Output.showResponse.u.report[0].status = 'Succeeded'
ShowRfc2544_Output.showResponse.u.report[1].reportName = 'Rep15'
ShowRfc2544_Output.showResponse.u.report[1].created =
'1970-01-02T01:57:34+00:00'
ShowRfc2544_Output.showResponse.u.report[1].status = 'Failed'
ShowRfc2544_Output.showResponse.u.report[2].reportName = 'Rep16'
ShowRfc2544_Output.showResponse.u.report[2].created =
'1970-01-02T02:08:12+00:00'
ShowRfc2544_Output.showResponse.u.report[2].status = 'Succeeded'
ShowRfc2544_Output.showResponse.u.report[3].reportName = 'profile1'
ShowRfc2544_Output.showResponse.u.report[3].created =
'1970-01-02T03:48:16+00:00'
ShowRfc2544_Output.showResponse.u.report[3].status = 'Failed'

 ShowRfc2544 Commit Success!!!
```

```
UCS(RFC2544PortType)# exit
```

Creating VLAN Profile on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.
Step 3	**setRfc2544Profile Rfc2544Profile** {**profileName** | **description** | **megLevel** | **egressPort** | **seqNoCheck** {**enable** | **disable**} | **dwellTime** | **mepType**	Creates RFC profile. • **profileName—**Specifies RFC 2544 profile name.

	Command or Action	Purpose

	{**portDownMep** | **vlanDownMep**} | **vlanId** | **pcp** | **dei** | **dMac**}
	
	Example:
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile profileName vlan-profile`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile egressPort 3`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile mepType vlanDownMep`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile vlanId 999`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile megLevel 4`
	`UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile description vlanprofile`
		• **description**—Adds a description to profile.
		Note We recommend that you add a description for the profile.
		• **megLevel**—Sets profile MEG level used in TST PDUs.
		• **egressPort**—Sets the egress interface on which PDUs are transmitted.
		• **seqNoCheck**—Enables sequence number checking of looped TST PDUs.
		◦ **enable**—Enables sequence number.
		◦ **disable**—Disables sequence number.
		• **dwellTime**—Controls the number of seconds that the execution pauses after each trial, before reading counters and status from hardware.
		• **mepType**—Specifies MEP type port. MEP or VLAN MEP.
		◦ **portDownMep**—Creates a port down MEP.
		◦ **vlanDownMep**—Creates a VLAN down MEP. All PDUs are then transmitted with a VLAN tag.
		• **vlanId**—Specifies VLAN ID incase of VLAN down MEP.
		• **pcp**—Specifies PCP value used in the VLAN tag incase of VLAN MEP.
		• **dei**—Specifies DEI value used in the VLAN tag incase vlan mep.
		• **dMac**—Specifies destination MAC address used in generation of the Y.1731 TST and 1DM frames.

Step 4	**setRfc2544Profile review** **Example:** `UCS(RFC2544PortType)# setRfc2544Profile review`	Displays the setRfc2544Profile.
Step 5	**setRfc2544Profile commit** **Example:** `UCS(RFC2544PortType)# setRfc2544Profile commit`	Sends the setRfc2544Profile configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to create VLAN profile on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile profileName vlan-profile
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile egressPort 3
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile mepType vlanDownMep
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile vlanId 999
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile megLevel 4
UCS(RFC2544PortType)# setRfc2544Profile Rfc2544Profile description vlanprofile
UCS(RFC2544PortType)# setRfc2544Profile review
UCS(RFC2544PortType)# setrfc2544profile commit
UCS(RFC2544PortType)# exit
```

Getting RFC 2544 Profile for VLAN on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.
Step 3	**getRfc2544Profile rfc2544Request profileName** *profileName* **Example:** `UCS(RFC2544PortType)# getRfc2544Profile rfc2544Request profileName vlan-profile`	Gets the RFC 2544 profile. • **rfc2544Request—**Specifies RFC2544 request parameter. • **profileName—**Specifies name of the profile.
Step 4	**getRfc2544Profile review** **Example:** `UCS(RFC2544PortType)# getRfc2544Profile review`	Displays the getRfc2544Profile configuration.
Step 5	**getRfc2544Profile commit** **Example:** `UCS(RFC2544PortType)# getRfc2544Profile commit`	Sends the getRfc2544Profile configuration to the ME 1200 NID.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to get RFC 2544 profile for VLAN on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# getRfc2544Profile rfc2544Request profileName vlan-profile
UCS(RFC2544PortType)# getRfc2544Profile review
UCS(RFC2544PortType)# getRfc2544Profile commit
```

```
GetRfc2544Profile_Output.Rfc2544Profile.profileName = 'vlan_profile'
GetRfc2544Profile_Output.Rfc2544Profile.description = 'vlanprofile'
GetRfc2544Profile_Output.Rfc2544Profile.megLevel = 4
GetRfc2544Profile_Output.Rfc2544Profile.egressPort = 3
GetRfc2544Profile_Output.Rfc2544Profile.seqNoCheck.t = 2
GetRfc2544Profile_Output.Rfc2544Profile.seqNoCheck.u.disable = ''
GetRfc2544Profile_Output.Rfc2544Profile.dwellTime = 2
GetRfc2544Profile_Output.Rfc2544Profile.mepType.t = 2
GetRfc2544Profile_Output.Rfc2544Profile.mepType.u.vlanDownMep = ''
GetRfc2544Profile_Output.Rfc2544Profile.vlanId = 999
GetRfc2544Profile_Output.Rfc2544Profile.pcp = 0
GetRfc2544Profile_Output.Rfc2544Profile.dei = 0
GetRfc2544Profile_Output.Rfc2544Profile.dMac = '00-00-00-00-00-01'

 GetRfc2544Profile Commit Success!!!
```

```
UCS(RFC2544PortType)# exit
```

Setting RFC 2544 Reporting Parameters for VLAN on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.

Step 3	**setReportParams rfc2544Reports reportAction** {**delete** *reportName* | **save** {*reportName* | *tftpPath*} | **start** {*reportName* | *profileName* | *description*} | **stop** *reportName* | **rename** {*oldName* | *newName*}}
	
	Example:
	`UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start profileName vlan-profile`
	`UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start reportName vlan-profile`
	`UCS(RFC2544PortType)# setReportParams`
		Sets RFC 2544 reporting parameters.
		• **reportAction**—Specifies action to be performed on the report.
		◦ **delete**—Deletes the existing report.
		◦ *reportName*—Specifies name of the report.
		◦ **save**—Saves the existing report.
		◦ *reportName*—Specifies the name of existing report.

	Command or Action	Purpose
	`rfc2544Reports reportAction start description vlan-profile`	◦ *tftpPath*—Specifies TFTP server URL tftp://*server[:port]/path-to-file*. ◦ **start**—Starts profile execution ◦ *reportName*—Specifies unique name of the resulting report. ◦ *profileName*—Specifies name of the profile to execute. ◦ *description*—(Optional) Provides a description of the report. **Note** We recommend that you add a description for the report. ◦ **stop**—Stops the report. ◦ *reportName*—Specifies name of the report to be stopped. ◦ **rename**—Renames the existing report. ◦ *oldName*—Specifies name of the old profile. ◦ *newName*—Specifies name of the new profile.
Step 4	**setReportParams review** **Example:** `UCS(RFC2544PortType)# setReportParams review`	Displays the setReportParams configuration.
Step 5	**setReportParams commit** **Example:** `UCS(RFC2544PortType)# setReportParams commit`	Sends the setReportParams configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to set the RFC 2544 reporting parameters for VLAN on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start profileName
vlan-profile
UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start reportName
vlan-profile
```

```
UCS(RFC2544PortType)# setReportParams rfc2544Reports reportAction start description
vlan-profile
UCS(RFC2544PortType)# setReportParams review
UCS(RFC2544PortType)# setReportParams commit
UCS(RFC2544PortType)# exit
```

Displaying RFC 2544 Report for VLAN on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.
Step 3	**showRfc2544 showRequest show** {**profiles** | **report**} **Example:** `UCS(RFC2544PortType)# showRfc2544 showRequest show profiles`	Displays RFC 2544 profile. • **show**—Displays profile or report. ◦ **profiles**—Displays profile information. ◦ **report**—Displays report information.
Step 4	**showRfc2544 review** **Example:** `UCS(RFC2544PortType)# showRfc2544 review`	Displays the showRfc2544 configuration.
Step 5	**showRfc2544 commit** **Example:** `UCS(RFC2544PortType)# showRfc2544 commit`	Sends the setRfc2544Profile configuration to the UCS controller.
Step 6	**showRfc2544 showRequest show** {**profiles** | **report**} **Example:** `UCS(RFC2544PortType)# showRfc2544 showRequest show report`	Displays RFC 2544 profile. • **show**—Displays profile or report. ◦ **profiles**—Displays profile information. ◦ **report**—Displays report information.
Step 7	**showRfc2544 review** **Example:** `UCS(RFC2544PortType)# showRfc2544 review`	Displays the showRfc2544 configuration.

	Command or Action	Purpose
Step 8	**showRfc2544 commit** **Example:** `UCS(RFC2544PortType)# showRfc2544 commit`	Sends the setRfc2544Profile configuration to the UCS controller.
Step 9	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to display RFC 2544 report for VLAN on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# showRfc2544 showRequest show report
UCS(RFC2544PortType)# showRfc2544 review
UCS(RFC2544PortType)# showRfc2544 commit
```

```
ShowRfc2544_Output.showResponse.t = 2
ShowRfc2544_Output.showResponse.u.report[0].reportName = 'Report1'
ShowRfc2544_Output.showResponse.u.report[0].created =
'1970-01-04T07:29:25+00:00'
ShowRfc2544_Output.showResponse.u.report[0].status = 'Succeeded'
ShowRfc2544_Output.showResponse.u.report[1].reportName = 'Rep15'
ShowRfc2544_Output.showResponse.u.report[1].created =
'1970-01-02T01:57:34+00:00'
ShowRfc2544_Output.showResponse.u.report[1].status = 'Failed'
ShowRfc2544_Output.showResponse.u.report[2].reportName = 'Rep16'
ShowRfc2544_Output.showResponse.u.report[2].created =
'1970-01-02T02:08:12+00:00'
ShowRfc2544_Output.showResponse.u.report[2].status = 'Succeeded'
ShowRfc2544_Output.showResponse.u.report[3].reportName = 'profile1'
ShowRfc2544_Output.showResponse.u.report[3].created =
'1970-01-02T03:48:16+00:00'
ShowRfc2544_Output.showResponse.u.report[3].status = 'Failed'

 ShowRfc2544 Commit Success!!!
```

```
UCS(RFC2544PortType)# exit
```

Deleting RFC 2544 Profile on NID-1

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.
Step 3	**deleterfc2544 rfc2544DeleteConfig** {**profileName** *profileName* | **delete** {**btob** | **dMAC** | **description** | **dwellTime** | **frameLoss** | **frameSizes** | **ifc** | **latency** | **megLevel** | **rfc2544** | **throughput** | **vid**}} **Example:** `UCS(RFC2544PortType)# deleteRfc2544 rfc2544DeleteConfig profileName profile1`	Deletes RFC profile. • **profileName—**Specifies RFC 2544 profile name. • *profileName*—Name of the RFC 2544 profile . • **delete—**Deletes the specific attributes of the profile. • **btob—**Removes back-to-back test. • **dMAC—**Removes destination MAC. • **description—**Removes description. • **dwellTime—**Removes dwell time. • **frameLoss—**Removes frame loss test. • **frameSizes—**Removes frame sizes. • **ifc—**Removes IFC. • **latency—**Removes latency test. • **megLevel—**Removes MEG level. • **rfc2544—**Removes RFC 2544 profile. • **throughput—**Removes throughput test. • **vid—**Removes version ID (VID).
Step 4	**deleteRfc2544 review** **Example:** `UCS(RFC2544PortType)# deleteRfc2544 review`	Displays the deleteRfc2544 configuration.
Step 5	**deleteRfc2544 commit** **Example:** `UCS(RFC2544PortType)# deleteRfc2544 commit`	Sends the deleteRfc2544 configuration to the UCS controller.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to delete RFC 2544 profile on NID-1:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# deleteRfc2544 rfc2544DeleteConfig profileName profile1
UCS(RFC2544PortType)# deleteRfc2544 review
UCS(RFC2544PortType)# deleteRfc2544 commit
UCS(RFC2544PortType)# exit
```

Modifying RFC 2544 with Frameloss and Backtoback

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.

Step 3	**SetRfc2544TestToRun testParameters** {**profileName** *profileName* | **frameSizes** | **testToRun** {**throughput** | **latency** | **frameLoss** | **backToBack**} | **thoughputTParams** {**trialDuration** | **minRate** | **maxRate** | **accuracy** | **allowedFrameLoss**} | **latencyTParams** {**trialDuration** | **delayMessInterval** | **allowedFrameLoss**} | **frameLossTParams** {**trialDuration** | **minRate** | **maxRate** | **rateStep**} | **backToBackTParams** {**trialDuration** | **trialCount**}}
	
	Example:
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters profileName vlan-profile`
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters backToBackTParams trialCount`
		Modifies RFC 2544 with Frameloss and Backtoback.
		• **testParameters—**Specifies RFC 2544 test parameters.
		• **profileName—**Specifies RFC 2544 profile name.
		• *profileName*—Name of the RFC 2544 profile.
		• **frameSizes—**Specifies frame sizes separated by a comma, for example, 1024,128,1280,1518, 2000, 256, 512, 64, 9600.
		• **testToRun—**Test to be run.
		  ◦ **throughput—**Enables throughput test and optionally set its parameters.
		  ◦ **latency—**Enables latency test and optionally set its parameters.

	Command or Action	Purpose

	2
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters frameLossTParams minRate 100`
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters frameLossTParams maxRate 200`
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters frameLossTParams rateStep 10`
		◦ **frameLoss—**Enables frame-loss test and optionally set its parameters.
		◦ **backToBack—**Enables back-to-back test and optionally set its parameters.
		• **thoughputTParams—**Specifies throughput test parameters.
		◦ **trialDuration—**Sets the duration of one trial.
		◦ **minRate—**Sets the minimum rate.
		◦ **maxRate—**Sets the maximum rate.
		◦ **accuracy—**Sets the accuracy (stop criterion).
		◦ **allowedFrameLoss—**Sets the maximum allowed test protocol data unit (PDU) loss at which the test is considered successful.
		• **latencyTParams—**Specifies latency test parameters.
		◦ **trialDuration—**Sets the duration of one trial.
		◦ **delayMessInterval—**Specifies interval between sending delay measurement frames.
		◦ **allowedFrameLoss—**Sets the maximum allowed test PDU loss at which the test is considered successful.
		• **frameLossTParams—**Specifies frame loss test parameters.
		◦ **trialDuration—**Sets the duration of one trial.
		◦ **minRate—** Sets the minimum rate.
		◦ **maxRate—** Sets the maximum rate.
		◦ **rateStep—**Sets the step rate.
		• **backToBackTParams—**Specifies back to back test parameters.
		◦ **trialDuration—**Specifies the time (in milliseconds) to transmit a burst of Y.1731 test frames at line rate and frame size.
		◦ **trialCount—**Specifies the number of times to repeat the burst.

	Command or Action	Purpose
Step 4	**SetRfc2544TestToRun review** **Example:** `UCS(RFC2544PortType)# SetRfc2544TestToRun review`	Displays the SetRfc2544TestToRun configuration.
Step 5	**SetRfc2544TestToRun commit** **Example:** `UCS(RFC2544PortType)# SetRfc2544TestToRun commit`	Sends the SetRfc2544TestToRun configuration to the UCS controller.

Step 6	**SetRfc2544TestToRun testParameters** {**profileName** *profileName* | **frameSizes** | **testToRun** {**throughput** | **latency** | **frameLoss** | **backToBack**} | **thoughputTParams** {**trialDuration** | **minRate** | **maxRate** | **accuracy** | **allowedFrameLoss**} | **latencyTParams** {**trialDuration** | **delayMessInterval** | **allowedFrameLoss**} | **frameLossTParams** {**trialDuration** | **minRate** | **maxRate** | **rateStep**} | **backToBackTParams** {**trialDuration** | **trialCount**}}
	
	Example:
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun backToBack enable`
	
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun frameLoss enable`
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun latency disable`
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun throughput disable`
	
	`UCS(RFC2544PortType)# setRfc2544TestToRun testParameters profileName vlan-profile`
		Modifies RFC 2544 with Frameloss and Backtoback.
		• **testParameters—**Specifies RFC 2544 test parameters.
		• **profileName—**Specifies RFC 2544 profile name.
		• *profileName*—Name of the RFC 2544 profile.
		• **frameSizes—**Specifies frame sizes separated by a comma, for example, 1024,128,1280,1518, 2000, 256, 512, 64, 9600.
		• **testToRun—**Test to be run.
		◦ **throughput—**Enables throughput test and optionally set its parameters.
		◦ **latency—**Enables latency test and optionally set its parameters.
		◦ **frameLoss—**Enables frame-loss test and optionally set its parameters.
		◦ **backToBack—**Enables back-to-back test and optionally set its parameters.
		• **thoughputTParams—**Specifies throughput test parameters.
		◦ **trialDuration—**Sets the duration of one trial.
		◦ **minRate—**Sets the minimum rate.
		◦ **maxRate—**Sets the maximum rate.
		◦ **accuracy—**Sets the accuracy (stop criterion).
		◦ **allowedFrameLoss—**Sets the maximum allowed test protocol data unit (PDU) loss at which the test is considered successful.
		• **latencyTParams—**Specifies latency test parameters.
		◦ **trialDuration—**Sets the duration of one trial.

	Command or Action	Purpose
		◦ **delayMessInterval—**Specifies interval between sending delay measurement frames. ◦ **allowedFrameLoss—**Sets the maximum allowed test PDU loss at which the test is considered successful. • **frameLossTParams—**Specifies frame loss test parameters. ◦ **trialDuration—**Sets the duration of one trial. ◦ **minRate—** Sets the minimum rate. ◦ **maxRate—** Sets the maximum rate. ◦ **rateStep—**Sets the step rate. • **backToBackTParams—**Specifies back to back test parameters. ◦ **trialDuration—**Specifies the time (in milliseconds) to transmit a burst of Y.1731 test frames at line rate and frame size. ◦ **trialCount—**Specifies the number of times to repeat the burst.
Step 7	**SetRfc2544TestToRun review** **Example:** `UCS(RFC2544PortType)# SetRfc2544TestToRun review`	Displays the SetRfc2544TestToRun configuration.
Step 8	**SetRfc2544TestToRun commit** **Example:** `UCS(RFC2544PortType)# SetRfc2544TestToRun commit`	Sends the SetRfc2544TestToRun configuration to the UCS controller.
Step 9	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortType mode.

Configuration Example

The example shows how to modify and enable RFC 2544 with Frameloss and Backtoback:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters profileName vlan-profile
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters backToBackTParams trialCount 2
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters frameLossTParams minRate 100
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters frameLossTParams maxRate 200
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters frameLossTParams rateStep 10
UCS(RFC2544PortType)# setRfc2544TestToRun review
```

```
UCS(RFC2544PortType)# setRfc2544TestToRun commit

UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun backToBack enable
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun frameLoss enable
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun latency disable
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters testToRun throughput disable
UCS(RFC2544PortType)# setRfc2544TestToRun testParameters profileName vlan-profile
UCS(RFC2544PortType)# setRfc2544TestToRun review
UCS(RFC2544PortType)# setRfc2544TestToRun commit

UCS(RFC2544PortType)# exit
```

Getting RFC 2544 Profile after Modifying Frameloss and Backtoback

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**RFC2544PortType** **Example:** `UCS# RFC2544PortType`	Enters the RFC2544PortType mode.
Step 3	**getRfc2544TestToRun rfc2544Request profileName** *profileName* **Example:** `UCS(RFC2544PortType)# getRfc2544TestToRun rfc2544Request profileName vlan-profile`	Gets the RFC 2544 profile. • **rfc2544Request—**Specifies RFC2544 request parameter. • **profileName—**Specifies name of the profile.
Step 4	**getRfc2544TestToRun review** **Example:** `UCS(RFC2544PortType)# getRfc2544TestToRun review`	Displays the getRfc2544Profile configuration.
Step 5	**getRfc2544TestToRun commit** **Example:** `UCS(RFC2544PortType)# getRfc2544TestToRun commit`	Sends the getRfc2544Profile configuration to the UCS controller.
Step 6	**exit** **Example:** `UCS(RFC2544PortType)# exit`	Exits the RFC2544PortTypemode.

Configuration Example

The example shows how to get RFC 2544 Profile after modifying frameloss and backtoback:

```
UCS# RFC2544PortType
UCS(RFC2544PortType)# getRfc2544TestToRun rfc2544Request profileName vlan-profile
UCS(RFC2544PortType)# getRfc2544TestToRun review
UCS(RFC2544PortType)# getRfc2544TestToRun commit
```

```
GetRfc2544TestToRun_Output.testParameters.profileName = 'vlan-profile'
GetRfc2544TestToRun_Output.testParameters.frameSizes =
'64-128-256-512-1024-1280-1518-2000'
GetRfc2544TestToRun_Output.testParameters.testToRun.throughput = false
GetRfc2544TestToRun_Output.testParameters.testToRun.latency = false
GetRfc2544TestToRun_Output.testParameters.testToRun.frameLoss = true
GetRfc2544TestToRun_Output.testParameters.testToRun.backToBack = true
GetRfc2544TestToRun_Output.testParameters.thoughputTParams.trialDuration
 = 60
GetRfc2544TestToRun_Output.testParameters.thoughputTParams.minRate = 800
GetRfc2544TestToRun_Output.testParameters.thoughputTParams.maxRate = 1000
GetRfc2544TestToRun_Output.testParameters.thoughputTParams.accuracy = 2
GetRfc2544TestToRun_Output.testParameters.thoughputTParams.allowedFrameLoss
 = 0
GetRfc2544TestToRun_Output.testParameters.latencyTParams.trialDuration =
 120
GetRfc2544TestToRun_Output.testParameters.latencyTParams.delayMessInterval
 = 10
GetRfc2544TestToRun_Output.testParameters.latencyTParams.allowedFrameLoss
 = 0
GetRfc2544TestToRun_Output.testParameters.frameLossTParams.trialDuration
 = 60
GetRfc2544TestToRun_Output.testParameters.frameLossTParams.minRate = 800
GetRfc2544TestToRun_Output.testParameters.frameLossTParams.maxRate = 1000
GetRfc2544TestToRun_Output.testParameters.frameLossTParams.rateStep = 5
GetRfc2544TestToRun_Output.testParameters.backToBackTParams.trialDuration
 = 2000
GetRfc2544TestToRun_Output.testParameters.backToBackTParams.trialCount =
 50

 GetRfc2544TestToRun Commit Success!!!
```

```
UCS(RFC2544PortType)# exit
```

Verifying RFC 2544

Use the following commands to verify the RFC 2544 status on the UCS controller.

- **showRfc2544 com**

  This command displays the RFC 2544 report. The following is a sample output from the command:

```
UCS(SPAN)# showRfc2544 com
UCS(SPAN)# showRfc2544 com review
```

```
Commands in queue:
     showRfc2544 com
```

```
UCS(SPAN)# showSpanConfig commit
```

```
ShowRfc2544_Output.showResponse.t = 2
ShowRfc2544_Output.showResponse.u.report[0].reportName = 'Jul3'
ShowRfc2544_Output.showResponse.u.report[0].created =
'1970-01-04T01:02:24+00:00'
ShowRfc2544_Output.showResponse.u.report[0].status = 'Failed'
ShowRfc2544_Output.showResponse.u.report[1].reportName = 'July3'
ShowRfc2544_Output.showResponse.u.report[1].created =
'1970-01-04T01:15:37+00:00'
ShowRfc2544_Output.showResponse.u.report[1].status = 'Failed'
ShowRfc2544_Output.showResponse.u.report[2].reportName = 'repjuly3'
ShowRfc2544_Output.showResponse.u.report[2].created =
'1970-01-04T01:52:07+00:00'
ShowRfc2544_Output.showResponse.u.report[2].status = 'Succeeded'
ShowRfc2544_Output.showResponse.u.report[3].reportName = 'Report1'
ShowRfc2544_Output.showResponse.u.report[3].created =
'1970-01-04T07:29:25+00:00'
ShowRfc2544_Output.showResponse.u.report[3].status = 'Succeeded'
ShowRfc2544_Output.showResponse.u.report[4].reportName = 'rep-vlan'
ShowRfc2544_Output.showResponse.u.report[4].created =
'1970-01-04T21:01:59+00:00'
ShowRfc2544_Output.showResponse.u.report[4].status = 'Failed'
ShowRfc2544_Output.showResponse.u.report[5].reportName = 'Report20'
ShowRfc2544_Output.showResponse.u.report[5].created =
'1970-01-01T08:15:17+00:00'
ShowRfc2544_Output.showResponse.u.report[5].status = 'Failed'
ShowRfc2544_Output.showResponse.u.report[6].reportName = 'Rep22'
ShowRfc2544_Output.showResponse.u.report[6].created =
'1970-01-01T09:36:14+00:00'
ShowRfc2544_Output.showResponse.u.report[6].status = 'Failed'
ShowRfc2544_Output.showResponse.u.report[7].reportName = 'profile2'
ShowRfc2544_Output.showResponse.u.report[7].created =
'1970-01-02T00:55:43+00:00'
ShowRfc2544_Output.showResponse.u.report[7].status = 'Failed'

ShowRfc2544 Commit Success!!!
```

Additional References

Related Documents

Related Topic	Document Title
Cisco ME 3800x and ME 3600x Switches Software Configuration Guide, Cisco IOS Release 15.4(1)S	http://www.cisco.com/c/en/us/td/docs/switches/metro/me3600x_3800x/software/release/15-4_1_S/configuration/guide/3800x3600xscg.html

MIBs

MIB	MIBs Link
MIBs Supporting Cisco IOS	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

CHAPTER 29

Configuring sFlow

This document describes the sampled flow (sFlow) feature and configuration steps to implement sFlow.

Prerequisites for Configuring sFlow

- You must enable sFlow on an interface on ME 1200 NID.
- NID must have an IP address.

Restrictions for Configuring sFlow

- ME 1200 NID does not support configuring more than one sFlow instance with maximum sample rate on the specified data source, either in the ingress or egress direction.

Information About sFlow

Using sFlow, a standards-based protocol mechanism, allows you to monitor Layer 2 traffic in data networks that contain switches and routers. It consists of :

- **sFlow Agent** (embedded on ME 1200 NID)—The sFlow Agent uses sampling technology to capture traffic statistics from the monitored device and then forwards the sampled data to a central sFlow Collector for analysis. Packet sampling is done using one or more sFlow instances, each configured with a sampling rate.

  - sFlow Instances—There may be one or more sFlow Instances associated with a single data source. Each sFlow instance operates independently of other sFlow instances. For example, Packet Flow Sampling instances have their own sampling rates and Counter Sampling instances have their own sampling intervals.

- **sFlow Collector—**The sFlow Collector is a software application that can receive sFlow datagrams and present a view of traffic and other network parameters which are output as type, length, and value (TLV) in the datagrams. The sFlow collectors can also read and configure sFlow-managed objects. Both counter and packet flow statistics are collected and sent as sFlow Datagrams (defined by maximum datagram size of 200-1468) to a sFlow Collector.
  - sFlow Datagram—The sFlow Datagram format specifies a standard format for the sFlow Agent to send sampled data to a remote sFlow Collector. The sFlow Datagram version 5 is supported.

    The format of the sFlow Datagram is specified using the External Data Representation (XDR) standard. This makes it simpler for the sFlow Agent to encode and the sFlow Collector to decode.

    Samples are sent as User Datagram Protocol (UDP) packets to the host and port specified in the SFLOW MIB or CLI. The assigned port for sFlow (and the default specified in the SFLOW MIB) is port 6343. All sFlow Agents and applications by default must use UDP port 6343.

By default, sFlow is disabled on ME 1200 NID. You can enable sFlow on a specific interface or port.

How to Provision sFlow

Enabling sFlow Globally

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**sflow** **Example:** `UCS# sflow`	Enters the sFlow mode.

Step 3	Flow_global {**agent-ip** {**ipv4** | **ipv6**} | **collector-ip** {**ipv4** | **ipv6**} | **collector-port** | **datagram-maxsize** | **rx-timeout**}
	
	Example:
	`UCS(SFlow)# setsFlowGlobalConfig sFlow_global agent-ip ipv4 7.25.16.63`
	`UCS(SFlow)# setsFlowGlobalConfig sFlow_global collector-ip ipv4 7.25.16.253`
		Enters the sFlow global configuration mode.
		• **agent-ip—**Specifies Agent IP address.
		◦ **ipv4—**Specifies IPv4 address.
		◦ **ipv6—**Specifies IPv6 address.
		• **collector-ip—**Specifies collector IP address.

	Command or Action	Purpose
	`UCS(SFlow)# setsFlowGlobalConfig sFlow_global collector-port 2033` `UCS(SFlow)# setsFlowGlobalConfig sFlow_global datagram-maxsize 512` `UCS(SFlow)#setsFlowGlobalConfig sFlow_global rx-timeout 50000`	◦ **ipv4—**Specifies IPv4 address. ◦ **ipv6—**Specifies IPv6 address. • **collector-port—**Specifies collector UDP port. The valid range is from 1 to 65535. • **datagram-maxsize—**Specifies maximum datagram size. The valid range is from 200 to 1468. • **rx-timeout—**Specifies the receive timeout in seconds. The valid range is from 0 to 2147483647. The switch decrements the timeout once every second, and samples are received as long as it is non-zero. Once it reaches zero, receiver and all its configurations are reset to defaults.
Step 4	**setsFlowGlobalConfig review** **Example:** `UCS(SFlow)# setsFlowGlobalConfig review`	(Optional) Displays the configuration.
Step 5	**setsFlowGlobalConfig commit** **Example:** `UCS(SFlow)# setsFlowGlobalConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(SFlow)# exit`	Exits the SFlow mode.

Configuration Example

- The example shows how to enable sFlow globally:

```
UCS# sflow
UCS(SFlow)# setsFlowGlobalConfig sFlow_global agent-ip ipv4 7.25.16.63
UCS(SFlow)# setsFlowGlobalConfig sFlow_global collector-ip ipv4 7.25.16.253
UCS(SFlow)# setsFlowGlobalConfig sFlow_global collector-port 2033
UCS(SFlow)# setsFlowGlobalConfig sFlow_global datagram-maxsize 512
UCS(SFlow)# setsFlowGlobalConfig sFlow_global rx-timeout 50000
UCS(SFlow)# setsFlowGlobalConfig review
```

```
Commands in queue:
          setsFlowGlobalConfig sFlow_global agent-ip ipv4 7.25.16.63
        setsFlowGlobalConfig sFlow_global collector-ip ipv4 7.25.16.253

         setsFlowGlobalConfig sFlow_global collector-port 6343
         setsFlowGlobalConfig sFlow_global datagram-maxsize 512
         setsFlowGlobalConfig sFlow_global rx-timeout 50000
```

```
UCS(SFlow)# setsFlowGlobalConfig commit
```

```
SetsFlowGlobalConfig Commit Success!!!
```

```
UCS(SFlow)# exit
```

Enabling sFlow on a Port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**sflow** **Example:** `UCS# sflow`	Enters the sFlow mode.

Step 3	sflow_port {**interface-id** | **enable** | **flow-sampler** {**enable-defaults** | **sampling-rate** | **sampling-maxsize**} | **counter-poller** {**enable** | **interval**}}
	
	Example:
	`UCS(SFlow)# setsFlowPortConfig sFlow_port interface-id 3`
	`UCS(SFlow)# setsFlowPortConfig sFlow_port enable enable`
	`UCS(SFlow)# setsFlowPortConfig sFlow_port counter-poller enable enable`
	`UCS(SFlow)# setsFlowPortConfig sFlow_port counter-poller interval 10`
	`UCS(SFlow)# setsFlowPortConfig sFlow_port flow-sampler enable-defaults enable`
		Enters the sFlow port specific configuration mode.
		• **interface-id—**Specifies physical port.
		• **enable—**Enables or disables sFlow on this port.
		• **flow-sampler—**Specifies sFlow flow sampler configuration.
		◦ **enable-defaults—**Enables the flow sampler default values.
		Note To configure **sampling-rate** and **sampling-maxsize** as per your requirement, you must set this option to disable.
		◦ **sampling-rate—**Specifies the statistical sampling rate. The valid range is from 1 to 4294967295.
		◦ **sampling-maxsize—**Specifies maximum number of bytes to transmit per flow sample. The valid range is from 14 to 200.
		• **counter-poller—**Specifies Interface counter poller configuration.
		◦ **enable—**Enables counter poller.
		◦ **interval—**Specifies counter poll interval. The valid range is from 1 to 3600 seconds.

	Command or Action	Purpose
Step 4	**setsFlowPortConfig review** **Example:** `UCS(SFlow)# setsFlowPortConfig review`	(Optional) Displays the configuration.
Step 5	**setsFlowPortConfig commit** **Example:** `UCS(SFlow)# setsFlowPortConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(SFlow)# exit`	Exits the SFlow mode.

Configuration Example

> **Note** sFlow configuration does not persist on the NID. Running **show running-config.xml** command does not display the sFlow configuration globally or per-port. This is working as designed.

- The example shows how to enable sFlow on a port with default values enabled:

```
UCS# sflow
UCS(SFlow)# setsFlowPortConfig sFlow_port interface-id 3
UCS(SFlow)# setsFlowPortConfig sFlow_port enable enable
UCS(SFlow)# setsFlowPortConfig sFlow_port flow-sampler enable-defaults enable
UCS(SFlow)# setsFlowPortConfig review
```

```
Commands in queue:
        setsFlowPortConfig sFlow_port interface-id 3
        setsFlowPortConfig sFlow_port enable enable
        setsFlowPortConfig sFlow_port flow-sampler enable-defaults
enable
```

```
UCS(SFlow)# setsFlowPortConfig commit
```

```
 SetsFlowPortConfig Commit Success!!!
```

```
UCS(SFlow)# exit
```

- The example shows how to enable sFlow on a port without any default values set:

```
UCS# sflow
UCS(Flow)# setsFlowPortConfig sFlow_port interface-id 1
UCS(SFlow)# setsFlowPortConfig sFlow_port enable enable
UCS(SFlow)# setsFlowPortConfig sFlow_port flow-sampler enable-defaults disable
UCS(SFlow)# setsFlowPortConfig review
```

```
Commands in queue:
        setsFlowPortConfig sFlow_port interface-id 1
        setsFlowPortConfig sFlow_port enable enable
```

```
        setsFlowPortConfig sFlow_port flow-sampler enable-defaults
disable
```

```
UCS(SFlow)# setsFlowPortConfig commit
```

```
 SetsFlowPortConfig Commit Success!!!
```

```
UCS(SFlow)# exit
```

- The example shows how enable sFlow on a port with user-configured parameters:

```
UCS# sflow
UCS(SFlow)# setsFlowPortConfig sFlow_port interface-id 1
UCS(SFlow)# setsFlowPortConfig sFlow_port enable enable
UCS(SFlow)# setsFlowPortConfig sFlow_port flow-sampler enable-defaults disable
UCS(SFlow)# setsFlowPortConfig sFlow_port flow-sampler sampling-maxsize 512
UCS(SFlow)# setsFlowPortConfig sFlow_port flow-sampler sampling-rate 200
UCS(SFlow)# setsFlowPortConfig sFlow_port counter-poller enable enable
UCS(SFlow)# setsFlowPortConfig sFlow_port counter-poller interval 30
UCS(SFlow)# setsFlowPortConfig review
```

```
Commands in queue:
        setsFlowPortConfig sFlow_port interface-id 1
        setsFlowPortConfig sFlow_port enable enable
        setsFlowPortConfig sFlow_port flow-sampler enable-defaults
disable
        setsFlowPortConfig sFlow_port flow-sampler sampling-maxsize
512
        setsFlowPortConfig sFlow_port flow-sampler sampling-rate 200

        setsFlowPortConfig sFlow_port counter-poller enable enable
        setsFlowPortConfig sFlow_port counter-poller interval 30
```

```
UCS(SFlow)# setsFlowPortConfig commit
```

```
 SetsFlowPortConfig Commit Success!!!
```

```
UCS(SFlow)# exit
```

Getting Current Global sFlow Values

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**sflow** **Example:** `UCS# sflow`	Enters the sFlow mode.

	Command or Action	Purpose
Step 3	sFlow_global_req **Example:** `UCS(SFlow)# getsFlowGlobalConfig sFlow_global_req`	Enters the sFlow global configuration mode.
Step 4	**getsFlowGlobalConfig review** **Example:** `UCS(SFlow)# getsFlowGlobalConfig review`	(Optional) Displays the configuration.
Step 5	**getsFlowGlobalConfig commit** **Example:** `UCS(SFlow)# getsFlowGlobalConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(SFlow)# exit`	Exits the SFlow mode.

Configuration Example

- The example shows how to get current global sFlow values:

```
UCS# sflow
UCS(SFlow)# getsFlowGlobalConfig sFlow_global_req
UCS(SFlow)# getsFlowGlobalConfig review
```

```
Commands in queue:
                getsFlowGlobalConfig sFlow_global_req
```

```
UCS(SFlow)# getsFlowGlobalConfig commit
```

```
GetsFlowGlobalConfig_Output.sFlow_global.agent_ip.t = 1
GetsFlowGlobalConfig_Output.sFlow_global.agent_ip.u.ipv4 = '0.0.0.0'
GetsFlowGlobalConfig_Output.sFlow_global.collector_ip.t = 1
GetsFlowGlobalConfig_Output.sFlow_global.collector_ip.u.ipv4 =
'0.0.0.0'
GetsFlowGlobalConfig_Output.sFlow_global.collector_port = 65535
GetsFlowGlobalConfig_Output.sFlow_global.datagram-maxsize = 1468
GetsFlowGlobalConfig_Output.sFlow_global.rx-timeout = 50000

 GetsFlowGlobalConfig Commit Success!!!
```

```
UCS(SFlow)# exit
```

The following is a sample output on the NID.

```
Decoding of Request message was successful urn:#getsFlowConfig
Decoded record:
GetsFlowGlobalConfig_Input.sFlow_global_req = '0'
Encoding of Response message was successful
Encoded record:
```

```
GetsFlowGlobalConfig_Output.sFlow_global.agent_ip.t = 1
GetsFlowGlobalConfig_Output.sFlow_global.agent_ip.u.ipv4 = '0.0.0.0'
GetsFlowGlobalConfig_Output.sFlow_global.collector_ip.t = 1
GetsFlowGlobalConfig_Output.sFlow_global.collector_ip.u.ipv4 =
'0.0.0.0'
GetsFlowGlobalConfig_Output.sFlow_global.collector_port = 65535
GetsFlowGlobalConfig_Output.sFlow_global.datagram-maxsize = 1468
GetsFlowGlobalConfig_Output.sFlow_global.rx-timeout = 50000
GetsFlowGlobalConfig_Output.xmlns:ns0 =
"http://new.webservice.namespace"
GetsFlowGlobalConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetsFlowGlobalConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetsFlowGlobalConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetsFlowGlobalConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetsFlowGlobalConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

Getting Current Port Specific sFlow Values

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**sflow** **Example:** `UCS# sflow`	Enters the sFlow mode.
Step 3	**sflowPortConfigReq** *port id* **Example:** `UCS(SFlow)# getsFlowPortConfig sflowPortConfigReq 2`	Enters the sFlow port-specific configuration mode.
Step 4	**getsFlowGlobalConfig review** **Example:** `UCS(SFlow)# getsFlowGlobalConfig review`	(Optional) Displays the configuration.
Step 5	**getsFlowGlobalConfig commit** **Example:** `UCS(SFlow)# getsFlowGlobalConfig commit`	Sends the configuration to NID.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(SFlow)# exit`	Exits the SFlow mode.

Configuration Example

- The example shows how to get current port-specific sFlow values:

```
UCS# sflow
UCS(SFlow)# getsFlowPortConfig sflowPortConfigReq 2
UCS(SFlow)# getsFlowGlobalConfig review
```

```
Commands in queue:
        getsFlowPortConfig sFlowPortConfigReq 2
```

```
UCS(SFlow)# getsFlowGlobalConfig commit
```

```
GetsFlowPortConfig_Output.sFlow_port.interface_id = 2
GetsFlowPortConfig_Output.sFlow_port.enable = false
GetsFlowPortConfig_Output.sFlow_port.flow_sampler.enable_defaults =
true
GetsFlowPortConfig_Output.sFlow_port.flow_sampler.sampling_rate = 4096
GetsFlowPortConfig_Output.sFlow_port.flow_sampler.sampling-maxsize =
 128
GetsFlowPortConfig_Output.sFlow_port.counter_poller.enable = false
GetsFlowPortConfig_Output.sFlow_port.counter_poller.interval = 60

 GetsFlowPortConfig Commit Success!!!
```

```
UCS(SFlow)# exit
```

The following is a sample output on the NID.

```
 GetsFlowPortConfig_Input.sFlowPortConfigReq = 2
Encoding of Response message was successful
Encoded record:
GetsFlowPortConfig_Output.sFlow_port.interface_id = 2
GetsFlowPortConfig_Output.sFlow_port.enable = false
GetsFlowPortConfig_Output.sFlow_port.flow_sampler.enable_defaults =
true
GetsFlowPortConfig_Output.sFlow_port.flow_sampler.sampling_rate = 4096
GetsFlowPortConfig_Output.sFlow_port.flow_sampler.sampling-maxsize =
 128
GetsFlowPortConfig_Output.sFlow_port.counter_poller.enable = false
GetsFlowPortConfig_Output.sFlow_port.counter_poller.interval = 60
GetsFlowPortConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetsFlowPortConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetsFlowPortConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetsFlowPortConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
```

```
GetsFlowPortConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetsFlowPortConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

Clearing sFlow Statistics

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**sflow** **Example:** `UCS# sflow`	Enters the sFlow mode.
Step 3	clear_sflow_stats **Example:** `UCS(SFlow)# clearsFlowStatistics clear_sflow_stats`	Clears sFlow statistics.
Step 4	**clearsFlowStatistics review** **Example:** `UCS(SFlow)# clearsFlowStatistics review`	(Optional) Displays the configuration.
Step 5	**clearsFlowStatistics commit** **Example:** `UCS(SFlow)# clearsFlowStatistics commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(SFlow)# exit`	Exits the SFlow mode.

Configuration Example

- The example shows how to clear sFlow statistics:

```
UCS# sflow
UCS((SFlow)# clearsFlowStatistics clear_sflow_stats
UCS((SFlow)# clearsFlowStatistics review
```

```
Commands in queue:
              clearsFlowStatistics clear_sflow_stats
```

```
UCS(SFlow)# clearsFlowStatistics commit
```

```
ClearsFlowStatistics Commit Success!!!
```

```
UCS(SFlow)# exit
```

Verifying sFlow

Use the following commands to verify the sFlow status on the UCS controller.

- showsFlowStatistics sFlow_stats_req

  This command displays the sFlow statistics on the NID. The following is a sample output from the command:

```
UCS(SFlow)# showsFlowStatistics sFlow_stats_req
```

```
UCS(SFlow)#  showsFlowStatistics review
```

```
Commands in queue:
                showsFlowStatistics sFlow_stats_req
```

```
UCS(SFlow)#  showsFlowStatistics commit
```

```
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[0].interface_id
 = 1
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[0].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[0].counter_samples
 = 42
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[1].interface_id
 = 2
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[1].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[1].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[2].interface_id
 = 3
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[2].tx_flow_samples
 = 1
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[2].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[3].interface_id
 = 4
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[3].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[3].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[4].interface_id
 = 5
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[4].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[4].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[5].interface_id
 = 6
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[5].tx_flow_samples
 = 0
```

```
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[5].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].tx_successes
 = 0
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].tx_errors
 = 43
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].flow_samples
 = 1
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].counter_samples
 = 42

 ShowsFlowStatistics Commit Success!!!
```

The following is a sample output on the NID.

```
 Decoding of Request message was successful urn:#showsFlowStatistics
Decoded record:
ShowsFlowStatistics_Input.sFlow_stats_req = '0'
Encoding of Response message was successful
Encoded record:
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[0].interface_id
 = 1
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[0].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[0].counter_samples
 = 42
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[1].interface_id
 = 2
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[1].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[1].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[2].interface_id
 = 3
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[2].tx_flow_samples
 = 1
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[2].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[3].interface_id
 = 4
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[3].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[3].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[4].interface_id
 = 5
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[4].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[4].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[5].interface_id
 = 6
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[5].tx_flow_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.samplers.interface_[5].counter_samples
 = 0
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].tx_successes
 = 0
```

```
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].tx_errors
 = 43
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].flow_samples
 = 1
ShowsFlowStatistics_Output.sFlow_stats.receiver.statistics[0].counter_samples
 = 42
ShowsFlowStatistics_Output.xmlns:ns0 =
"http://new.webservice.namespace"
ShowsFlowStatistics_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
ShowsFlowStatistics_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
ShowsFlowStatistics_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
ShowsFlowStatistics_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
ShowsFlowStatistics_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

Additional References

Related Documents

Related Topic	Document Title
Cisco ME 3800x and ME 3600x Switches Software Configuration Guide, Cisco IOS Release 15.4(1)S	http://www.cisco.com/c/en/us/td/docs/switches/metro/me3600x_3800x/software/release/15-4_1_S/configuration/guide/3800x3600xscg.html

MIBs

MIB	MIBs Link
MIBs Supporting Cisco IOS	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

CHAPTER 30

Configuring UDLD

This document describes the Unidirectional Link Detection (UDLD) feature and configuration steps to implement UDLD.

Prerequisites for Configuring UDLD

- To identify and disable the unidirectional links, devices at both ends must support UDLD.
- UDLD Hello packet timers must be same for both the devices.
- NID must have an IP address.

Restrictions for Configuring UDLD

- Access Control Lists (ACLs) cannot be used to block the UDLD traffic.
- UDLD on ME 1200 NID cannot interoperate with other devices. UDLD can be enabled only between ME 1200 NIDs.

Information About UDLD

UDLD is a Layer 2 protocol that enables devices connected through Ethernet cables to monitor the physical configuration and detect presence of a unidirectional link. A unidirectional link occurs when traffic sent by a local device is received by its neighbor but traffic coming from the neighboring device is not received by the local device. When a unidirectional link is detected, the affected port is disabled and user is alerted. This can help prevent spanning tree topology loops.

UDLD supports two operation modes:

- **Normal**—In this mode, UDLD detects a unidirectional link due to misconnected fibers on a fiber-optic link that is not detected by Layer 1 mechanisms.

  Note If port connections are correct and traffic is one way, UDLD does not detect the unidirectional link. In this case, no action is taken and link is considered undetermined.

- **Aggressive**—In this mode, unidirectional link due to one-way traffic on fiber-optic and twisted pair links, and misconnected ports on fiber-optic links can be detected. Specifically, if one end of the link cannot send or receive traffic, or one of the ports is down and the other is up, the unidirectional link can be detected. Using the loss of hello packets as indication to detect bi-directional link that cannot be re-established, UDLD disables the affected port.

In addition, UDLD can detect the identities of neighbors by caching the information contained in UDLD hello packet.

By default, UDLD is disabled on ME 1200 NID. The normal or aggressive mode can be

- enabled globally on all ports or
- enabled or modified on the individual ports

How to Provision UDLD

Enabling UDLD Mode Globally

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**UDLDPortType** **Example:** `UCS# UDLDPortType`	Enters the UDLD mode.
Step 3	**udldGlobalConfig** {**mode** {**normal** | **aggressive**} | **message_interval**}	Enters UDLD global configuration mode. Sub-command options:

	Command or Action	Purpose

	Example:
	`UCS(UDLDPortType)# setGlobalUDLDConfig udldGlobalConfig mode normal enable`
	`UCS(UDLDPortType)# setGlobalUDLDConfig udldGlobalConfig message_interval 10`
		• **mode—**Specifies UDLD configuration mode.
		◦ **normal—**Enables UDLD in normal mode on all fiber-optic ports.
		◦ **aggressive—**Enables UDLD in aggressive mode on all fiber-optic ports.
		• **message-interval—**Specifies time interval between UDLD probe messages on ports (7-90 seconds).

Step 4	**setGlobalUDLDConfig review** **Example:** `UCS(UDLDPortType)# setGlobalUDLDConfig review`	(Optional) Displays the configuration.
Step 5	**setGlobalUDLDConfig commit** **Example:** `UCS(UDLDPortType)# setGlobalUDLDConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(UDLDPortType)# exit`	Exits the UDLDPortType mode.

Configuration Example

- The example shows how to enable UDLD globally in normal mode:

```
UCS# UDLDPortType
UCS(UDLDPortType)# setGlobalUDLDConfig udldGlobalConfig mode normal enable
UCS(UDLDPortType)# setGlobalUDLDConfig udldGlobalConfig message-interval 10
UCS(UDLDPortType)# setGlobalUDLDConfig review
```

```
Commands in queue:
        setGlobalUDLDConfig udldGlobalConfig mode normal enable
        setGlobalUDLDConfig udldGlobalConfig message_interval 10
```

```
UCS(UDLDPortType)# setGlobalUDLDConfig commit
```

```
 SetGlobalUDLDConfig Commit Success!!!
```

```
UCS(UDLDPortType)# exit
```

This enables UDLD on all ports of ME 1200 NID with a time interval of 10 seconds.

Disabling UDLD Mode Globally

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**UDLDPortType** **Example:** `UCS# UDLDPortType`	Enters the UDLD mode.
Step 3	**udldGlobalConfig** {**mode** {**normal** | **aggressive**} | **message-interval**} **Example:** `UCS(UDLDPortType)# setGlobalUDLDConfig udldGlobalConfig mode normal disable`	Enters UDLD global configuration mode. Sub-command options: • **mode—**Specifies UDLD configuration mode. ◦ **normal—**Enables UDLD in normal mode on all fiber-optic ports. ◦ **aggressive—**Enables UDLD in aggressive mode on all fiber-optic ports. • **message-interval—**Specifies time interval between UDLD probe messages on ports. The valid range is from 7 to 90 seconds.
Step 4	**setGlobalUDLDConfig review** **Example:** `UCS(UDLDPortType)# setGlobalUDLDConfig review`	(Optional) Displays the configuration.
Step 5	**setGlobalUDLDConfig commit** **Example:** `UCS(UDLDPortType)# setGlobalUDLDConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(UDLDPortType)# exit`	Exits the UDLDPortType mode.

Configuration Example

- The example shows how to disable UDLD globally in normal mode:

```
UCS# UDLDPortType
UCS(UDLDPortType)# setGlobalUDLDConfig udldGlobalConfig mode normal disable
UCS(UDLDPortType)# setGlobalUDLDConfig review
```

```
Commands in queue:
        setGlobalUDLDConfig udldGlobalConfig mode normal disable
```

```
UCS(UDLDPortType)# setGlobalUDLDConfig commit
```

```
 SetGlobalUDLDConfig Commit Success!!!
```

```
UCS(UDLDPortType)# exit
```

This disables UDLD on all ports of ME 1200 NID.

Enabling UDLD Mode on a Port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS#Configure NID 1`	Opens a new session for NID 1.
Step 2	**UDLDPortType** **Example:** `UCS# UDLDPortType`	Enters the UDLD mode.

Step 3	**udldInterfaceConfig** {**mode** {**enable** | **aggressive**} | **message-interval** *seconds* | **port** *port number*}
	
	Example:
	`UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig mode aggressive enable`
	`UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig message_interval 20`
	`UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig port 3`
		Enters UDLD global configuration mode.
		
		Sub-command options:
		
		• **mode—**Specifies UDLD configuration mode.
		  ◦ **enable—**Enables UDLD in normal mode on selected interface.
		  ◦ **aggressive—**Enables UDLD in aggressive mode on selected interface.
		
		• **message-interval—**Specifies time interval between UDLD probe messages on ports.
		  ◦ *seconds***—**Time interval in seconds. Valid range is from 7 to 90 seconds.
		
		• **port—**Specifies targeted interface.

	Command or Action	Purpose
		◦ *port number*—Specifies number of the selected port.
Step 4	**setIntfUDLDConfig review** **Example:** `UCS(UDLDPortType)# setIntfUDLDConfig review`	(Optional) Displays the configuration.
Step 5	**setIntfUDLDConfig commit** **Example:** `UCS(UDLDPortType)# setIntfUDLDConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(UDLDPortType)# exit`	Exits the UDLDPortType mode.

Configuration Example

- The example shows how to enable UDLD on a specific port in an aggressive mode:

```
UCS# UDLDPortType
UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig mode aggressive enable
UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig message_interval 20
UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig port 3
UCS(UDLDPortType)# setIntfUDLDConfig review
```

```
Commands in queue:
       UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig mode
 aggressive enable
        UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig
message_interval 20
       UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig port
 3
```

```
UCS(UDLDPortType)# setIntfUDLDConfig commit
```

```
 SetIntfUDLDConfig Commit Success!!!
```

```
UCS(UDLDPortType)# exit
```

This enables UDLD in aggressive mode only on port 3 of ME 1200 NID with a time interval of 20 seconds.

Disabling UDLD Mode on a Port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**UDLDPortType** **Example:** `UCS# UDLDPortType`	Enters the UDLD mode.

Step 3	**udldInterfaceConfig** {**mode** {**enable** | **aggressive**} | **message-interval** *seconds* | **port** *port number*}
	
	Example:
	`UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig port 3`
	`UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig mode aggressive disable`
		Enters UDLD port specific configuration mode.
		Sub-command options:
		• **mode**—Specifies UDLD configuration mode.
		  ◦ **enable**—Enables UDLD in normal mode on selected interface.
		  ◦ **aggressive**—Enables UDLD in aggressive mode on selected interface.
		• **message-interval**—Specifies time interval between UDLD probe messages on ports.
		  ◦ *seconds*—Time interval in seconds. Valid range is from 7 to 90 seconds.
		• **port**—Specifies targeted interface.
		  ◦ *port number*—Specifies number of the selected port.

Step 4	**setIntfUDLDConfig review** **Example:** `UCS(UDLDPortType)# setIntfUDLDConfig review`	(Optional) Displays the configuration.
Step 5	**setIntfUDLDConfig commit** **Example:** `UCS(UDLDPortType)# setIntfUDLDConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(UDLDPortType)# exit`	Exits the UDLDPortType mode.

Configuration Example

- The example shows how to disable UDLD on a specific port in an aggressive mode:

```
UCS# UDLDPortType
UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig port 3
UCS(UDLDPortType)# setIntfUDLDConfig udldInterfaceConfig mode aggressive disable
UCS(UDLDPortType)# setIntfUDLDConfig review
```

```
Commands in queue:
        setIntfUDLDConfig udldInterfaceConfig port 3
        setIntfUDLDConfig udldInterfaceConfig mode aggressive disable
```

```
UCS(UDLDPortType)# setIntfUDLDConfig commit
```

```
 SetIntfUDLDConfig Commit Success!!!
```

```
UCS(UDLDPortType)# exit
```

This disables UDLD in an aggressive mode on port 3 of ME 1200 NID.

Getting Current Global UDLD Values

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS#Configure NID 1`	Opens a new session for NID 1.
Step 2	**UDLDPortType** **Example:** `UCS# UDLDPortType`	Enters the UDLD mode.
Step 3	**getGlobalUDLDConfReq** **Example:** `UCS(UDLDPortType)# getGlobalUDLDConfig getGlobalUDLDConfReq`	Enters UDLD global configuration mode.
Step 4	**getGlobalUDLDConfig review** **Example:** `UCS(UDLDPortType)# getGlobalUDLDConfig review`	(Optional) Displays the configuration.
Step 5	**getGlobalUDLDConfig commit** **Example:** `UCS(UDLDPortType)# getGlobalUDLDConfig commit`	Sends the configuration to NID.

	Command or Action	Purpose
Step 6	**exit** **Example:** `UCS(UDLDPortType)# exit`	Exits the UDLDPortType mode.

Configuration Example

- The example shows how to get current global UDLD values:

```
UCS# UDLDPortType
UCS(UDLDPortType)# getGlobalUDLDConfig getGlobalUDLDConfReq
UCS(UDLDPortType)# getGlobalUDLDConfig review
```

```
Commands in queue:
        getGlobalUDLDConfig getGlobalUDLDConfReq
```

```
UCS(UDLDPortType)# getGlobalUDLDConfig commit
```

```
GetGlobalUDLDConfig_Output.udldGlobalConfig.mode.t = 1
GetGlobalUDLDConfig_Output.udldGlobalConfig.mode.u.normal = false
GetGlobalUDLDConfig_Output.udldGlobalConfig.message_interval = 7

 GetGlobalUDLDConfig Commit Success!!!
```

```
UCS(UDLDPortType)# exit
```

The following is a sample output on the NID.

```
Decoding of Request message was successful
Decoded record:
GetGlobalUDLDConfig_Input.getGlobalUDLDConfReq = '0'

 Set UDLD global to defaultsEncoding of Response message was successful
Encoded record:
GetGlobalUDLDConfig_Output.udldGlobalConfig.mode.t = 1
GetGlobalUDLDConfig_Output.udldGlobalConfig.mode.u.normal = false
GetGlobalUDLDConfig_Output.udldGlobalConfig.message_interval = 7
GetGlobalUDLDConfig_Output.xmlns:ns0 =
"http://new.webservice.namespace"
GetGlobalUDLDConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetGlobalUDLDConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetGlobalUDLDConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetGlobalUDLDConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetGlobalUDLDConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

Getting Current Port Specific UDLD Values

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ConfigureNID** **Example:** `UCS#Configure NID 1`	Opens a new session for NID 1.
Step 2	**UDLDPortType** **Example:** `UCS# UDLDPortType`	Enters the UDLD mode.
Step 3	**etGlobalUDLDConfReq** **Example:** `UCS(UDLDPortType)# getIntfUDLDConfig udldPhyPort 3`	Enters UDLD port specific configuration mode.
Step 4	**setGlobalUDLDConfig review** **Example:** `UCS(UDLDPortType)# getIntfUDLDConfig review`	(Optional) Displays the configuration.
Step 5	**setGlobalUDLDConfig commit** **Example:** `UCS(UDLDPortType)# getIntfUDLDConfig commit`	Sends the configuration to NID.
Step 6	**exit** **Example:** `UCS(UDLDPortType)# exit`	Exits the UDLDPortType mode.

Configuration Example

- The example shows how to get current port specific UDLD values when UDLD is disabled:

```
UCS# UDLDPortType
UCS(UDLDPortType)# getIntfUDLDConfig udldPhyPort 3
UCS(UDLDPortType)# getIntfUDLDConfig review
```

```
Commands in queue:
        getIntfUDLDConfig udldPhyPort 3
```

```
UCS(UDLDPortType)# getIntfUDLDConfig commit
```

```
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.t = 1
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.u.enable = false
GetIntfUDLDConfig_Output.udldInterfaceConfig.message_interval = 7
```

```
GetIntfUDLDConfig_Output.udldInterfaceConfig.port = 3

 GetIntfUDLDConfig Commit Success!!!
```

```
UCS(UDLDPortType)# exit
```

The following is a sample output on the NID.

```
# Decoding of Request message was successful
Decoded record:
GetIntfUDLDConfig-Input.udldPhyPort = 3
Set UDLD intf to defaultsEncoding of Response message was successful
Encoded record:
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.t = 1
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.u.enable = false
GetIntfUDLDConfig_Output.udldInterfaceConfig.message_interval = 7
GetIntfUDLDConfig_Output.udldInterfaceConfig.port = 3
GetIntfUDLDConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetIntfUDLDConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetIntfUDLDConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetIntfUDLDConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetIntfUDLDConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetIntfUDLDConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

- The example shows how to get current port specific UDLD values when UDLD is enabled:

```
UCS# UDLDPortType
UCS(UDLDPortType)# getIntfUDLDConfig udldPhyPort 3
UCS(UDLDPortType)# getIntfUDLDConfig review
```

```
Commands in queue:
        getIntfUDLDConfig udldPhyPort 3
```

```
UCS(UDLDPortType)# getIntfUDLDConfig commit
```

```
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.t = 1
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.u.enable = true
GetIntfUDLDConfig_Output.udldInterfaceConfig.message_interval = 7
GetIntfUDLDConfig_Output.udldInterfaceConfig.port = 3

 GetIntfUDLDConfig Commit Success!!!
```

```
UCS(UDLDPortType)# exit
```

The following is a sample output on the NID.

```
# Decoding of Request message was successful
Decoded record:
GetIntfUDLDConfig-Input.udldPhyPort = 3

Set UDLD intf to defaultsEncoding of Response message was successful
Encoded record:
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.t = 1
GetIntfUDLDConfig_Output.udldInterfaceConfig.mode.u.enable = true
GetIntfUDLDConfig_Output.udldInterfaceConfig.message_interval = 7
```

```
GetIntfUDLDConfig_Output.udldInterfaceConfig.port = 3
GetIntfUDLDConfig_Output.xmlns:ns0 = "http://new.webservice.namespace"
GetIntfUDLDConfig_Output.xmlns:http =
"http://schemas.xmlsoap.org/wsdl/http/"
GetIntfUDLDConfig_Output.xmlns:mime =
"http://schemas.xmlsoap.org/wsdl/mime/"
GetIntfUDLDConfig_Output.xmlns:soap =
"http://schemas.xmlsoap.org/wsdl/soap/"
GetIntfUDLDConfig_Output.xmlns:soapenc =
"http://schemas.xmlsoap.org/soap/encoding/"
GetIntfUDLDConfig_Output.xmlns:wsdl =
"http://schemas.xmlsoap.org/wsdl/"
```

Verifying UDLD

Use the following command to verify the UDLD status on the UCS controller.

- **showUDLDStatusReq**

  This command displays the UDLD configuration status on the NID. The following is a sample output from the command:

```
ucs(config-controller-UDLDPortType)# showUDLDStatus showUDLDStatusReq
ucs(config-controller-UDLDPortType)# showUDLDStatus review
```

```
Commands in queue:
               showUDLDStatus showUDLDStatusReq
```

```
ucs(config-controller-UDLDPortType)# showUDLDStatus commit
```

```
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].port = 1
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].mode =
'Disable'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].messageInterval
 = 7
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].adminState
 = false
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].localDeviceId
 = 'B8-38-61-68-7B-BC'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].localDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].bidirState
 = 'Indeterminant'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].nbrPortID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].nbrDeviceID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].nbrDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[0].nbrLinkState
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].port = 2
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].mode =
'Disable'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].messageInterval
 = 7
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].adminState
```

```
 = false
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].localDeviceId
 = 'B8-38-61-68-7B-BC'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].localDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].bidirState
 = 'Indeterminant'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].nbrPortID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].nbrDeviceID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].nbrDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[1].nbrLinkState
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].port = 3
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].mode =
'Normal'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].messageInterval
 = 10
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].adminState
 = true
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].localDeviceId
 = 'B8-38-61-68-7B-BC'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].localDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].bidirState
 = 'Indeterminant'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].nbrPortID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].nbrDeviceID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].nbrDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[2].nbrLinkState
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].port = 4
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].mode =
'Normal'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].messageInterval
 = 10
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].adminState
 = true
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].localDeviceId
 = 'B8-38-61-68-7B-BC'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].localDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].bidirState
 = 'Indeterminant'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].nbrPortID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].nbrDeviceID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].nbrDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[3].nbrLinkState
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].port = 5
```

```
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].mode =
'Normal'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].messageInterval
 = 10
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].adminState
 = true
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].localDeviceId
 = 'B8-38-61-68-7B-BC'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].localDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].bidirState
 = 'Indeterminant'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].nbrPortID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].nbrDeviceID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].nbrDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[4].nbrLinkState
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].port = 6
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].mode =
'Disable'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].messageInterval
 = 7
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].adminState
 = false
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].localDeviceId
 = 'B8-38-61-68-7B-BC'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].localDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].bidirState
 = 'Indeterminant'
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].nbrPortID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].nbrDeviceID
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].nbrDeviceName
 = ''
ShowUDLDStatus_Output.showUDLDStatusResp.udldStatusList[5].nbrLinkState
 = ''
```

Additional References

Related Documents

Related Topic	Document Title
Cisco ME 3800x and ME 3600x Switches Software Configuration Guide, Cisco IOS Release 15.4(1)S	http://www.cisco.com/c/en/us/td/docs/switches/metro/me3600x_3800x/software/release/15-4_1_S/configuration/guide/3800x3600xscg.html

MIBs

MIB	MIBs Link
MIBs Supporting Cisco IOS	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

CHAPTER 31

Configuring LST

This chapter describes how to configure Link State Tracking feature on the UCS Controller.

Prerequisites for Configuring LST

- Configure UP MEP using ProvisionMepPortType template. To know more, refer creating MEP configuration.
- Configure ccmTLV in UP MEP in LSTPortType template.

Note While configuring UP MEP, the ports should be of same VLAN.

Understanding How Link State Tracking Works

Link-state tracking, also known as trunk failover, is a feature that binds the link state of multiple interfaces. When LST is enabled in an instance, Local SF or received 'isDown' in CCM Interface Status TLV, will bring down the residence port. Only valid in Up-MEP. The CCM rate must be 1 f/s or faster.

Configuring mepTLV

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**LSTPortType** **Example:** `uCS# LSTPortType`	Enters the LST Port Type configuration mode.
Step 3	**mepTLVConfiguration** **Example:** `UCS(LSTPortType)# mepTLVConfiguration`	Enters mepTLVConfiguration mode.

Step 4	**ccmTLVConfigccmEnabled**{**enable**|**disable**}
	
	Example:
	`UCS(LSTPortType)# mepTLVConfiguration`
	`UCS(LSTPortType)# mepTLVConfiguration ccmTLVConfig ccmEnabled enable`
		Enables or Disables ccmTLVConfig.

Step 5	**mepTLVConfigurationccmTLVConfigmepInstance***mepInstance_id*
	
	Example:
	`UCS(LSTPortType)# mepTLVConfiguration`
	`UCS(LSTPortType)# mepTLVConfiguration ccmTLVConfig mepInstance 1-100`
		Creates Link State Tracking Configuration at MEP instance number. The valid number is 1 to 100.

Step 6	**mepTLVConfiguration review** **Example:** `UCS(LSTPortType)# mepTLVConfiguration review`	Reviews the mepTLVConfiguration mode.
Step 7	**mepTLVConfiguration commit** **Example:** `UCS(LSTPortType)# mepTLVConfiguration commit`	Sends the mepTLVConfiguration to the NID.
Step 8	**exit** **Example:** `UCS(LSTPortType)# exit`	Exits the LSTPortType mode.

Checking ccmTLV Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**LSTPortType** **Example:** `UCS# LSTPortType`	Enters LST Port Type configuration mode.

Step 3	**LinkStateTrackingConfigurationlinkStateTrackingConfigmepInstance***mepInstance_id*
	
	Example:
	`UCS(LSTPortType)# LinkStateTrackingConfiguration`
	`UCS(LSTPortType)# LinkStateTrackingConfiguration linkStateTrackConfig mepInstance 1-100`
		Creates Link State Tracking Configuration at MEP instance number. The valid number is 1 to 100.

Step 4	**getmepTLVConfiguration review** **Example:** `UCS(LSTPortType)# getmepTLVConfiguration review`	Reviews the mepTLVConfiguration mode.
Step 5	**getmepTLVConfiguration commit** **Example:** `UCS(LSTPortType)# getmepTLVConfiguration commit`	Sends the mepTLVConfiguration to the NID.

Configuring LST

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**LSTPortType** **Example:** `UCS# LSTPortType`	Enters the LSTPortType configuration mode.

	Command or Action	Purpose
Step 3	**LinkStateTrackingConfiguration** **Example:** `UCS(LSTPortType)# LinkStateTrackingConfiguration`	Enters LinkStateTrackingConfiguration mode.

Step 4	**LinkStateTrackingConfigurationlinkStateTrackingConfigmepInstance***mepInstance_id*
	
	Example:
	`UCS(LSTPortType)# LinkStateTrackingConfiguration`
	`UCS(LSTPortType)# LinkStateTrackingConfiguration linkStateTrackConfig mepInstance 1-100`
		Creates Link State Tracking Configuration at MEP instance number. The valid number is 1 to 100.

Step 5	**LinkStateTrackingConfigurationlinkStateTrackingConfiglstEnabled**{**disable**|**enable**}
	
	Example:
	`UCS(LSTPortType)# LinkStateTrackingConfiguration`
	`UCS(LSTPortType)# LinkStateTrackingConfiguration linkStateTrackConfig lstEnabled enable/disable`
		Sets up LSTPortType configuration.
		• enable- Enables the LST configuration on the ports.
		• disable- Disables the LST configuration on the ports.

Step 6	**LinkStateTrackingConfiguration review** **Example:** `UCS(LSTPortType)# LinkStateTrackingConfiguration review`	Reviews the LinkStateTrackingConfiguration mode.
Step 7	**LinkStateTrackingConfiguration commit** **Example:** `UCS(LSTPortType)# LinkStateTrackingConfiguration commit`	Sends the LinkStateTrackingConfiguration to the NID.
Step 8	**exit** **Example:** `UCS(LSTPortType)# exit`	Exits the LSTPortType mode.

Checking LST Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**LSTPortType** **Example:** `UCS# LSTPortType`	Enters the LSTPortType configuration mode.
Step 3	**getLSTConfiguration** **Example:** `UCS(LSTPortType)# getLSTConfiguration`	Gets the link state tracking configuration response.

Step 4	**getLSTConfigurationmepInstance***mepInstance_id*
	
	Example:
	`UCS(LSTPortType)# getLSTConfiguration`
	`UCS(LSTPortType)# getLSTConfiguration mepInstance 1-100`
		Creates Link State Tracking Configuration at MEP instance number. The valid number is 1 to 100.

Step 5	**getLSTConfiguration review** **Example:** `v(LSTPortType)# getLSTConfiguration review`	Reviews the LSTConfiguration mode.
Step 6	**getLSTConfiguration commit** **Example:** `UCS(LSTPortType)# getLSTConfiguration commit`	Sends the LSTConfiguration mode to the NID.

Viewing LST Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**LSTPortType** **Example:** `UCS# LSTPortType`	Enters the LSTPortType configuration mode.
Step 3	**displayLSTConfiguration** **Example:** `UCS(LSTPortType)# displayLSTConfiguration`	Displays the link state tracking configuration response.

	Command or Action	Purpose

Step 4	**displayLSTConfigurationmepInstance***mepInstance_id*
	
	Example:
	`UCS(LSTPortType)# displayLSTConfiguration`
	`UCS(LSTPortType)# displayLSTConfiguration mepInstance 1-100`
		Displays Link State Tracking Configuration at MEP instance number. The valid number is 1 to 100.

Step 5	**displayLSTConfiguration review** **Example:** `UCS(LSTPortType)# displayLSTConfiguration review`	Display the LSTConfiguration mode.
Step 6	**displayLSTConfiguration commit** **Example:** `UCS(LSTPortType)# displayLSTConfiguration commit`	Displays the LSTConfiguration mode in the NID.

CHAPTER 32

Configuring Flex Links

This document describes the Flex Link feature and configuration steps to implement Flex Links. They also describe how to configure the MAC address table move update feature.

Note Flex Links does not currently support stacking, duo switches structure, or VLAN separation.

Prerequisites for Configuring Flex Links

- Disable STP before configuring Flex Links. If STP is disabled on the switch, make sure that there are no Layer 2 loops in the topology.
- Flex Links is supported on the Serval CEServices application.

Default Configuration

Default Flex Links configuration is when there is no configuration for Flex Links pairs or for the MAC address move update transmit feature.

Restrictions for Configuring Flex Links

- Only one Flex Links backup link can be configured for any active link, and it must be a different interface from the active interface.

- The backup link does not have to be the same type as the active link. However, they should be configured with similar characteristics so that there are no loops or changes in operation if the standby link becomes active.
- An active link cannot belong to another Flex Links pair.
- The Flex Links pair cannot belong to the same port channel. However, a Flex Links pair can be a port channel and a physical interface, or two port channels or physical interfaces.
- The port channel interface should be active when included in the Flex Links pair, for it to be configured properly.

Information about Flex Links

Flex Links configuration provides link-level redundancy in the absence of Spanning Tree Protocol (STP). Flex Links consists of a pair of interfaces (ports or port channels) with one interface configured as the primary interface (forwarding status) and the other as the backup interface (standby status). When a failure occurs on the primary interface, the backup interface moves to forwarding status and starts to forward traffic.

Flex Links works by detecting link down on a primary interface and then bringing up the backup interface that has been defined as backup. It is most commonly implemented at the access layer where the switch has dual uplinks to the distribution layer.

Flex Links is designed to interact with supporting modules, such as the port module, the aggregation module, the packet module, and the configuration module. The basic Flex Links protocol functions are as follows:

- Initialize module configurations
- Interact with the packet module to transmit/receive MAC address table update frames
- Interact with the configuration module to read/write FL configurations
- Register with the port module to receive the port up/down event

The Flex Links API layer provides direct interaction with the switch for the implementation of the active and backup ports groups, the setup of the port status, and the MAC-address table read.

MAC Address Table Move Update

The MAC address table move update is an optional Flex Links feature. It allows the switch to provide rapid bidirectional convergence when an active link goes down and the backup link starts forwarding traffic.

Note MAC address table move update enables fast recovery of network connectivity but consumes CPU resources.

How to Configure Flex Links

Configuring Flexlink Ports

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**FlexlinksPortType** **Example:** `UCS# FlexlinksPortType`	Enters FlexlinksPortType mode to provision Flex Links.

Step 3	**flexlinkPortConfigurationflexlinksConfiguration**{**activePort**|**backupPort**|**flexlinkEnabled**}
	
	Example:
	`UCS(FlexlinkPortType)#flexlinkPortConfiguration flexlinksConfiguration activePort activePortId 4`
	`flexlinkPortConfiguration flexlinksConfiguration backupPort backupPortId 6`
	`flexlinkPortConfiguration flexlinksConfiguration flexlinkEnabled enable`
		• **activePort—** Enter the Port number of interface to be configured .
		• **backupPort—** Enter the backup interface port number. It can be a physical port number or LLAG/LACP group ID.
		• **flexlinkEnabled—** Enter enable or disable to configure Flexlink port number.

Step 4	**flexlinkPortConfiguration review** **Example:** `UCS(FlexlinksPotType)# flexlinkPortConfiguration review`	Displays the Flexlink configuration commands in the queue.
Step 5	**flexlinkPortConfiguration commit** **Example:** `UCS(FlexlinksPotType)# flexlinkPortConfiguration commit`	Sends the Flexlink port configuration to the NID.
Step 6	**exit** **Example:** `UCS(FlexLinksPortType)# exit`	Exits the Flex Links provisioning mode.

Provisioning the UCS Controller to Configure Flex Links

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**FlexLinksPortType** **Example:** `UCS#`**`ProvisionFlexLinksPortType`**	Enters Flex Links provisioning mode.

Step 3	**ProvisionFlexlinksPortType** {**do** | **end** | **exit** | **flexlinkPortConfiguration** | **getFlexlinksConfiuration** | **getMACMoveUpdateConfig** | **help** | **macMoveUpdatePortConfig** | **showFlexlinksConfig**}
	
	Example:
	`UCS(ProvisionFlexlinksPortType)# ?`
	`  ProvisionFlexLinksPortType sub-mode commands:`
	`  do                               To run exec commands in config mode`
	
	`  end                               Go back to EXEC mode`
	`  exit                             Exit from ProvisionFlexlinksPortType`
	` sub configuration mode`
	`  FlexlinkPortConfiguration        Get Flexlinks configuration response`
	
	`  getFlexlnksConfiguration         Get Flexlinks configuration request`
	
	`  getMACMoveUpdateConfig           Get Mac Move Update config response`
	
	`  help                             Description of the interactive help`
	` system`
	`  macMoveUpdatePortConfig           Show MAC move update Response`
	`  showFlexlinksConfig               Display Flexlinks configuration`
	`response`
		Displays the supported configurations for Flex Links.

Step 4	**exit** **Example:** `UCS(ProvisionFlexlinksPortType)# `**`exit`**	Exits the Flex Links provisioning mode.

Configuration Example

The following example shows the supported Flex Link configuration:

```
UCS(ProvisionFlexlinksPortType)# ?
  ProvisionFlexLinksPortType sub-mode commands:
  do                               To run exec commands in config mode
```

```
  end                                Go back to EXEC mode
  exit                               Exit from ProvisionFlexlinksPortType sub configuration
mode
  FlexlinkPortConfiguration          Get Flexlinks configuration response
  getFlexlnksConfiguration           Get Flexlinks configuration request
  getMACMoveUpdateConfig             Get Mac Move Update config response
  help                               Description of the interactive help system
  macMoveUpdatePortConfig            Show MAC move update Response
  showFlexlinksConfig                Display Flexlinks configuration response
```

Viewing Flex Link Configuration at Port Level

Before You Begin

- Perform the steps to provision Flex Links on the NID.

DETAILED STEPS

	Command or Action	Purpose

Step 1	**getFlexlinksConfig** {**commit** | **flexlinksPhysicalPort** *port_num* | **review**}
	
	Example:
	`UCS(FlexlinksPortType)# getflexlinkConfiguration ?`
	` commit                    commit getFlexlinksConfig`
	` flexlinksPhysicalPort   Get flexlinks physical port`
	`configuration request`
	` review                    review getFlexlinksConfig`
	`commands`
	`UCS(FlexlinksPortType)#`
	`getFlexlinksConfigRequestportphyPortId 1`
		Retrieves the Flex Links configuration at port.
		• **commit—**Sends the Flex Links configuration to NID.
		• **flexlinksPhysicalPort** *port-num*—Retrieves the Flex Links configuration for specified port on the controller.
		• **review—**Displays the configuration on the controller.

Step 2	**getFlexlinksConfiguration review**
	
	Example:
	`UCS(FlexlinksPortType)# getFlexlinksConfiguration review`
	`Commands in queue:`
	`       getFlexlinksConfiguration flexlinksPhysicalPort`
	` 1`
		Displays the Flex Links configuration.

Step 3	**getFlexlinksConfiguration commit** **Example:** `UCS(FlexlinksPortType)# getFlexlinksConfiguration commit`	Sends the Flex Links configuration to the NID.
Step 4	**exit** **Example:** `UCS(FlexLinksPortType)# exit`	Exits the Flex Links provisioning mode.

Configuration Example

The example retrieves the Flex Links configuration for port 1 on the NID:

```
UCS(FlexlinksPortType)# getFlexlinksConfigRequestportphyPortId 1
UCS(FlexlinksPortType)# getFlexlinksConfiguration review
Commands in queue:
       getFlexlinksConfigRequestportphyPortId 1
UCS(FlexlinksPortType)# getFlexlinksConfiguration commit
GetFlexlinksConfiguration_Output.getFlexlinksConfiguration.portNumber = 1
GetFlexlinksConfiguration_Output.getFlexlinksConfiguration.flexlinksEnable = false
GetFlexlinksConfiguration_Output.getFlexlinksConfiguration.key = 1
GetFlexlinksConfiguration_Output.getFlexlinksConfiguration.role.t = 1
GetFlexlinksConfiguration_Output.getFlexlinksConfiguration.role.u.active = true
GetFlexlinksConfiguration_Output.getFlexlinksConfiguration.portPriority = 32768
GetFlexlinksConfiguration_Output.getFlexlinksConfiguration.timeout.t = 1
getFlexlinksConfiguration_Output.getFlexlinksConfiguration.timeout.u.fast = true

 GetFlexlinksConfiguration Commit Success!!!
UCS(FlexlinksPortType)# exit
```

Displaying Flex Link ActivePort Configuration

Before You Begin

- Perform the following steps to display Flex Links active port configuration on the NID.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure nid** **Example:** `UCS# configure nid 1`	Opens a new session for NID 1.
Step 2	**FlexlinksPortType** **Example:** `UCS# FlexlinksPortType`	Enters FlexlinksPortType mode to provision Flex Links.

Step 3	**ShowFlexlinksConfig** {**commit** | **displayFlexlinksConfig** |**flush** | **review**}
	
	Example:
	`UCS(FlexlinksPortType)# showflexlinksConfig ?`
	` commit                   commit showFlexlinksConfig`
	` displayFlexlinksConfig   Display Flexlinks Configuration State on physical or LLAG interfaces`
	` review                   review getFlexlinksConfig commands`
	` flush                    flush getFlexlinksConfig commands`
	`UCS(FlexlinksPortType)# showFlexlinksConfig display FlexlinkConfig portphyPortId 1`
		Displays the Flex Links configuration at port.
		
		• **commit—**Sends the Flex Links configuration to NID.
		• **displayFlexlinksConfig—**Shows the Flex Links configuration for specified port on the NID.
		• **review—**Displays the configuration.

	Command or Action	Purpose

Step 4	**showFlexlinksConfigreview**
	
	Example:
	`UCS(FlexlinksPortType)# `**`showFlexlinksConfig review`**
	`Commands in queue:`
	`        showFlexlinksConfig flexlinksPhysicalPort 1`
		Displays the configuration

Step 5	**showFlexlinksConfigcommit** **Example:** `UCS(FlexlinksPortType)# `**`showFlexlinksConfig commit`**	Sends the Flex Links Configuration to the NID.
Step 6	**exit** **Example:** `UCS(FlexLinksPortType)# `**`exit`**	Exits the Flex Links mode.

The example shows the Flex Links configuration for port 1 on the controller:

```
UCS(FlexlinksPortType)# showFlexlinksConfig display FlexlinkConfig portphyPortId 1
UCS(FlexlinksPortType)# showFlexlinksConfig review
Commands in queue:
       showFlexlinksConfigportphyPortId 1
UCS(FlexlinksPortType)# showFlexlinksConfig commit
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.portNumber = 1
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.flexlinksEnable = false
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.key = 1
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.role.t = 1
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.role.u.active = true
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.portPriority = 32768
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.timeout.t = 1
ShowFlexlinksConfiguration_Output.showFlexlinksConfig.timeout.u.fast = true

ShowFlexlinksConfiguration Commit Success!!!
UCS(FlexlinksPortType)# exit
```

Enabling macMoveupdate on Active Port

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure nid** **Example:** `UCS# configure nid 1`	Opens a new session for NID 1.
Step 2	**FlexlinksPortType** **Example:** `UCS# FlexlinksPortType`	Enters the FlexlinksPortType mode to provision Flex Links.

	Command or Action	Purpose

Step 3	**macMoveUpdatePortConfig macMoveUpdateConfig** { **llagGroupId** *llagGroup-Id* | **mmuEnabled** { **enable|disable** } | **portNumber** *portNumber* }
	
	Example:
	`UCS(FlexlinksPortType)# macMoveUpdatePortConfig macMoveUpdateConfig`
	`UCS(FlexlinksPortType)# macMoveUpdatePortConfig macMoveUpdateConfig llagGroup id 2`
	`UCS(FlexlinksPortType)# macMoveUpdatePortConfig macMoveUpdateConfig portNumber 4`
	`UCS(FlexlinksPortType)# macMoveUpdatePortConfig macMoveUpdateConfig mmuEnabled enable`
		Displays the macMoveUpdateConfig mode.
		Note User can enable macMoveUpdate, only after flex link is configured.
		• **llagGroupId—** Configures llag as an active port in flex link. The range is from 1-4
		• **portNumber—** Configures port number in flex link. The range is from 1-124
		• **mmuEnabled—**Updates the MAC Move Transmitr in flex link to either enale o disable.

Step 4	**macMoveUpdatePortConfig review** **Example:** `UCS(FlexlinksPotType)# macMoveUpdatePortConfig review`	Displays the macMoveUpdatePortConfig commands.
Step 5	**macMoveUpdatePortConfig commit** **Example:** `UCS(FlexlinksPotType)# macMoveUpdatePortConfig commit`	Sends the macMoveUpdatePortConfig commands to the NID.
Step 6	**exit** **Example:** `UCS(FlexLinksPortType)# exit`	Exits the Flex Links mode.

Viewing macMoveUpdate Active Port Configuration

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**FlexlinksPortType** **Example:** `UCS# FlexlinksPortType`	Enters FlexlinksPortType mode to provision Flex Links.
Step 3	**getMACMoveUpdateConfiggetFlexlinkConfigRequestport**{**llagGroupId**|**phyPortId**}	• **port—** Displays the targeted active port.

	Command or Action	Purpose
	Example: `UCS(FlexlinksPortType)# getMACMoveUpdateConfig getFlexlinkConfigRequest port phyPortId 4`	• **llagGroupID—** Displays the active llagGroupID number. • **phyPortId—** Displays the active physical port number.
Step 4	**getMACMoveUpdateConfig review** **Example:** `UCS(FlexlinksPotType)# getMACMoveUpdateConfig review`	Displays the ports for Flexlink configuration.
Step 5	**getMACMoveUpdateConfig commit** **Example:** `UCS(FlexlinksPotType)# getMACMoveUpdateConfig review`	Sends the Flexlink configuration to the NID.
Step 6	**exit** **Example:** `UCS(FlexLinksPortType)# exit`	Exits the Flex Links mode.

Configuration Example

The example shows the flexlink configuration in active ports.

```
UCS# FlexlinksPortType
UCS(FlexlinksPortType)# getMACMoveUpdateConfig getFlexlinkConfigRequest port phyPortId 4


UCS(FlexlinksPortType)# getMACMoveUpdateConfig review

Commands in queue: 1

getMACMoveUpdateConfig getFlexlinkConfigRequest port phyPortId 4
UCS(FlexlinksPortType)# getMACMoveUpdateConfig commit
GetMACMoveUpdateConfig_Output.macMoveUpdateConfig._choice1.t = 1
GetMACMoveUpdateConfig_Output.macMoveUpdateConfig._choice1.u.portNumber = 4
GetMACMoveUpdateConfig_Output.macMoveUpdateConfig.mmuEnabled = true

GetMACMoveUpdateConfig Commit Success!!!(FlexlinksPortType)#
```

CHAPTER 33

Configuring Y.1564

This document describes the Y.1564 test feature and configuration steps to execute Y.1564 feature.

Prerequisites for Configuring Y.1564

- You must disable:
    - Link Layer Discovery Protocol (LLDP) transmit and receive on source port.
    - Loop protection on destination port or Spanning Tree Protocol (STP) on destination and source port.
    - Spanning Tree Protocol (STP).
- NID must have an IP address.
- Loop should not be configured.

Information About Y.1564

ITU-T Y.1564 (Or sometimes called Y.156sam or EtherSAM - Ethernet Service Activation Methodology) is a QoS and network performance ITU-T Ethernet-based service test methodology. This testing procedure tests service turn-up, installation and troubleshooting of Ethernet-based services.

Y.1564 allows simultaneous testing of multiple Ethernet services and measures. It also validates the different QoS mechanisms provisioned in the network to prioritize different service types - allowing faster deployment, easier service and network troubleshooting.

Y.1564 allows simultaneous testing of multiple Ethernet services and measures. It also validates the different QoS mechanisms provisioned in the network to prioritize different service types - allowing faster deployment, easier service and network troubleshooting.

Configuring New Y.1564 Profile

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.

Step 3	**setY1564Profile y1564ProfileProfile name**|**description**|**acceptable_fdv** *acceptable_gdv*|**acceptable_flr** *acceptable_flr*|**acceptable_ftd** *acceptable-_td* | **cir_test** {**dm_interval** *dm-_nterval* |**duration** *duration* | **step_count** *step_count* | **start** { **enable** | **disable** } } | **dst_oam_ aware** { **enable** | **disable** } | **dwell_time** *dwell_time* | **eir_test** { **enable** | **disable** } | **meg_level** *meg_level duration* | **performance_test** { **enable** | **disable** } | **traffic_policing_test** { *dm_interval* | **duration** } | } **traffic_type** { **customer_simulated** | **oam** } | **emix** | **user_defined_frame_size** }
	
	Example:
	`UCS(ciscoY1564)# setY1564Profile y1564Profile profileName`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile description`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile acceptable_fdv 0-10000`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile acceptable_flr 0-1000`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile acceptable_ftd 0-10000`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile cir_test start enable`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile cir_test dm-interval 100-10000`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile cir_test duration <cr>`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile cir_test step-count 1-1000`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile dst_oam-aware enable`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile dwell_time 100-10000`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile eir_test enable`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile meg_level 0-7`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile performance_test enable`
		• **profileName—** Enter the name of the profile for Y1564 .
		• **description—** Enter a brief description about the profile .
		• **acceptable_fdv—** Enter frame delay variation in milliseconds to configure Y1564. Acceptable limit is 0-10000. The default value is 0, which disables the test.
		• **acceptable_flr—** Enter frame loss ratio per mile to configure Y1564. Acceptable limit is 0-1000. The default value is 0 and 1000 disables the test.
		• **acceptable_ftd—** Enter frame transfer delay in milliseconds to configure Y1564. Acceptable limit is 0-10000.Acceptable limit is 0-10000. The default value is 0, which disables the test.
		• **cir_test—** Enter frame delay variation in milliseconds to configure Y1564. Acceptable limit is 0-10000.
		◦ **dm_interval—** Enter the interval of between sending delay measurement frame. Acceptable limit is 100-10000.
		◦ **duration—** Enter the duration of one step.
		◦ **step_count—** Enter the number of steps to configure CIR. Acceptable limit is 1-1000
		◦ **start—**Enter enable or disable to configure CIR test.
		• **dst_oam_aware—** Enter enable or disable to configure remote end Y.1731 OAM aware.
		• **dwell_time—** Enter the time frame of execution, pauses in milliseconds after each trial before reading counters, and status from hardware. Acceptable limit is 100-10000. Default value is 500.

	Command or Action	Purpose

	`UCS(ciscoY1564)# setY1564Profile y1564Profile traffic_policing_test dm_interval 100-10000`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile traffic_policing_test duration <cr>`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile traffic_policing_test duration start enable`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile traffic_type`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile traffic_type customer-simulated <cr>`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile traffic_type oam <cr>`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile emix`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile e emix U`
	`UCS(ciscoY1564)# setY1564Profile y1564Profile user_defined_frame_size <cr>`
		• **eir_test—** Enter EIR configuration test and optionally set its parameters to configure Y1564. Parameters are dm-interval, duration and start.
		• **meg_level—** Enter the profile MEG level to configure Y1564 . Acceptable limit is 0-7.
		• **performance_test—** Enter the performance test parameters.
		◦ **dm_interval—** Enter the time interval in milliseconds between sending delay measurement frame. Acceptable limit is 100-10000.
		◦ **duration—** Enter the duration of performance test.
		◦ **start—**Enter enable or disable to start the performance test.
		• **traffic_policing_test—** Enter the traffic policing test parameters..
		◦ **dm_interval—** Enter the time interval in milliseconds between sending delay measurement frame. Acceptable limit is 100-10000.
		◦ **duration—** Enter the duration of traffic policing test.
		◦ **start—**Enter enable or disable to start the traffic policing test.
		• **traffic_type—** Enter the type of traffic generated at the near end.
		◦ **customer_simulated—** Enter the frames that simulate real customer traffic as background traffic.
		◦ **oam—** Enter the duration of traffic policing test.
		• **emix—** select the frame size(EMIX letter-encoded) that the enabled tests will use. Encoding is as follows: a: 64, b: 128, c: 256, d: 512, e: 1024,f: 1280, g: 1518, h: MTU, u: user-defined.
		• **user_defined_frame_size—** Enter the frame size if emix is set to 'U'.

Step 4	**review** **Example:** `UCS(ciscoY1564)# setY1564Profile review`	Reviews the ciscoY1564 profile configuration parameters.

	Command or Action	Purpose
Step 5	**commit** **Example:** `UCS(ciscoY1564)# setY1564Profile commit`	Sends the ciscoY1564 profile parameters to the NID.
Step 6	**exit** **Example:** `UCS(ciscoY1564)# exit`	Exits ciscoY1564 profile configuration mode.

Getting the Profile Configuration using Profile Name

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.
Step 3	**getY1564ProfilegetY1564ProfileReq** **Example:** `UCS(ciscoY1564)# getY1564Profile getY1564ProfileReq`	Retrieves the Profile configuration.
Step 4	**getY1564Profilereview** **Example:** `UCS(ciscoY1564)# getY1564Profile review`	Displays the Y.1564 profile configuration.
Step 5	**getY1564Profilecommit** **Example:** `UCS(ciscoY1564)# getY1564Profile commit`	Sends the Y.1564 profile configuration information to the NID.
Step 6	**exit** **Example:** `UCS(ciscoY1564)# exit`	Exits ciscoY1564 profile configuration mode.

The following example shows the Profile Configuration using the Profile Name:

```
UCS # getY1564Profile commit
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.profileName = 'cisco123456'
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.description = 'oamunaware'
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.acceptable_fdv = 0
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.acceptable_ftd = 0
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.acceptable_flr = 0
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.dst_oam_aware = false
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.dwell_time = 500
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.emix = '1024'
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.meg_level = 7
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.traffic_type.t = 1
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.traffic_type.u.oam = '0'
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.user_defined_frame_size = 2000
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.cir_test.start = true
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.cir_test.duration = 60
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.cir_test.dm_interval = 500
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.cir_test.step_count = 4
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.eir_test.start = true
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.eir_test.duration = 60
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.eir_test.dm_interval = 500
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.performance_test.start = true
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.performance_test.duration = 10
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.performance_test.dm_interval = 100
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.traffic_policing_test.start = true
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.traffic_policing_test.duration =
10
GetY1564Profile_Output.y1564Profile.y1564Profile_ELEM_0.traffic_policing_test.dm_interval
= 100

 GetY1564Profile Commit Success
```

Viewing Profile Names

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.
Step 3	**showY1564showY1564Req**{**profiles**|**reports**} **Example:** `UCS(ciscoY1564)# showY1564 showY1564Req`	Displays existing profiles or report information.

	Command or Action	Purpose
Step 4	**showY1564review** **Example:** `UCS(ciscoY1564)# showY1564 review`	Displays the profile configurations.
Step 5	**exit** **Example:** `UCS(ciscoY1564)# exit`	Exits ciscoY1564 profile configuration mode.

The following example shows the configurations to display a particular Profiles using the profile name or description:

```
UCS (ciscoy1564)# showY1564 commit
ShowY1564_Output.showY1564Resp.t = 1
ShowY1564_Output.showY1564Resp.u.profile[0].profileName = 'NewProfile1'
ShowY1564_Output.showY1564Resp.u.profile[0].description = ''
ShowY1564_Output.showY1564Resp.u.profile[1].profileName = 'cisco123456'
ShowY1564_Output.showY1564Resp.u.profile[1].description = 'oamunaware'
ShowY1564 Commit Success
```

Managing Y.1564 Profile Names

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.

Step 3	**y1564ProfileManagementy1564ProfileManagmentReq**{**delete**|**rename**{**new-name**|**old_name**}}
	
	Example:
	`UCS(ciscoY1564)# y1564ProfileManagement y1564ProfileManagmentReq rename old-name cisco123456`
	`UCS(ciscoY1564)# y1564ProfileManagement y1564ProfileManagmentReq rename new-name cisco`
	`UCS(ciscoY1564)# y1564ProfileManagement y1564ProfileManagmentReq delete cisco`
		• **rename—**Set rename to rename a old profile name.
		• **delete—**Set delete to delete an existing profile.

	Command or Action	Purpose
Step 4	**review** **Example:** `UCS(ciscoY1564)# y1564ProfileManagement review`	Reviews the ciscoY1564 profile names.
Step 5	**commit** **Example:** `UCSh(ciscoY1564)# y1564ProfileManagement commit`	Sends the changed or deleted ciscoY1564 profile names to the NID .
Step 6	**exit** **Example:** `UCS(ciscoY1564)# exit`	Exits ciscoY1564 profile configuration mode.

Configuring Y.1564 Test Parameters

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.

Step 3	**setY1564TestParams y1564TestsReq** {**dei** |**description** |**dscp**|**ece** |**evc** |**interface** |**pcp**|**peer_mac** |**profile_name** |**report-name** | **vlan***vlan_id* }
	
	Example:
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq profile-name cisco123456`
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq report-name controllerreport123456`
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq description dstmodeno`
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq evc 1`
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq ece 1`
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq interface 3`
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq peer-mac 00-00-00-00-00-01`
	`UCS(ciscoY1564)#setY1564TestParams y1564TestsReq`
		• **dei—** Enter the DEI number of the profile. The valid range is 0-1.
		• **description—** Enter the description about the test.
		• **dscp—** Enter the DSCP number of the profile. The valid range is 0-63.
		• **ece—** Enter the ECE ID number of the profile, on which the test needs to be executed. The valid range is 1-1024.
		• **evc—** Enter the EVC ID number of the profile. The valid range is 1-1024.
		• **interface—** Enter the UNI port. The valid range is 1-125.

	Command or Action	Purpose
	pcp 2 UCS(ciscoY1564)#setY1564TestParams y1564TestsReq vlan untagged UCS(ciscoY1564)#setY1564TestParams y1564TestsReq dei 0 UCS(ciscoY1564)#setY1564TestParams review UCS(ciscoY1564)#setY1564TestParams commit	• **pcp—** Enter the PCP number of the profile. The valid range is 0-7. • **peer-mac—** Enter peer MAC address. • **profile-name—** Enter the name of the existing profile, that needs to be tested. • **report-name—** Enter a unique name for the test report. • **vlan—** Enter the Vlan ID.
Step 4	**review** **Example:** UCS(ciscoY1564)# setY1564TestParams review	Reviews the ciscoY1564 profile test parameters.
Step 5	**commit** **Example:** UCS(ciscoY1564)# setY1564TestParams commit	Sends the test parameter reports to the NID.
Step 6	**exit** **Example:** UCS(ciscoY1564)# exit	Exits ciscoY1564 profile configuration mode.

Configuration Example

when profile is configured as DST, then OAM-aware port and peer-mac address need to be specified in setY1564TestParams.

```
UCS# ciscoY1564
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq profile-name cisco123456
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq report-name controllerreport123456
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq description dstmodeno
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq peer-mac 00-02:01:00:01:03
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq evc 1
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq ece 1
UCS(ciscoY1564)# setY1564TestParams y1564TestsReq interface 3
UCS(ciscoY1564)# setY1564TestParams review
UCS(ciscoY1564)# setY1564TestParams commit
```

Viewing Y.1564 Test Parameters

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.
Step 3	**getY1564TestParamsgetY1564TestsReq** **Example:** `UCS(ciscoY1564)# getY1564TestParams getY1564TestsReq`	Retrieves the parameters set for latest tet.
Step 4	**review** **Example:** `UCS(ciscoY1564)# getY1564TestParams review`	Reviews the ciscoY1564 profile configuration parameters.
Step 5	**commit** **Example:** `UCS(ciscoY1564)# getY1564TestParams commit`	Sends the test parameter reports to the NID.
Step 6	**exit** **Example:** `UCS(ciscoY1564)# exit`	Exits ciscoY1564 profile configuration mode.

Saving Y.1564 Test Report

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.

	Command or Action	Purpose
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.

Step 3	**y1564ReportManagement y1564ReportManagementReq** {**save**|**stop**}
	
	Example:
	`UCS(ciscoY1564)# y1564ReportManagemen y1564ReportManagementReq save reportName controllerreport`
	`UCS(ciscoY1564)# y1564ReportManagemen y1564ReportManagementReq save tftpPath tftp://202.153.144.25/auto/tftp-blr-users1/sharsh`
		• **Save—**Set save to save a profile test report.
		• **Stop—**Set stop to stop an ongoing profile test .

Step 4	**review** **Example:** `UCS(ciscoY1564)#y 1564ReportManagement review`	Reviews the ciscoY1564 profile test report.
Step 5	**commit** **Example:** `UCS(ciscoY1564)# y1564ReportManagement commit`	sends the ciscoY1564 profile test report to the NID.
Step 6	**exit** **Example:** `UCS(ciscoY1564)# exit`	Exits ciscoY1564 profile configuration mode.

Deleting Y.1564 Test Report

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Configure NID** **Example:** `UCS# Configure NID 1`	Opens a new session for NID 1.
Step 2	**ciscoY1564** **Example:** `UCS# ciscoY1564`	Enters ciscoY1564 configuration mode.
Step 3	**y1564ReportManagement y1564ReportManagementReq** {**delete**|**stop**}	• **delete—**Set delete to delete an existing profile test report.

	Command or Action	Purpose
	Example: `UCS(ciscoY1564)# y1564ReportManagemen y1564ReportManagementReq delete controlerreport`	• **Stop—**Set stop to stop an ongoing profile test .
Step 4	**review** **Example:** `UCS(ciscoY1564)# y1564ReportManagement review`	Reviews the ciscoY1564 profile test report.
Step 5	**commit** **Example:** `UCS(ciscoY1564)# y1564ReportManagement commit`	sends the ciscoY1564 profile test report to the NID.
Step 6	**exit** **Example:** `UCS(ciscoY1564)# exit`	Exits ciscoY1564 profile configuration mode.

CHAPTER 34

Configuring Bulk Provisioning

Bulk provisioning feature allows you to add any numbers of NID to a group and all the NIDs in that group can be configured simultaneously, thus reducing time and effort to configure individual NID in your network.

Pre-requisite for Bulk Provisioning

- NID must be added to the controller.
- NID must be accessible from the controller.

How to Configure Bulk Provisioning

Creating a NID group for Bulk Provisioning

DETAILED STEPS

	Command or Action	Purpose

Step 1	**platform nid-controller add**
	
	Example:
	`UCS# platform nid-controller add 102.120.12.12`
	`Assigned NID-ID is 1`
	`Static NID Provisioning Successful!`
	`UCS# platform nid-controller add 102.120.12.11`
	`Assigned NID-ID is 2`
	`Static NID Provisioning Successful!`
	`UCS# platform nid-controller add 7.25.16.220`
	`Assigned NID-ID is 3`
	`Static NID Provisioning Successful!`
	`UCS# platform nid-controller add 7.25.17.223`
		Adds NIDs to the UCS controller.

	Command or Action	Purpose
	`Assigned NID-ID is 4` `Static NID Provisioning Successful!`	
Step 2	**platform nid-group** **Example:** `UCS# platform nid-group`	Enters NID group management mode.

Step 3	**create** | **name** *group_name* | **add-nids** *gnid_id* **processing-mode** {*enable* | *disable* }**serial-interval** {*enable*| *disable* }
	
	Example:
	`UCS(nid-group)# create name g1`
	`UCS(nid-group)# create add-nids 1-4`
	`UCS(nid-group)# create processing-mode parallel`
	`UCS(nid-group)# create processing-mode serial`
		Creates a group name and adds NIDs to the group.
		• **name—**Enter a group name.
		• **add-nids—**Enter the series of NID-IDs to be added in the group name.
		• **processing-mode—**If processing-mode is enabled, all the NIDs in the group are configured in parallel .
		• **serial-interval—**If serial-interval is enabled, all the NIDs in the group are configured ane after the other sequentially after a specific time interval.

Step 4	**create review** **Example:** `UCS(nid-group)# create review`	Displays the group name with respective NID-IDs.
Step 5	**create commit** **Example:** `UCS(nid-group)# create commit`	Sends the group name with respective NID-ID information to the UCS controller.
Step 6	**exit** **Example:** `UCS(nid-group)# exit`	Exits the NID group management mode.

Configuration Example

This example shows platform nid-controllers group-nids:

```
UCS# show platform nid-controllers group-nids

GRP_NAME        MODE           S-Interval           MEMBERS
---------     --------         ----------         -------------
   g1          Serial              0                  1,3

   g2          Serial              0                  2-3

   g4          Serial              0                  2-3
```

Configuration Example

The example shows how to create group using Bulk Provisioning Feature:

```
UCS# configure group g2
UCS(ProvisionPhyPortType)#
clearPhyStats           debug                   exit
getPhyCurrent           getPhyDefaultConf       help
history                 logout                  script
setPhyCurrent           setPhyPortDefault       showPhyPortCapabilities
showPhyPortStatistic    showPhyPortStatus       top
UCS(ProvisionPhyPortType)# setPhyCurrent physicalPortConf physicalPort 5
UCS(ProvisionPhyPortType)# setPhyCurrent physicalPortConf adminState disable
UCS(ProvisionPhyPortType)# setPhyCurrent review

Commands in queue:
        setPhyCurrent physicalPortConf physicalPort 5
        setPhyCurrent physicalPortConf adminState disable
UCS(ProvisionPhyPortType)# setPhyCurrent commit
Member NIDS: 2-3
NID:2 result is SUCCESS
        log file is at
"ucsNIDCtrlr/logs/nid-group/g2/10002_1453372217_SetPhyCurrent_nid_2.log"
NID:3 result is SUCCESS
        log file is at
"ucsNIDCtrlr/logs/nid-group/g2/10002_1453372217_SetPhyCurrent_nid_3.log"
UCS(ProvisionPhyPortType)#
```

CHAPTER 35

Template Management

Template management allows users to create and manage XML configuration templates locally on the UCS NID controller. These configuration templates can be stored on UCS controller and applied to one or more ME1200 NIDs. When used along with Bulk Provisioning feature, user can apply template XML to multiple NIDs which are part of one or more pre-configured groups. By definition, a configuration template is any technology-specific (for example, EVC, VLAN, QoS, MEP, and so on) or service-specific (ELINE, ELAN etc.) set of operations that result in provisioning of Metro Ethernet services on ME1200 NID. The benefits of using template management include the following:

- Ability to create a customized template XML file based on user configuration.
- Dedicated configuration mode for template management operations.
- Fetches existing running configuration template from operational NID, locally storing the same or customized template, and applying these to a single or group of NIDs.

Prerequisites for Configuring Template Management

- NID must be added to the controller.
- NID must be accessible from the controller.

How to Configure Template Management

Configuring Template Management

DETAILED STEPS

	Command or Action	Purpose
Step 1	**platform template** **Example:** `UCS# platform template`	Enter template management configuration mode.
Step 2	**create template** **Example:** `UCS(template)# create template <filename.xml>`	Create custom template by specifying XML file name `<filename.xml>`. This command causes a new terminal window to be opened to configure one or more technology templates.
Step 3	**Template Create operations**	The new terminal created in step 2 **Template Create** can be used to configure template operations using the various technologies. For each technology template below, you can perform the corresponding Set or Get operation. The available technologies for template management are: • **ACLDefinition**: Template for actual ACL definitions. • **EpsPortType**: Provision EPS. • **ErpsPortType**: Provision ERPS. • **FlexlinksPortType**: Provision Flexlink Backup Interface Configuration. • **LinkOamPortType**: Provision linkoam. • **LSTPortType**: Provision Link State Tracking. • **NtpPortType**: Provision NTP. • **OperationsMepPortType**: Enables fault management and performance monitoring on MEP. • **ProvisionACL**: Provision ACL Configuration. • **ProvisionEVC**: EVC operations. • **ProvisionL2CPPortType**: Provision L2CP Service on EVCs. • **ProvisionLacpPortType**: Provision LACP service. • **ProvisionLldpPortType**: Provision LLDP Service. • **ProvisionMacTableSecurityType**: Clear MAC Address Table Entries. • **ProvisionMepPortType**: Provision MEP.

	Command or Action	Purpose
		• **ProvisionPhyPortType**: Physical Port Configuration. • **ProvisionPortVlanPortType**: Provision Vlan and SwitchPort Service. • **ProvisionQos**: Configure QoS features. • **ProvisionSnmpConf**: Set snmp config. • **ProvisionStormControl**: Provision Storm Control Service. • **ProvisionStpPortType**: Provision STP Services. • **PTPPortType**: ME1200 PTP operations. • **RFC2544PortType**: RFC2544 port type. • **SFlow**: sFlow operations. • **SPAN**: SPAN Global Configuration. • **SyncE**: Set SyncE global configuration. • **sysLog**: Syslog Properties and Methods. • **UDLDPortType**: Provision UDLD Configuration. • **y1564**: Provision Y1564.
Step 4	**exit** **Example:** `UCS# exit`	After configuring the required operations, exit the template shell. A file by the name`<filename.xml>` is created in the following directory: `<user-home-dir>/ucsNIDCtrlr/config-templates/`. Check the XML data written into template file.

Step 5	**configure nid 1**
	
	Example:
	`UCS# configure nid <nid_id> or`
	`configure group <nid_group_name>`
	
	Example:
	`UCS# ConfigTemplatePortType`
	`UCS(ConfigTemplatePortType)#`
	`UCS(ConfigTemplatePortType)#`
	`applyCfgTemplate`
	`pushConfigXML`
	`templateFile file1.xml`
		Apply the template XML file created on ME1200 NID to a single NID or NID group.

Step 6	**applyCfgTemplate review**
	
	Example:
	`UCS(ConfigTemplatePortType)#`
	`applyCfgTemplate`
	`review`
		Review apply or push configuration template command to NID or NID group.

	Command or Action	Purpose
Step 7	**applyCfgTemplate commit** **Example:** `UCS(ConfigTemplatePortType)# applyCfgTemplate commit`	Commit apply or push configuration template command to NID or NID group.
Step 8	**exit** **Example:** `UCS(ConfigTemplatePortType)# exit`	Exit the ConfigTemplatePortType mode.
Step 9	**UCS(template)# fetchconfigxml** **Example:** `UCS(template)# fetchconfigxml nid 2`	Fetches the configurations from a file and saving locally.
Step 10	**fetchconfigxml commit** **Example:** `UCS(template)# fetchconfigxml commit`	Fetch file is created and saved at = `/users/<user_name>/UCS/ucsNIDCtrlr/config-templates/nid2-running-config.xml` Commits the template.
Step 11	**fetchconfigxml review** **Example:** `UCS(template)# fetchconfigxml review`	Review the command for fetching XML configuration from operational NID.
Step 12	**fetchconfigxml commit** **Example:** `UCS(template)# fetchconfigxml commit`	Commit Fetch operation
Step 13	**Exit** **Example:** `UCS(template)# exit`	Exit from template management mode.

Configuration Example

```
UCS# ProvisionEVC ?
EVC operations
UCS# ProvisionEVC
UCS(ProvisionEVC)#
addECE
addECE_v2      Add ECE [Port range extended]
addEVC
addEVC_v2      Add EVC [Port range extended]
addPolicerEVC  Add EVC policer
debug          Change to the debug mode
deleteECE
deleteEVC
deletePolicerEVC   Delete EVC policer
disableEVCpolicer
editECEConfiguration
editEVCConfiguration
```

```
editEVCConfiguration_v2 Update EVC parameters [Port range extended]
enableEVCpolicer
exit              Exit the named view
getECEBlankForm
getECEBlankForm_v2  Get ECE Blank Form [Port range extended]
getECEconfiguration
getECEconfiguration_v2 Get ECE Configuration [Port range extended]
getECECounters
getEVC_Counters     getevc Counters
getEVCBlankform     getEVCBlankform
getEVCBlankForm_v2   Get EVC Blank Form [Port range extended]
getEVCconfiguration
getEVCconfiguration_v2   Get EVC configuration [Port range extended]
help    Display an overview of the CLI syntax
history Display the current session's command line history
logout  Logout of the current CLI session
modifyEVCpolicer
reorderECEentries
script        Change to the script demo mode
top           Return to the default mode
```

Configure a new ECE ID.

```
UCS(ProvisionEVC)# addECE ece_configuration ece_id 1
AddECE Commit Success!!!
```

Validate configuration for newly configured ECE ID.

```
UCS(ProvisionEVC)# getECEconfiguration getECEconfig 1
UCS(ProvisionEVC)# getECEconfiguration review

Commands in queue:
        getECEconfiguration getECEconfig 1
UCS(ProvisionEVC)# getECEconfiguration commit
Connection to web service opened
Decoding was successful
Decoded record:
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.ece_id = 1
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.uni_ports.
GigabitEthernet_1_UNI = false
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.uni_ports.
GigabitEthernet_2_UNI = false
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.uni_ports.
GigabitEthernet_3_UNI = false
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.uni_ports.
GigabitEthernet_4_UNI = false
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.uni_ports.
GigabitEthernet_5_UNI = false
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.uni_ports.
GigabitEthernet_6_UNI = false
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.
outer_tag_match.match_type.t = 1
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.
outer_tag_match.match_type.u.any = 'default'
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.
outer_tag_match.match_fields.vlan_id_filter.t = 2
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.
outer_tag_match.match_fields.vlan_id_filter.u.range = '0-4095'
GetECEconfiguration_Output.ece_config_get_response.ece_configuration.control.ingress_match.
outer_tag_match.match_fields.inner_pcp.t = 15
…
UCS(ProvisionEVC)#exit
UCS#
```