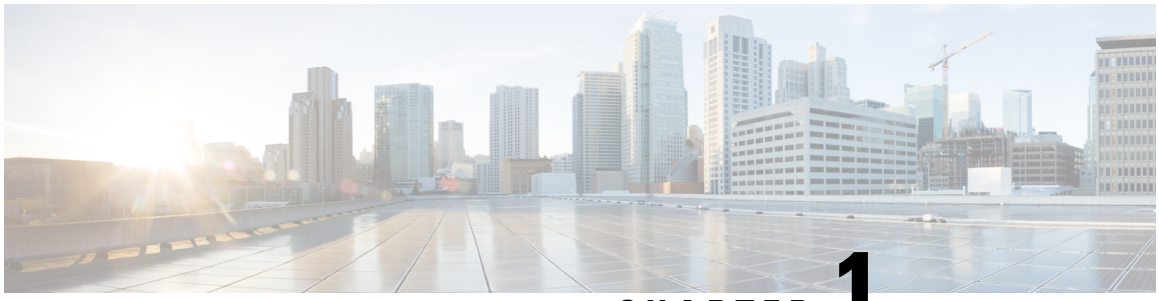
Layer 2 Configuration Guide, Cisco IOS XE 17 (Cisco NCS 4200 Series)

First Published: 2020-07-30

Last Modified: 2023-07-31

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



CHAPTER 1

Feature History

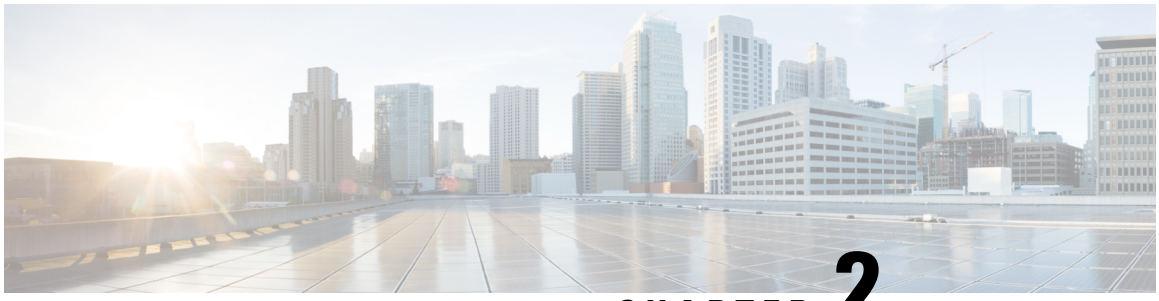
The following table lists the new features supported in the Layer 2 Configuration Guide in Cisco IOS XE 16 releases, on Cisco NCS 4206 and Cisco NCS 4216 routers.

Feature	Description
Cisco IOS XE Cupertino 17.8.1	
802.1AE WAN MACsec Enhancement for 1GE and 10GE NCS4200-1T16G-PS	The 802.1AE WAN MACsec supports 10GE physical layer (PHY) interfaces for NCS4200-1T16G-PS interface module. From this release, full HA, Power on Self Test (POST) and double tag support are available on NCS4200-1T16G-PS interface module
Cisco IOS XE Bengaluru 17.6.1	
802.1AE WAN MACsec for 1GE and 10GE NCS4200-1T16G-PS	The WAN MACsec and MKA feature introduce MACsec support on WAN and uplink support and pre-shared key support for the MACsec Key Agreement protocol (MKA). The WAN MACsec supports 1GE and 10GE interfaces for NCS4200-1T16G-PS interface module.
Cisco IOS XE Bengaluru 17.5.1	
MAC Security	The MACsec and Macsec Key Agreement protocol (MKA) features are introduced on the main interface with pre-shared key support for the MKA. This feature is supported on the Cisco RSP3 module.
Cisco IOS XE Amsterdam 17.3.1	
RSPAN over VPLS Pseudowire Network	This feature allows the traffic mirroring destination port to be configured as a pseudowire rather than a physical port. This feature lets the designated traffic on the source port to be mirrored over the pseudowire to a remote location. This feature is supported on the Cisco RSP3 module.

The following table lists the new features supported in the Layer 2 Configuration Guide in Cisco IOS XE 17 releases, on Cisco NCS 4201 and Cisco NCS 4202 routers.

Feature	Description
Cisco IOS XE Cupertino 17.8.1	

Feature	Description
Support for Ethernet Data Plane Loopback on Bundle Interface	This feature enables ethernet data plane loopback on bundle interfaces. You can also configure the feature when the router is not physically connected and the port is in down state. This feature is only applicable on internal or terminal loopback in up or down state.
Cisco IOS XE Bengaluru 17.6.1	
MAC Security	The MACsec and Macsec Key Agreement protocol (MKA) features are introduced on the main interface with pre-shared key support for the MKA. This feature is supported on the Cisco RSP3 module.
Cisco IOS XE Bengaluru 17.4.1	
Enhanced Ethernet Data Plane Loopback	The Ethernet data plane loopback feature is enhanced to avoid control packets getting dropped. The enhancement supports internal shaper configuration, when terminal ELB session is activated or deactivated to rate the limit the ELB session traffic.



CHAPTER 2

Configuring Ethernet Dataplane Loopback

Ethernet data plane loopback provides a means for remotely testing the throughput of an Ethernet port.

Prerequisites for Ethernet Data Plane Loopback

- Ethernet loopback sessions are supported only of EFPs (service instances, Ethernet flow points, EVCs).
- Dot1q tags must be configured while configuring Ethernet loopback sessions on EFPs. However, loopback sessions can be configured using dot1q/QinQ, even if the underlying EFP has the dot1q/QinQ range configured.
- Internal loopback sessions configured must be within the 1 GB reserved bandwidth.
- Internal loopback can be launched even when the physical interface port state is down.

Restrictions for Ethernet Data Plane Loopback on Physical Interfaces

- Data plane loopback on routed port infrastructure is *not* supported.
- Etype, src-mac, and llc-oui based loopback traffic filtering is *not* supported.
- Port-level QoS is *not* bypassed.
- Port shaper cannot be bypassed in facility loopback.

- Facility and terminal Ethernet data plane loopback (ELB) are *not* supported on dot1ad nni interface.
- Internal loopback sessions configured must be within the 1 GB reserved bandwidth for Cisco ASR 900 Series RSP2 Module.
- A maximum number of 20 facility loopback sessions can be created per system, provided 16 sessions are with Dot1Q and 4 sessions are with Dot1Q and destination MAC address. This scale reduces if SPAN or RSPAN is configured. This scale is supported on the Cisco ASR 900 Series RSP2 module.
- A maximum number of 12 terminal loopback sessions can be created per system, provided 8 sessions are with Dot1Q and 4 sessions are with Dot1Q and destination MAC address. This scale reduces if RSPAN or SADT is configured. This scale is supported on the Cisco ASR 900 Series RSP2 module.
- Only one Ethernet loopback (terminal or facility) session can be active on an EFP at any instance.
- Local SPAN and ELB cannot be enabled on a physical interface at the same time.
- Loopback sessions cannot be initiated on a port configured with SPAN or RSPAN.
- Ethernet loopback is not supported on a range of dot1q tags.
- Ethernet Data Plane Loopback is affected on STP enabled interface.
- Dynamic addition of rewrite ingress tags with default EFP is not supported.
- Dynamic changes at EFP and interface level are not supported when Ethernet Data Plane Loopback is active.
- Egress EFP is not updated for external Ethernet data plane loopback statistics.
- For internal Ethernet data plane loopback ingress and egress interface statistics are not updated on interface, where internal ELB is enabled.
- If traffic is more than 650Mbps and if the packet size is less than a frame size of 64, then BFD and OSPF flaps are expected.
- Starting from Cisco IOS XE Cupertino Release 17.7.1, SADT is *not* supported over EFP with untagged VLAN.

RSP3 Module

- Starting from Cisco IOS XE Amsterdam 17.1.x release, the template sr_5_label_push_enable is not supported with Ethernet loopback.
- Etype, VLAN, COS, src-mac, and llc-oui based loopback traffic filtering is *not* supported.
- Port-based ELB is *not* supported. Ethernet loopback sessions are supported only on EFPs and Trunk EFPs on the RSP3 module.
- Internal ELB is *not* supported when the physical interface port state is down.
- Data filtering of loopback is *not* enforced for the traffic coming in the opposite direction.
- Filtering based on specific VLAN is *not* supported. ELB is applicable to all the VLANs configured in EFP on the RSP3 module.
- Random unknown IP packets may be looped back with MAC address swap, but RSP3 module does *not* support IP address swap. If the packet has a destination MAC address and a destination IP address similar

to the BDI MAC and IP address, then the packet is punted and is *not* looped back. The L3 packets that must be routed is also *not* looped back. Hence, dscp/prec marking with ELB is *not* supported.

- Traffic to TEFP, which is not part of BDI or CFM, is looped back based on filters. But traffic to TEFP with VLAN which is part of BDI or CFM, is *not* looped back on the RSP3 module. This is applicable for both types of Ethernet data plane loopback.
- All packets with broadcast or multicast destination MAC is *not* qualified for ELB, hence such packets will *not* be looped back on the RSP3 module.
- Three-level HQOS shaper/policer *not* supported with ELB.
- Dot1Q filter is *not* supported.
- Internal loopback sessions configured must be within the 100 GB reserved recycle bandwidth.
- MAC-ACL *cannot* be bypassed in with facility loopback.
- A maximum number of 20 facility loopback and 12 terminal loopback sessions are supported.

Information on Ethernet Data Plane Loopback

The Ethernet data plane loopback feature provides a means for remotely testing the throughput of an Ethernet port. You can verify the maximum rate of frame transmission with no frame loss. This feature allows for bidirectional or unidirectional throughput measurement, and on-demand/out-of-service (intrusive) operation during service turn-up. This feature supports two types of Ethernet loopback. RSP3 supports the following types of loopback from Cisco IOS XE Everest 16.5.1 release.

- Facility loopback (external)—Traffic loopback occurs at the Ingress interface. Traffic does not flow into the router for loopback.
- Terminal loopback (internal)—Traffic loopback occurs at the Egress interface. Traffic loopback occurs after the traffic flows into the router to the other interface.

Prior to Cisco IOS XE Cupertino Release 17.8.1, this feature was only supported on the physical interfaces.

Starting with Cisco IOS XE Cupertino Release 17.8.1, this feature is also supported on the bundle interfaces.

QoS Support for Ethernet Data Plane Loopback

- Ingress QoS is bypassed in external loopback on service instances.
- Internal loopback sequence is as follows:
  - Ingress QoS
  - Egress QoS (egress port) (both, shaper and policer are supported).
  - Ingress QoS on ingress port and egress QoS on egress port (both, shaper and policer are supported) on the RSP3 module.
  - Ingress QoS on egress port and egress QoS on ingress port on the RSP3 module.
- All port-level and EFP-level QoS is applicable for internal Ethernet data plane loopback.

- For external Ethernet data plane loopback:
  - All port-level and EFP-level QoS is bypassed except for shaper.
  - Port-level shaper cannot be bypassed.

How to Configure Ethernet Data Plane Loopback on Physical Interfaces

Enabling Ethernet Data Plane Loopback on Physical Interfaces

Table 1: Feature History

Feature Name	Release Information	Feature Description
EDPL support on dot1ad	Cisco IOS XE Cupertino 17.8.1	This features enables configuration of Ethernet Data Plane Loopback on interfaces configured with 802.1ad encapsulation. This helps measure the interface throughput handing encapsualted traffic.

```
enable
configure terminal
interface gigabitethernet 0/2/1
service instance 1 ethernet
encapsulation dot1ad 101 dot1q 100
bridge-domain 120
ethernet loopback permit external
end
```

Note ELB is supported using a MAC filter for UP-MEP session. If you are starting ELB without the MAC filter, the UP-MEP session will go DOWN.

Starting an Ethernet Data Plane Loopback Session on Physical Interfaces

Note To start a loopback for untagged and default EFPs, dot1q and second-dot1q are not needed. Dot1q is *not* applicable to start a loopback session on the RSP3 module.

Note By default the session would be running for 300 seconds unless you explicitly specify and automatically stops after the session time expiry.

```
enable
configure terminal
ethernet loopback start local interface gigabitEthernet 0/4/1 service instance 10 external
 dot1q 10 cos 1 destination mac-address 0000.0000.0001 timeout none
end
This is an intrusive loopback and the packets matched with the service will not be able
to pass through.
Continue? (yes/[no]): yes
```

Dot1q and COS-based filtering is not supported on the RSP3 module.

```
enable
configure terminal
ethernet loopback start local interface gigabitEthernet 0/4/1 service instance 10 external
 destination mac-address 0000.0000.0001 timeout none
end
```

Stopping an Active Session on Physical Interfaces

Use the **ethernet loopback stop** command to stop an active session on an interface or to stop all sessions based on the session id.

```
Router# ethernet loopback stop local interface gigabitEthernet 0/4/1 id 1
```

Configuration Examples

Example: Configuring External Loopback on Physical Interfaces

This example shows how to configure external (facility) loopback.

```
Router(config)# interface gigabitEthernet 0/4/1
Router(config-if)# service instance 1 ethernet
Router(config-if-srv)# encapsulation dot1ad 100 dot1q 120
Router(config-if-srv)# bridge-domain 120
Router(config-if-srv)# ethernet loopback permit external
```

This example shows external (facility) loopback on the Gigabit Ethernet 0/4/1 interface:

```
interface GigabitEthernet0/4/1
 no ip address
 negotiation auto
 service instance 10 ethernet
  encapsulation dot1q 10
  rewrite ingress tag pop 1 symmetric
  bridge-domain 10
  ethernet loopback permit external ===? For facility loopback
 !
end
```

This example below shows how to start external (facility) loopback on the router. A warning message is displayed. Type **yes** to continue.

```
Router# ethernet loopback start local interface gigabitEthernet 0/4/1 service instance 10
external dot1q 10 cos 1
 destination mac-address 0000.0000.0001 timeout none
```

```
This is an intrusive loopback and the packets matched with the service will not  be able
to pass through.
Continue? (yes/[no]): yes
```

Note Dot1q and COS-based filtering is not supported on the RSP3 module.

Example: Configuring Terminal Loopback on Physical Interfaces

This example shows how to configure internal (terminal) loopback.

```
Router(config)# interface gigabitEthernet 0/0/0
Router(config-if)# service instance 1 ethernet
Router(config-if-srv)# encapsulation dot1q 120
Router(config-if-srv)# bridge-domain 120
Router(config-if-srv)# ethernet loopback permit internal
```

This example shows internal (terminal) loopback on Gigabit Ethernet 0/0/0 interface:

```
interface TenGigabitEthernet0/0/0
 no ip address
service instance 10 ethernet
  encapsulation dot1q 10
  rewrite ingress tag pop 1 symmetric
  bridge-domain 10
  ethernet loopback permit internal
 !
end
```

Verifying Ethernet Data Plane Loopback

Example: Verifying Ethernet Dataplane Loopback on Physical Interfaces

Use the **show ethernet loopback** {**active** | **permitted**} [**interface** *interface numbe*r] command.

- The following example displays the loopback capabilities per interface. The output shows internal (terminal) loopback has been permitted on Ten Gigabit Ethernet 0/0/0 interface and external (facility) loopback has been permitted on Gigabit Ethernet 0/4/1 interface.

```
Router# show ethernet loopback permitted

-------------------------------------------------------------------------------
Interface                               SrvcInst Direction
Dot1q/Dot1ad(s)                         Second-Dot1q(s)
-------------------------------------------------------------------------------
Te0/0/0                                 10                              Internal
10
Gi0/4/1                                 10                              External
10
```

- This example shows all active sessions on the router.

```
Router# show ethernet loopback active

============================================================
Loopback Session ID        : 1
```

```
Interface                 : GigabitEthernet0/4/1
Service Instance          :10
Direction                 : External
Time out(sec)             : none
Status                    : on
Start time                : 10:31:09.539 IST Mon Aug 26 2013
Time left                 : N/A

Second-dot1q(s)           :
Source Mac Address        : Any
Destination Mac Address   : 0000.0000.0001
Ether Type                : Any
Class of service          : 1
Llc-oui                   : Any


Total Active Session(s)   : 1
Total Internal Session(s) : 0
Total External Session(s) : 1
```

- This example shows how to stop the sessions on the router.

```
Router# ethernet loopback stop local interface GigabitEthernet
0/4/1 id 1
```

Information on Enhanced Ethernet Data Plane Loopback

Table 2: Feature History

Feature Name	Release	Description
Enhanced Ethernet Data Plane Loopback	Cisco IOS XE Bengaluru 17.4.1	The Ethernet data plane loopback feature is enhanced to avoid control packets getting dropped. The enhancement supports internal shaper configuration, when terminal ELB session is activated or deactivated to rate the limit the ELB session traffic. The enhancement is applicable only on internal loopback.

The Ethernet data plane loopback feature is enhanced to avoid control packets getting dropped, besides the ELB traffic drop. If terminal ELB is configured on 1 GB interface, then other priority traffic like BFD is dropped due to congestion.

The enhancement supports internal shaper configuration, when terminal ELB session is activated or deactivated to limit the rate for ELB session traffic.

Restrictions

- After starting EDPL, even if you remove ELB_SHAPE from EFP, then EDPL continues to run.
- Do not modify the ELB_SHAPE directly.
    - Use the platform command to make any changes in ELB_SHAPE PM.

- If you directly modify the ELB_SHAPE PM, then the shaper is applied based on the updated values that are defined under ELB_SHAPE.

- If you remove ELB_SHAPE PM from the global config accidentally, then you must remove the **platform edpl_internal_shaper xxx** platform command and reapply.
- For any change to the EDPL parameters, we recommended you to perform the following:
  1. Deactivate or Stop the EDPL session.
  2. Do the required configuration changes.
  3. Activate or Start the EDPL session.

  Do not dynamically change or delete the EDPL shaper or the interface bandwidth.
- In an xconnect scenario, when the interface goes down, the xconnect goes down, but EDPL remains active. Hence, you must stop the EDPL session and start back the EDPL session, after the xconnect is UP.

Configuring Ethernet Data Plane Loopback Session

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **configure EDPL shaper**

Example:

```
Device# platform edpl_internal_shaper  600000000
```

Enables EDPL shaper.

Step 4 **interface GigabitEthernet**

Example:

```
Device(config)# interface GigabitEthernet 0/0/9
```

```
Device(config-if)# service instance 1401 ethernet
Device(config-if-srv)# encapsulation dot1q 1401
Device(config-if-srv)# rewrite ingress tag pop 1 symmetric
Device(config-if-srv)# ethernet loopback permit internal
Device(config-if-srv)# end
```

Configures internal loopback on the interface.

Step 5 **ethernet loopback start local**

Example:

```
Device# ethernet loopback start local interface GigabitEthernet 0/0/9 service instance 1401
 internal dot1q 1401 timeout none
```

Starts EDPL loopback session.

Example: Configuring Ethernet Loopback Active

The following example shows the activated and running EDPL session:

```
Router# show ethernet loopback active

Load for five secs: 5%/1%; one minute: 6%; five minutes: 6%
Time source is NTP, 18:18:23.680 IST Wed Aug 26 2020

============================================================
Loopback Session ID     : 1
Interface               : GigabitEthernet0/0/9
Service Instance        : 1401
Direction               : Internal
Time out(sec)           : none
Status                  : on
Start time              : 18:17:58.360 IST Wed Aug 26 2020
Time left               : N/A
Dot1q/Dot1ad(s)         : 1401
Second-dot1q(s)         :
Source Mac Address      : Any
Destination Mac Address : Any
Ether Type              : Any
Class of service        : Any
Llc-oui                 : Any

Total Active Session(s): 1
Total Internal Session(s): 1
Total External Session(s): 0
```

To deactivate an EDPL session.

```
Router# ethernet loopback stop local interface gigabitEthernet 0/0/9 id 1

003846: Aug 26 18:40:56.528 IST: %E_DLB-6-DATAPLANE_LOOPBACK_STOP: Ethernet Dataplane
Loopback Stop on interface GigabitEthernet0/0/9 service instance 1401 with session id 1
```

Example: Configuring Ethernet Data Plane Loopback Session

The following example shows the global shaper configuration:

Note If the interface bandwidth is less than the shaper value, then EDPL does not get started. You can refer to the syslogs.

```
Interface bandwidth 400Mb, shaper value 600Mb

platform edpl_internal_shaper  600000000

interface GigabitEthernet0/0/9
 bandwidth 400000

service instance 1401 ethernet
  encapsulation dot1q 1401
  rewrite ingress tag pop 1 symmetric
  xconnect 192.211.92.3 1401 encapsulation mpls
  ethernet loopback permit internal
 !

Router# ethernet loopback start local interface gigabitEthernet 0/0/9 service instance 1401
 internal dot1q 1401 timeout none

This is an intrusive loopback and the packets matched with the service will not be able to
 pass through. Continue? (yes/[no]): yes
QoS Configuration failed !!! Shaper value is greater than interface speed

 Failed to config EDPL shaper, EDPL not activated !!!
```

Use Cases or Deployment Scenarios

ELB is Supported with MAC Filter for UP-MEP Session

In the following scenario, ELB is supported using a MAC filter for UP-MEP session. If you starting ELB with out MAC filter, the UP-MEP session will go DOWN.

```
enable
configure terminal
service instance 800 ethernet 800
encapsulation dot1q 800
service-policy input <NAME>
xconnect 10.0.0.2 880 encapsulation mpls
cfm mep domain <NAME> mpid 200
cos 7
ethernet loopback permit external
ethernet loopback permit internal

Router#ethernet loopback start local interface gi0/0/0 service instance 800 internal dot1q
 800 destination mac-address f078.1685.313f timeout none

This is an intrusive loopback and the packets matched with the service will not  be able
to pass through. Continue? (yes/[no]): yes


Router#show ethernet cfm maintenance-points remote
-------------------------------------------------------------------------------
MPID  Domain Name                                MacAddress          IfSt  PtSt
 Lvl  Domain ID                                  Ingress
```

```
 RDI  MA Name                                        Type Id             SrvcInst
      EVC Name                                                           Age
      Local MEP Info
------------------------------------------------------------------------------
220   CCI                                           f078.1685.313f      Up    Up
 0    CCI                                           Gi0/0/0:(10.0.0.2, 880)
 -    800                                           XCON N/A            800
      800                                                               0s
      MPID: 200 Domain: CCI MA: 800

Total Remote MEPs: 1
```

Support for Ethernet Data Plane Loopback on Bundle Interface

Table 3: Feature History

Feature Name	Release Information	Feature Description
Support for Ethernet Data Plane Loopback on Bundle Interface	Cisco IOS XE Cupertino 17.8.1	This feature enables ethernet data plane loopback on bundle interfaces. You can also configure the feature when the router is not physically connected and the port is in down state. This feature is only applicable on internal or terminal loopback in up or down state.

Bundle interface or a link bundle is a group of one or more ports that are aggregated together and treated as a single link. This allows you to group multiple point-to-point links together into one logical link and provide higher bidirectional bandwidth, redundancy, and load balancing between two routers. A virtual interface is assigned to the bundled link. The component links can be dynamically added and deleted from the virtual interface. The virtual interface is treated as a single interface on which you can configure an IP address and other software features used by the link bundle. Packets sent to the link bundle are forwarded to one of the links in the bundle.

Bundle interfaces increase bandwidth availability, because traffic is forwarded over all available members of the bundle. Therefore, traffic can if one of the links within a bundle fails. can without interrupting packet flow. The ethernet dataplane loopback feature configured on bundle interfaces provides a methodology to verify the maximum rate of frame transmission with no frame loss.

Prior to Cisco IOS XE Cupertino 17.8.1, you could only configure ethernet data plane loopback on the physical interfaces.

Starting with Cisco IOS XE Cupertino 17.8.1, you can also configure ethernet data plane loopback feature on the bundle interfaces. But, you can only configure internal or terminal loopback in up or down state.

This feature is only supported on Cisco RSP2 module.

Scenario: Support for Ethernet Dataplane Loopback on Link Down Port

Consider a scenario when you need to configure ethernet dataplane loopback feature before the router is physically connected. Thus, you need to configure the feature when the port link is down. Starting with Cisco IOS XE Cupertino 17.8.1, you can configure the terminal or internal ethernet dataplane loopback feature even

when the router is not physically connected and the link is down. But, as this feature is not supported on external or facility loopback, you cannot configure external loopback feature when the port link is down.

A port channel bundles individual interfaces into a group to provide increased bandwidth and redundancy. Previously, the internal ethernet dataplane loopback was not supported on the port channel interface. Starting with Cisco IOS XE Cupertino 17.8.1, you can now configure internal loopback on the port channel interface even when the interface is down.

Restrictions for Ethernet Dataplane Loopback on Bundle Interface

- The internal ethernet dataplane loopback feature is only available with service instance for port channel interface. It is not available when you configure MPLS or IP or Layer 3 on port channel interface.
- At least one member link must be added to the port channel interface for ethernet dataplane loopback.
- External ethernet loopback session on port channel interface is *not* supported.
- This feature will only function for traffic flow on first member of the port channel.
- You cannot configure the feature when the bundle members are in suspended state.
- The maximum traffic performance of terminal loopback is 1GBPS.

Configure Ethernet Dataplane Loopback Start Session on Bundle Interface

1. **Activate Loopback on the EFP**

   To activate terminal loopback on the EFP:

```
interface Po1
 no ip address
service instance 10 ethernet
  encapsulation dot1q 10
  rewrite ingress tag pop 1 symmetric
  bridge-domain 10
  ethernet loopback permit internal  === For Terminal Loopback
 !
```

2. **Start Loopback Session**

   To start a terminal loopback session on bundle interface:

```
R11#ethernet loopback start local in po1 ser ins 2 inte dot1q 2 destination mac-address
 3333.0001.0003 tim non

This is an intrusive loopback and the packets matched with the service will not  be able
 to pass through. Continue? (yes/[no]): yes
```

Configure Ethernet Dataplane Loopback Stop Session on Bundle Interface

1. **Stop Loopback Session**

   To stop a terminal loopback session on a bundle interface:

```
R2#ethernet loopback stop local interface po1 id 1
```

2. **Deactivate Loopback Session on the EFP**

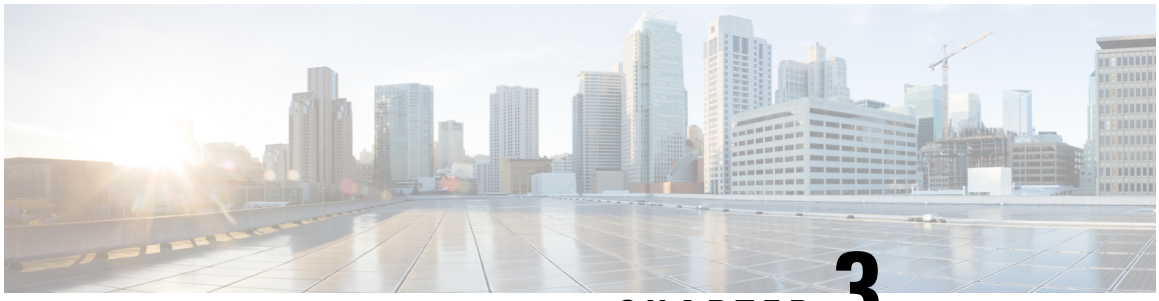
To stop the terminal loopback session in the EFP:

```
interface Po1
 no ip address
 service instance 10 ethernet
  encapsulation dot1q 10
  rewrite ingress tag pop 1 symmetric
  bridge-domain 10
  no ethernet loopback permit internal
```

Verification of Ethernet Dataplane Loopback Configuration on Bundle Interface

Use the **show ethernet loopback active** command to display all active sessions on the router.

```
R11#show ethernet loopback active
============================================================
Loopback Session ID     : 1
Interface               : Port-channel1
Service Instance        : 2
Direction               : Internal
Time out(sec)           : none
Status                  : on
Start time              : 10:35:16.940 IST Fri Dec 17 2021
Time left               : N/A
Dot1q/Dot1ad(s)         : 2
Second-dot1q(s)         :
Source Mac Address      : Any
Destination Mac Address : 3333.0001.0003
Ether Type              : Any
Class of service        : Any
Llc-oui                 : Any


Total Active Session(s): 1
Total Internal Session(s): 1
Total External Session(s): 0
```



CHAPTER 3

Configuring Switched Port Analyzer

A local Switched Port Analyzer (SPAN) session is an association of a destination interface with a set of source interfaces. Local SPAN sessions allow you to monitor traffic on one or more interfaces and to send either ingress traffic, egress traffic, or both to one destination interface.

RSPAN allows remote monitoring of traffic where the source and destination switches are connected by L2VPN networks. The RSPAN source is either ports or VLANs as in a traditional RSPAN. However, the SPAN source and destination devices are connected through an L2 pseudowire associated with the RSPAN VLAN over an MPLS/IP network.

This document describes how to configure local Switched Port Analyzer (SPAN), remote SPAN (RSPAN) and RSPAN over VPLS Network on the Cisco router.

Local SPAN Session

A local Switched Port Analyzer (SPAN) session is an association of a destination interface with a set of source interfaces. You configure local SPAN sessions using parameters that specify the type of network traffic to monitor. Local SPAN sessions allow you to monitor traffic on one or more interfaces and to send either ingress traffic, egress traffic, or both to one destination interface.

Local SPAN sessions do not interfere with the normal operation of the switch. You can enable or disable SPAN sessions with command-line interface (CLI) commands. When enabled, a local SPAN session might become active or inactive based on various events or actions, and this would be indicated by a syslog message. The **show monitor session span** *session number* command displays the operational status of a SPAN session.

A local SPAN session remains inactive after system power-up until the destination interface is operational.

The following configuration guidelines apply when configuring local SPAN:

- When enabled, local SPAN uses any previously entered configuration.
- Use the **no monitor session** *session number* command with no other parameters to clear the local SPAN session number.

RSPAN Session

An RSPAN source session is an association of source ports or VLAN across your network with an RSPAN Vlan. The RSPAN VLAN/BD on the router is the destination RSPAN session.

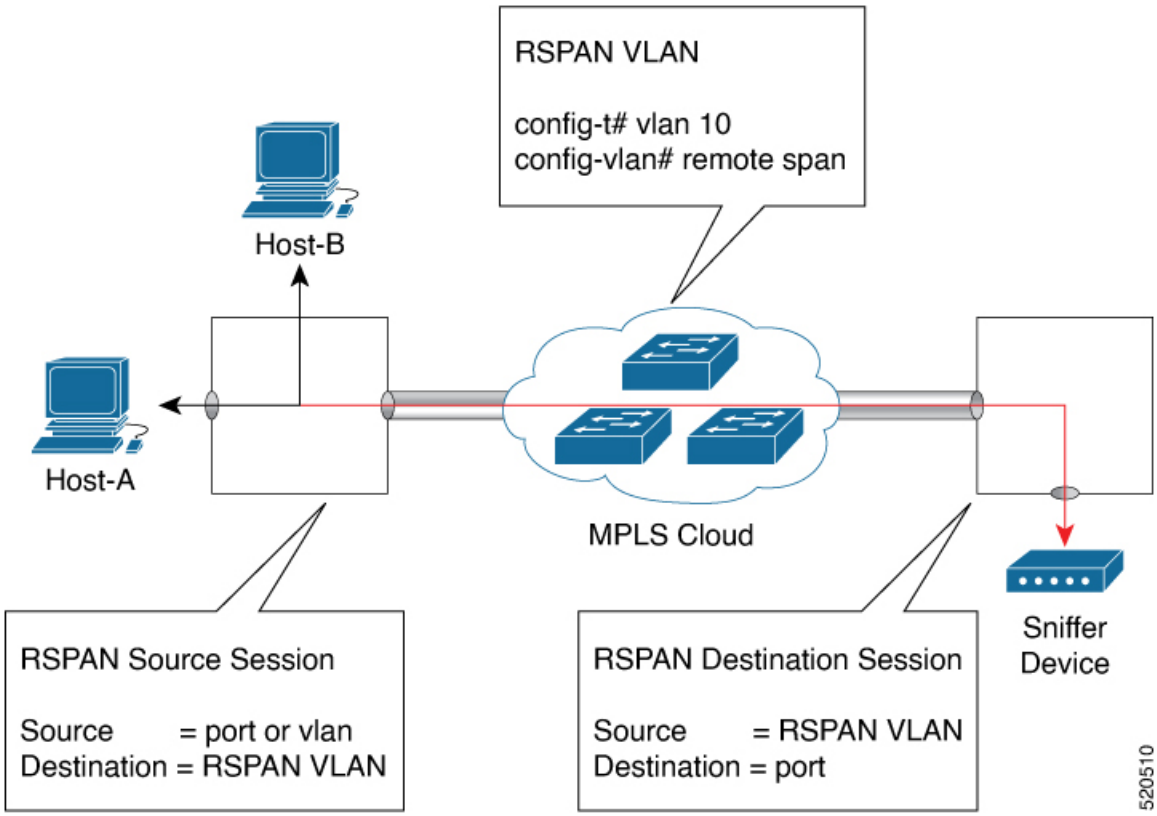
RSPAN over VPLS Network for RSP3 Module

Table 4: Feature History

Feature Name	Release	Description
RSPAN over VPLS Pseudowire Network	Cisco IOS XE Amsterdam 17.3.1	This feature allows the traffic mirroring destination port to be configured as a pseudowire rather than a physical port. This feature lets the designated traffic on the source port to be mirrored over the pseudowire to a remote location. This feature is supported on the Cisco RSP3 module.

RSPAN allows remote traffic monitoring, where the source and destination routers are connected by VPLS pseudowire network. The SPAN Source and Destination routers are connected through a VPLS Pseudowire connected with the RSPAN VLAN over an MPLS or IP network. The VPLS pseudowire is dedicated only to the RSPAN traffic. All the mirrored traffic from the source port is carried over the VPLS Pseudowire connected with the RSPAN VLAN towards the destination port. On the destination router, a port belonging to the RSPAN VLAN or EVC BD is connected to the sniffer device.

Figure 1: RSPAN Traffic over VPLS Network on the Cisco RSP3 module



Prerequisites for Configuring Local SPAN and RSPAN

Local SPAN

- Use a network analyzer to monitor interfaces.

RSPAN

- Before configuring RSPAN sessions, you must first configure:
  1. Source interface
  2. Destination Bridge Domain over VPLS

Restrictions for Local Span and RSPAN

Local Span

- Local SPAN is only supported on physical ports.

- SPAN monitoring of port-channel interfaces or port-channel member-links is *not* supported.
- Combined Egress local SPAN bandwidth supported on Cisco ASR 900 Series RSP2 module is 1 GB.
- Local SPAN is not supported on logical interfaces such as Vlans or EFPs.
- Up to 14 active local SPAN sessions (ingress and egress) are supported. The router supports up to 14 ingress sessions and up to 12 egress sessions.
- Only one local SPAN destination interface is supported. You *cannot* configure a local SPAN destination interface to receive ingress traffic.
- Outgoing Cisco Discovery Protocol (CDP), Bridge Protocol Data Unit (BPDU), IS-IS, and OSPF packets are not replicated.
- When enabled, local SPAN uses any previously entered configuration.
- When you specify source interfaces and do not specify a traffic direction (**Tx**, **Rx**, or **both**), **both** is used by default.
- The SPAN port does not work for Rx traffic on the pseudowire for interfaces, when the SPAN port is in different ASIC of the RSP2 module.
- Local SPAN destinations never participate in any spanning tree instance. Local SPAN includes BPDUs in the monitored traffic, so any BPDUs seen on the local SPAN destination are from the local SPAN source.
- Local SPAN sessions with overlapping sets of local SPAN source interfaces or VLANs are *not* supported.
- Configuring SPAN and netflow on the same interface is not supported. If SPAN and netflow have been mistakenly configured on the same interface, reset the interface. Use the **default interface** command to set the interface back to its default values, and then configure SPAN.

  The following code shows how to reset the interface:

```
router(config)#default interface GigabitEthernet0/0/0
router(config)#interface GigabitEthernet0/0/0
router(config)#ip address 192.168.16.1 255.255.255.0
router(config)#negotiation auto
router(config)#cdp enable
```

  For the SPAN configuration, see Configuring Sources and Destinations for Local SPAN, on page 26.

RSP3 module

- Destination port of SPAN session, *cannot* be used for other network data traffic flow.
- Multiple destinations for same SPAN session is *not* supported on the Cisco ASR 900 Series RSP3 module.
- Jumbo sized packets and bad CRC packets are *not* spanned.
- Combined Egress local SPAN bandwidth supported is about 100GB depending on other traffic on the internal recycle interface.
- Port-channel *cannot* be used as the SPAN destination.

RSPAN

- RSPAN Vlan/BD is *not* used for data traffic.

- The maximum number of supported RSPAN sessions are 14.
- Only one source port is supported per RSPAN.
- Source ranges (vlan range or port range) is *not* supported.
- Vlan filtering is not supported.
- If two RSPAN configurations sessions are configured on two RSPAN BDs associated to the same Trunk EFP, the traffic from the first session flows to the second session after it is configured.
- RSPAN destination configuration for Layer2 pseudowire is *not* supported.
- If RSPAN BD is associated with a VPLS pseudowire, the traffic flows through the VPLS pseudowire.
- If RSPAN source and destination are separated by pseudowire, then the RSPAN VLAN details must be updated to both RSPAN source switch and destination switch. The pseudowire should also be dedicated for RSPAN traffic.
- BDI should not be created when that BD is part of RSPAN.
- Monitor session should be created only after RSPAN BD is created.
- Do not have RSPAN bridge domain as part of RSPAN source interface.
- Source and destination ports for a Tx SPAN or RSPAN session should be in the same ASIC. This is applicable to Cisco RSP2 module.

RSP3 module

- RSPAN is *not* supported on the Cisco ASR 900 Series RSP3 module.

Understanding Local SPAN and RSPAN

Local SPAN Traffic

Network traffic, including multicast, can be monitored using SPAN. Multicast packet monitoring is enabled by default. In some SPAN configurations, multiple copies of the same source packet are sent to the SPAN destination interface. For example, a bidirectional (both ingress and egress) SPAN session is configured for sources a1 and a2 to a destination interface d1. If a packet enters the switch through a1 and gets switched to a2, both incoming and outgoing packets are sent to destination interface d1; both packets would be the same (unless a Layer-3 rewrite had occurred, in which case the packets would be different).

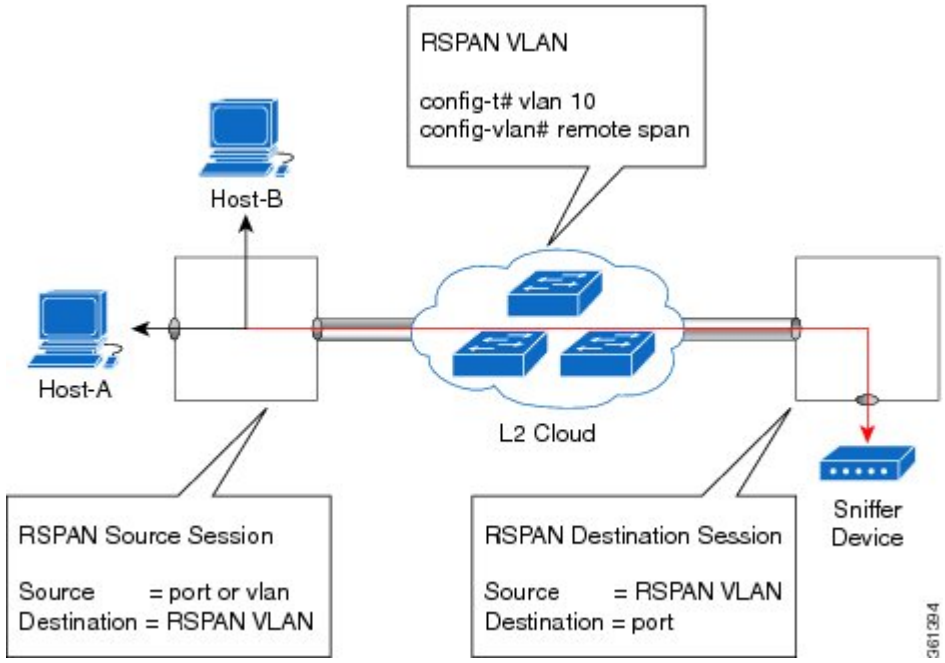
RSPAN Traffic for RSP2 Module

RSPAN supports source ports and source VLANs in the source switch and destination as RSPAN VLAN/BD.

The figure below shows the original traffic from the Host A to Host B via the source ports or VLANs on Host A. The source ports or VLANs of Host A is mirrored to Host B using RSPAN VLAN 10. The traffic for each RSPAN session is carried over a user-specified RSPAN VLAN that is dedicated for that RSPAN session in all participating devices. The traffic from the source ports or VLANs are mirrored into the RSPAN VLAN and forwarded over Trunk or the EVC bridge domain (BD) ports carrying the RSPAN VLAN to a destination session monitoring the RSPAN VLAN.

Each RSPAN source must have either ports or VLANs as RSPAN sources. On RSPAN destination, the RSPAN VLAN is monitored and mirrored to the destination physical port connected to the sniffer device.

Figure 2: RSPAN Traffic



RSPAN allows remote monitoring of traffic where the source and destination switches are connected by L2VPN networks

The RSPAN source is either ports or VLANs as in a traditional RSPAN. However, the SPAN source and destination devices are connected through a L2 pseudowire associated with the RSPAN VLAN over an MPLS/IP network. The L2 pseudowire is dedicated for only RSPAN traffic. The mirrored traffic from the source port or VLAN is carried over the pseudowire associated with the RSPAN VLAN towards the destination side. On the destination side, a port belonging to the RSPAN VLAN or EVC BD is connected to sniffer device.

Destination Interface

A destination interface, also called a monitor interface, is a switched interface to which SPAN or RSPAN sends packets for analysis. You can have only one destination interface for SPAN sessions.

An interface configured as a destination interface cannot be configured as a source interface. Specifying a trunk interface as a SPAN or RSPAN destination interface stops trunking on the interface.

Source Interface

A source interface is an interface monitored for network traffic analysis. An interface configured as a destination interface cannot be configured as a source interface.

Traffic Directions

Ingress SPAN (Rx) copies network traffic received by the source interfaces for analysis at the destination interface. Egress SPAN (Tx) copies network traffic transmitted from the source interfaces to the destination interface. Specifying the configuration option (both) copies network traffic received and transmitted by the source interfaces to the destination interface.

The following table lists the supported traffic types for RSPAN.

Table 5: RSPAN over VPLS Traffic for RSP3 module

Source	Ingress Mirror (Rx)	Egress Mirror (Tx)	Both
CFM	Not Supported	Supported	Not Supported
Layer 2	Supported	Supported	Supported
Layer 3	Incoming Ethernet and VLAN header are stripped off and RSPANed over VPLS	Supported	Not Supported
L2VPN	Not Supported	Supported	Not Supported
L3VPN	Not Supported	Supported	Not Supported
L3VPN over BDI	Not Supported	Supported	Not Supported
MPLS	Incoming Ethernet and VLAN header are stripped off and RSPANed over VPLS	Supported	Not Supported
Routed PW	Not Supported	Supported	Not Supported
VPLS	Not supported for bidirectional traffic	Supported	Not Supported

Table 6: RSPAN Traffic

Source	Ingress Mirror (Rx)	Egress Mirror (Tx)	Both
Layer2 or Layer3	Supported	Supported	Supported
VLAN	Supported	Not supported	Not supported
EFP	Not supported	Not supported	Not supported
Pseudowire	Not supported	Not supported	Not supported

The following table lists the supported **rewrite** traffic for RSPAN on the EFP, Trunk with the associated RSPAN Bridge Domains (BD).

Table 7: Rewrite Traffic for RSPAN BD

Rewrite Operations	Source	EFP/Trunk associated with RSPAN BD
no-rewrite	Pop1, Pop2, Push1	Only Pop1

The following tables lists the format of the spanned packets at the destination port for both Ingress and Egress RSPAN. The tables lists the formats of untagged, single, and double tagged source packets for EFPs under source port configured with **rewrite** operations (no-rewrite, pop1, pop2 and push1).

Table 8: Destination Port Ingress and Egress Spanned Traffic for EVC RSPAN BD

	Ingress Traffic	**Egress Traffic**
(Untagged Traffic) - Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + packet	RSPAN BD tag + packet
pop1 tag	NA	NA
pop2 tag	NA	NA
push1 tag	NA	NA
(Single Traffic)-Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + packet	RSPAN BD tag + source-outer-tag + packet
pop1 tag		
pop2 tag		NA
push1 tag		RSPAN BD tag + source-outer-tag + packet
(Double traffic) - Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + source-inner-tag + packet	RSPAN BD tag + Source-inner-tag + packet
pop1 tag		
pop2 tag		
push1 tag		

Table 9: Destination Port Ingress and Egress Spanned Traffic for TEFP RSPAN BD

	Ingress Traffic	**Egress Traffic**
(Untagged traffic)- Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + packet	RSPAN BD tag + packet
pop1 tag	NA	NA
pop2 tag	NA	NA

	Ingress Traffic	Egress Traffic
push1 tag	NA	NA
(Single traffic)-Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outertag + packet	RSPAN BD tag + source-outertag + packet
pop1 tag		
pop2 tag		NA
push1 tag		RSPAN BD tag + source-outertag + packet
(Double traffic) -Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outertag + source-innertag+ packet	RSPAN BD tag + source-outertag + source-innertag + packet
pop1 tag		
pop2 tag		
push1 tag		

Table 10: Destination Port Ingress and Egress Spanned Traffic for RSPAN BD with VPLS Pseudowire (RSP2 module)

	Ingress Traffic	Egress Traffic
(Untagged traffic) - Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + packet	RSPAN BD tag + packet
pop1 tag	NA	NA
pop2 tag	NA	NA
push1 tag	NA	NA
(Single traffic)- Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + packet	RSPAN BD tag + source-outer-tag + packet
pop1 tag		
pop2 tag	NA	NA
push1 tag	RSPAN BD tag + source-outer-tag + packet	RSPAN BD tag + source-outer-tag + packet

	Ingress Traffic	Egress Traffic
(Double traffic)-Source port rewrite	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**	**RSPAN VLAN (BD) rewrite pop1 tag symmetric**
no-rewrite	RSPAN BD tag + source-outer-tag + source-inner-tag + packet	RSPAN BD tag + source-outer-tag + source-inner-tag + packet
pop1 tag		
pop2 tag		
push1 tag		

Configuring Local SPAN and RSPAN

Configuring Sources and Destinations for Local SPAN

To configure sources and destinations for a SPAN session:

Procedure

Step 1 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 2 **monitor session {***session_number***} type local**

Example:

```
Router(config)# monitor session 1 type local
```

Specifies the local SPAN session number and enters the local monitoring configuration mode.

- *session_number*—Indicates the monitor session. The valid range is 1 through 14.

Step 3 **source interface** *interface_type slot/subslot/port* **[, | - | rx | tx | both]**

Example:

```
Router(config-mon-local)# source interface gigabitethernet 0/2/1 rx
```

Specifies the source interface and the traffic direction:

- *interface_type*—Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.
- ","—List of interfaces
- "–"—Range of interfaces

- rx—Ingress local SPAN
- tx—Egress local SPAN
- both

Step 4 **destination interface** *interface_type slot/subslot/port* **[, | -]**

Example:

```
Router(config-mon-local)# destination interface gigabitethernet 0/2/4
```

Specifies the destination interface that sends both ingress and egress local spanned traffic from source port to the prober or sniffer.

- *interface_type*—Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.
- “,”—List of interfaces
- “–”—Range of interfaces

Step 5 **no shutdown**

Example:

```
Router(config-mon-local)# no shutdown
```

Enables the local SPAN session.

Step 6 **End**

Removing Sources or Destinations from a Local SPAN Session

To remove sources or destinations from a local SPAN session, use the following commands beginning in EXEC mode:

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **no monitor session** *session-number*

Example:

```
Router(config)# no monitor session 2
```

Clears existing SPAN configuration for a session.

Configuring RSPAN Source Session

To configure the source for a RSPAN session:

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **monitor session** *RSPAN_source_session_number* **type rspan-source**

Example:

```
Router(config)# monitor session 1
 type rspan-source
```

Configures an RSPAN source session number and enters RSPAN source session configuration mode for the session.

- *RSPAN_source_session_number*—

  Valid sessions are 1 to 14.
- **rspan-source**—Enters the RSPAN source-session configuration mode.

Step 4 **Filter vlan***vlan id*

Example:

```
filter vlan 100
```

Applies the VLAN access map to the VLAN ID; valid values are from 1 to 4094.

Step 5 **source** {*single_interface* slot/subslot/port| *single_vlan* [**rx** | **tx** | **both**]

Example:

```
Router(config-mon-rspan-src)# source interface gigabitethernet 0/2/1 tx
```

Specifies the RSPAN session number, the source interfaces and the traffic direction to be monitored.

- *single_interface*—Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.
- *single_vlan*

  —Specifies the single VLAN.
- **both**

  —(Optional) Monitors the received and the transmitted traffic.
- **rx**

  —(Optional) Monitors the received traffic only.
- **tx**—(Optional) Monitors the transmitted traffic only.

Step 6 **destination remote vlan** *rspan_vlan_ID*

Example:

```
Router(config-mon-rspan-src)# destination remote vlan2
```

Associates the RSPAN source session number session number with the RSPAN VLAN.

- *rspan_vlan_ID*—Specifies the Vlan ID.

Note *rspan_vlan_ID* is the RSPAN BD that is configured under the EFP or port which carries the RSPANd traffic.

Step 7 **no shutdown**

Example:

```
Router(config-mon-rspan-src)# no shutdown
```

Enables RSPAN source.

Step 8 **end**

Example:

```
Router(config-mon-rspan-src)# end
```

Exists the configuration.

Configuring RSPAN Destination Session

To configure the destination for a RSPAN session for remote Vlan:

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **monitor session** *RSPAN_destination_session_number* **type rspan-destination**

Example:

```
Router(config)# monitor session 1 type rspan-destination
```

Configures a RPAN session.

- *RSPAN_destination_session_number*—Valid sessions are 1 to 80.
- **rspan-destination**—Enters the RSPAN destination-session configuration mode.

Step 4 **source remote vlan** *rspan_vlan_ID*

Example:

```
Router(config-mon-rspan-dst)# source remote vlan2
```

Associates the RSPAN destination session number RSPAN VLAN.

- *rspan_vlan_ID*—Specifies the Vlan ID

Step 5 **destination** {*single_interface slot/subslot/port}*

Example:

```
Router(config-mon-rspan-dst)# destination interface gigabitethernet 0/0/1
```

Associates the RSPAN destination session number with the destination port.

- *single_interface* —Specifies the Gigabit Ethernet or Ten Gigabit Ethernet interface.
  - *slot/subslot/port*—The location of the interface.

Step 6 **no shutdown**

Example:

```
Router(config-mon-rspan-dst)# no shutdown
```

Restarts the interface

Step 7 **end**

Example:

```
Router(config-mon-rspan-dst)# end
```

Exists the configuration

Removing Sources or Destinations from a RSPAN Session

To remove source or destination from a RSPAN session, delete and recreate the RSPAN session. The following are the steps:

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **no monitor session** *session number*

Example:

```
Router(config)# no monitor session 1
```

Exits monitor session.

Step 4 **end**

Example:

```
Router(config-mon-rspan-src)# end
```

Exits configuration mode.

Configuring RSPAN Source Session over VPLS Network

To configure the source for a RSPAN over VPLS Network:

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **monitor session** *RSPAN_source_session_number* **type rspan-source**

Example:

```
Router(config)#source int g0/0/1 [tx |rx|both]
Router(config)#destination remote VLAN 1000
```

Configures an RSPAN source session number and enters RSPAN over VPLS Network source session configuration mode for the session.

Step 4 **no shutdown**

Example:

```
Router(config-mon-rspan-src)# no shutdown
```

Enables RSPAN over VPLS Network source.

Step 5 **end**

Example:

```
Router(config-mon-rspan-src)# end
```

Exits the configuration.

Note You must ensure that the BDI number should match RSPAN destination remote VLAN number.

Configuring L2VPN VFI in RSPAN Source Session

To configure the source for a RSPAN over VPLS Network using L2VPN Virtual Forwarding Instance (VFI):

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Step 3 **l2vpn vfi context VPLS** *1000*

Example:

```
l2vpn vfi context VPLS1000
 vpn id 1000
 member 10.0.0.1 encapsulation mpls
```

source int g0/0/1 [tx |rx|both]

Step 4 **source interface***[tx |rx|both]*

Example:

```
l2vpn vfi context VPLS1000
 vpn id 1000
 member 10.0.0.1 encapsulation mpls
source int g0/0/1 [tx |rx|both]
```

Step 5 **no shutdown**

Example:

```
Router(config-mon-rspan-sour)# no shutdown
```

Enables RSPAN over VPLS Network at source.

Step 6 **end**

Example:

```
Router(config-mon-rspan-sour)# end
```

Exits the configuration.

Configuring L2VPN VFI in RSPAN Destination Session

To configure the destination for a RSPAN over VPLS Network using L2VPN Virtual Forwarding Instance (VFI):

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Step 3 **l2vpn vfi context VPLS***1000*

Example:

```
l2vpn vfi context VPLS1000
 vpn id 1000
 member 10.0.0.2 encapsulation mpls

bridge-domain 1000
 member GigabitEthernet0/1/0 service-instance 1
 member vfi VPLS1000
```

Step 4 **destination interface***[tx |rx|both]*

Example:

```
l2vpn vfi context VPLS1000
 vpn id 1000
 member 10.0.0.2 encapsulation mpls

destination int g0/0/1 [tx |rx|both]
```

Step 5 **no shutdown**

Example:

```
Router(config-mon-rspan-dest)# no shutdown
```

Enables RSPAN over VPLS Network at destination.

Step 6 **end**

Example:

```
Router(config-mon-rspan-dest)# end
```

Exits the configuration.

Configuring MAC Limit on RSPAN over VPLS Network in RSPAN Destination Session

To configure the MAC Limit 0 on RSPAN BD on destination router:

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Step 3 **mac-address-table-limit** *bdomain <bd-id> maximum 0 action limit*

Example:

```
router#show bridge-domain 1000
Bridge-domain 1000 (2 ports in all)
State: UP
Mac learning: Enabled
Aging-Timer: 300 second(s)

Maximum address limit: 0
    GigabitEthernet0/1/0 service instance 1
    vfi VPLS1000 neighbor 10.0.0.2 1000
```

Sample Configurations

The following sections contain configuration examples for SPAN and RSPAN.

Configuration Example: Local SPAN

The following example shows how to configure local SPAN session 8 to monitor bidirectional traffic from source interface Gigabit Ethernet interface to destination:

```
Router(config)# monitor session 8 type local
Router(config)# source interface gigabitethernet 0/0/10
Router(config)# destination interface gigabitethernet 0/0/3
Router(config)# no shut
```

Configuration Example: Removing Sources or Destinations from a Local SPAN Session

This following example shows how to remove a local SPAN session:

```
Router(config)# no monitor session 8
```

Configuration Example: RSPAN Source

The following example shows how RSPAN session 2 to monitor bidirectional traffic from source interface Gigabit Ethernet 0/0/1:

```
Router(config)# monitor session 2 type RSPAN-source
Router(config-mon-RSPAN-src)# source interface gigabitEthernet0/0/1 [tx |rx|both]
Router(config-mon-RSPAN-src)# destination remote VLAN 100
Router(config-mon-RSPAN-src)# no shutdown
Router(config-mon-RSPAN-src)# end
```

The following example shows how RSPAN session 3 to monitor bidirectional traffic from source Vlan 200:

```
Router(config)# monitor session 3 type RSPAN-source
Router(config-mon-RSPAN-src)# filter vlan 100
Router(config-mon-RSPAN-src)# source interface Te0/0/23 rx
Router(config-mon-RSPAN-src)# destination remote VLAN 200
Router(config-mon-RSPAN-src)# no shutdown
Router(config-mon-RSPAN-src)# end
```

Configuration Example: RSPAN Destination

The following example shows how to configure interface Gigabit Ethernet 0/0/1 as the destination for RSPAN session 2:

```
Router(config)# monitor session 2 type RSPAN-destination
Router(config-mon-RSPAN-dst)# source remote VLAN 100
Router(config-mon-RSPAN-dst)# destination interface gigabitEthernet 0/0/1
Router(config-mon-RSPAN-dst)# end
```

Verifying Local SPAN and RSPAN

Use the **show monitor session** command to view the sessions configured.

- The following example shows the Local SPAN source session with Tx as source:

```
Router# show monitor session 8
Session 8
---------
Type : Local Session
Status : Admin Enabled
Source Ports :
TX Only : Gi0/0/10
Destination Ports : Gi0/0/3
MTU : 1464
Dest RSPAN VLAN : 100
```

- The following example shows the RSPAN source session with Gigabit Ethernet interface 0/0/1 as source:

```
Router# show monitor session 2
Session 2
---------
Type                  : Remote Source Session
Status                : Admin Enabled
Source Ports          :
    Both              : Gi0/0/1
MTU                   : 1464
```

- The following example shows the RSPAN source session with Vlan 20 as source:

```
Router# show monitor session 3
Session 3
---------
Type                  : Remote Source Session
Status                : Admin Enabled
Source VLANs          :
    RX Only           : 20
MTU                   : 1464
```

- The following example shows the RSPAN destination session with Gigabit Ethernet interface 0/0/1 as destination:

```
Router# show monitor session 2
Session 2
---------
Type                  : Remote Destination Session
Status                : Admin Enabled
Destination Ports     : Gi0/0/1
MTU                   : 1464
Source RSPAN VLAN : 100
```

Verifying RSPAN over VPLS Network

Use the **show monitor session** command to view the sessions configured.

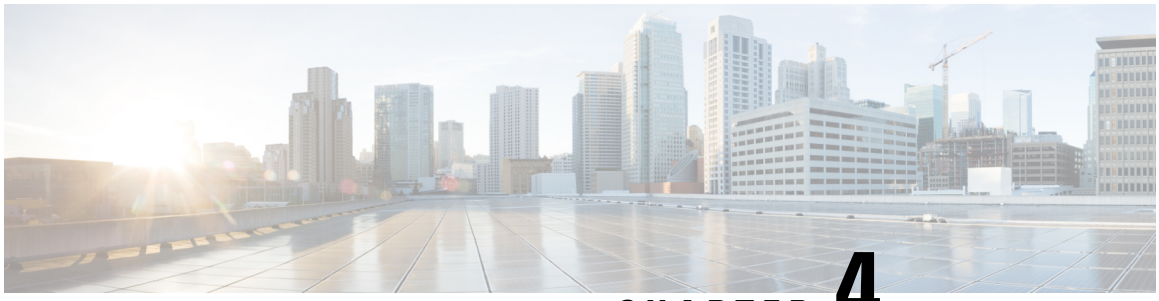
The following example shows the RSPAN over VPLS Source session

```
Router(config)#show mpls l2transport vc

Local intf     Local circuit              Dest address    VC ID      Status
-------------  -------------------------- --------------- ---------- ----------
```

```
VFI VPLS1000   vfi                         10.0.0.1         1000       UP
```

The following example shows the RSPAN over VPLS Destination session

```
Router(config)#show mpls l2transport vc

Local intf     Local circuit              Dest address    VC ID      Status
-------------  -------------------------- --------------- ---------- ----------
VFI VPLS1000   vfi                         10.0.0.2         1000       UP
```



CHAPTER 4

Layer 2 Access Control Lists on EVCs

The ability to filter packets in a modular and scalable way is important for both network security and network management. Access Control Lists (ACLs) provide the capability to filter packets at a fine granularity. In Metro Ethernet networks, ACLs are directly applied on Ethernet virtual circuits (EVCs).

Layer 2 Access Control Lists on EVCs is a security feature that allows packet filtering based on MAC addresses. This module describes how to implement ACLs on EVCs.

Prerequisites for Layer 2 Access Control Lists on EVCs

- Knowledge of how service instances must be configured.
- Knowledge of extended MAC ACLs and how they must be configured.

Prerequisites for Layer 2 Access Control Lists on EVCs

- Knowledge of how service instances must be configured.
- Knowledge of extended MAC ACLs and how they must be configured.

Restrictions for Layer 2 Access Control Lists on EVCs

- You can enable a packet capture on the host, based on Layer 2 packet header per EFP (for example, **dst-mac**, **src-mac** and CoS field). Create a **pcap** of the captured packet on host machine.
- A maximum of 16512 access control entries (ACEs) are allowed for a given ACL, with the limitation that it does not exceed the maximum tcam entries.

- Only 256 different or unique Layer 2 ACLs can be configured on a line card. (More than 256 ACLs can be configured on a router and it depends on the number of TCAM that is free for programming these ACLs.)
- L2 ACL is supported over port channel with Normal EFPs.
- Egress L2 ACL on EVC is *not* supported.
- L2 ACLs are *not* supported on Trunk EFP.
- L2 ACL counters are *not* supported.
- Layer2 ACL can be applied on layer 2 frame without IPv4 or IPv6 header as layer 2 ACL does not support filter on IPv4 or IPv6 traffic.
- Layer 2 ACLs function inbound only. The Layer 2 ACLs are *not* supported at physical interface level.
- Current Layer 2 ACLs provide Layer 3 filtering options in permit and deny rules. Options that are not relevant to service instances are ignored.

EVCs

An Ethernet virtual circuit (EVC) as defined by the Metro Ethernet Forum is a port-level point-to-point or multipoint-to-multipoint Layer 2 circuit. It is an end-to-end representation of a single instance of a Layer 2 service being offered by a provider to a customer. An EVC contains the different parameters on which the service is being offered. A service instance is the instantiation of an EVC on a specified port.

Service instances are configured under a port channel. The traffic carried by the service instance is load balanced across member links. Service instances under a port channel are grouped and each group is associated with one member link. Ingress traffic for a single EVC can arrive on any member of the bundle. All egress traffic for a service instance uses only one of the member links. Load balancing is achieved by grouping service instances and assigning them to a member link.

Ethernet virtual connection services (EVCS) uses the EVCs and service instances to provide Layer 2 switched Ethernet services. EVC status can be used by a customer edge (CE) device either to find an alternative path to the service provider network or in some cases, to fall back to a backup path over Ethernet or over another alternative service such as ATM.

For information about the Metro Ethernet Forum standards, see the Standards table in the “Additional References” section.

Relationship Between ACLs and Ethernet Infrastructure

The following points capture the relationship between ACLs and Ethernet Infrastructure (EI):

- ACLs can be directly applied on an EVC using the command-line interface (CLI). An ACL is applied to a service instance, which is the instantiation of an EVC on a given port.
- One ACL can be applied to more than one service instance at any time.
- One service instance can have one ACL at most applied to it at any time. If a Layer 2 ACL is applied to a service instance that already has a Layer 2 ACL, the new one replaces the old one.
- Only named ACLs can be applied to service instances. The command syntax ACLs is retained; the **mac access-list extended** command is used to create an ACL.

- The **show ethernet service instance id** *id* **interface** *type number* detail **show ethernet service instance** command can be used to provide details about ACLs on service instances.

Information About Layer 2 Access Control Lists on EVCs

Creating a Layer 2 ACL

Perform this task to create a Layer 2 ACL with a single ACE.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **mac access-list extended** *name*

Example:

```
Device(config)# mac access-list extended test-12-acl
```

Defines an extended MAC ACL and enters mac access list control configuration mode.

Step 4 **permit** {{*src-mac mask* | **any**} {*dest-mac mask* | **any**} [*protocol* [**vlan** *vlan*] [*cos value*]]}

Example:

```
Device(config-ext-macl)# permit 00aa.00bb.00cc 0.0.0 any
```

Allows forwarding of Layer 2 traffic if the conditions are matched. Creates an ACE for the ACL.

Applying a Layer 2 ACL to a Service Instance

Perform this task to apply a Layer 2 ACL to a service instance. Note that packet filtering takes place only after the ACL has been created and applied to the service instance.

Before you begin

Before applying an ACL to a service instance, you must create it using the **mac access-list extended command. See the "Creating a Layer 2 ACL" section.**

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet 1/0/0
```

Specifies the type and location of the interface to configure, where:

- *type* --Specifies the type of the interface.
- *number* --Specifies the location of the interface.

Step 4 **service instance** *id* ethernet

Example:

```
Device(config-if)# service instance 100 ethernet
```

Configures an Ethernet service instance on an interface and enters Ethernet service configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used in order to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **mac access-group** *access-list-name* in

Example:

```
Device(config-if-srv)# mac access-group test-12-acl in
```

Applies a MAC ACL to control incoming traffic on the interface.

Step 7 **bridge -domain** *bridge-id* in

Example:

```
Device(config-if-srv)# bridge-domain 100
```

Configure the bridge domain ID.

Configuring a Layer 2 ACL with ACEs on a Service Instance

Perform this task to configure the same ACL with three ACEs and stop all other traffic on a service instance.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **mac access-list extended** *name*

Example:

```
Device(config)# mac access list extended test-12-acl
```

Defines an extended MAC ACL and enters mac access control list configuration mode.

Step 4 **permit** {*src-mac mask* | **any**} {*dest-mac mask* | **any**}

Example:

```
Device(config-ext-macl)# permit 00aa.bbcc.ddea 0.0.0 any
```

Allows forwarding of Layer 2 traffic if the conditions are matched. This creates an ACE for the ACL.

Step 5 **permit** {*src-mac mask* | **any**} {*dest-mac mask* | **any**}

Example:

```
Device(config-ext-macl)# permit 00aa.bbcc.ddeb 0.0.0 any
```

Allows forwarding of Layer 2 traffic if the conditions are matched. This creates an ACE for the ACL.

Step 6 **permit** {*src-mac mask* | **any**} {*dest-mac mask*} | **any**}

Example:

```
Device(config-ext-macl)# permit 00aa.bbcc.ddec 0.0.0 any
```

Allows forwarding of Layer 2 traffic if the conditions are matched. This creates an ACE for the ACL.

Step 7 **deny any any**

Example:

```
Device(config-ext-macl)# deny any any
```

Prevents forwarding of Layer 2 traffic except for the allowed ACEs.

Step 8 **exit**

Example:

```
Device(config-ext-macl)# exit
```

Exits the current command mode and returns to global configuration mode.

Step 9 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet 1/0/0
```

Specifies the interface.

Step 10 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 200 ethernet
```

Configures an Ethernet service instance on an interface and enters service instance configuration mode.

Step 11 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used to map ingress dot1q frames on an interface to the appropriate service instance.

Step 12 **mac access-group** *access-list-name* **in**

Example:

```
Device(config-if-srv)# mac access-group test-12-acl in
```

Applies a MAC ACL to control incoming traffic on the interface.

Verifying the Presence of a Layer 2 ACL on a Service Instance

Perform this task to verify that a Layer 2 ACL is present on an EVC. This verification task can be used after an ACL has been configured to confirm its presence.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 configure terminal

Example:

```
Device# show ethernet service instance id 100 interface gigabitethernet 3/0/1 detail
```

Enters global configuration mode.

Step 3 **show ethernet service instance id** *id* **interface** *type* *number* detail

Example:

```
Device# show ethernet service instance id 100 interface gigabitethernet 3/0/1 detail
```

Displays detailed information about Ethernet customer service instances.

Configuration Examples for Layer 2 Access Control Lists on EVCs

Example Applying a Layer 2 ACL to a Service Instance

The following example shows how to apply a Layer 2 ACL called mac-20-acl to a service instance. The ACL has five permitted ACEs and all other traffic is not allowed.

```
enable
configure terminal
 mac access-list extended mac-20-acl
```

```
 permit 00aa.bbcc.adec 0.0.0 any

 permit 00aa.bbcc.bdec 0.0.0 any

 permit 00aa.bbcc.cdec 0.0.0 any

 permit 00aa.bbcc.edec 0.0.0 any

 permit 00aa.bbcc.fdec 0.0.0 any

 deny any any
 exit
interface gigabitethernet 10/0/0
 service instance 100 ethernet
 encapsulation dot1q 100
 mac access-group mac-20-acl in
```

Example Applying a Layer 2 ACL to Three Service Instances on the Same Interface

The following example shows how to apply a Layer 2 ACL called mac-07-acl to three service instances on the same interface:

```
enable
configure terminal
mac access-list extended mac-07-acl

permit 00aa.bbcc.adec 0.0.0 any

permit 00aa.bbcc.bdec 0.0.0 any

permit 00aa.bbcc.cdec 0.0.0 any

deny any any
exit
interface gigabitethernet 10/0/0
service instance 100 ethernet
encapsulation dot1q 100
mac access-group mac-07-acl in
service instance 101 ethernet
encapsulation dot1q 101
mac access-group mac-07-acl in
service instance 102 ethernet
encapsulation dot1q 102
mac access-group mac-07-acl in
```

Verifying the Presence of a Layer 2 ACL on a Service Instance

Perform this task to verify that a Layer 2 ACL is present on an EVC. This verification task can be used after an ACL has been configured to confirm its presence.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 configure terminal

Example:

```
Device# show ethernet service instance id 100 interface gigabitethernet 3/0/1 detail
```

Enters global configuration mode.

Step 3 **show ethernet service instance id** *id* **interface** *type number* detail

Example:

```
Device# show ethernet service instance id 100 interface gigabitethernet 3/0/1 detail
```

Displays detailed information about Ethernet customer service instances.

Example Displaying the Details of a Layer 2 ACL on a Service Instance

The following sample output displays the details of a Layer 2 ACL called test-acl on a service instance.

```
Device# show ethernet service instance id 100 interface ethernet0/0 detail
Service Instance ID: 100
L2 ACL (inbound): test-acl
Associated Interface: Ethernet0/0
Associated EVC: test
L2protocol drop
CEVlans:
Interface Dot1q Tunnel Ethertype: 0x8100
State: Up
L2 ACL permit count: 10255
L2 ACL deny count: 53
```

```
Device# show ethernet service instance id 100 interface gig3/0/1 detail
Service Instance ID: 100
L2 ACL (inbound): test-acl
Associated Interface: Gig3/0/1
Associated EVC: test
L2protocol drop
CEVlans:
Interface Dot1q Tunnel Ethertype: 0x8100
State: Up
L2 ACL permit count: 10255
L2 ACL deny count: 53
```

The table below describes the significant fields in the output.

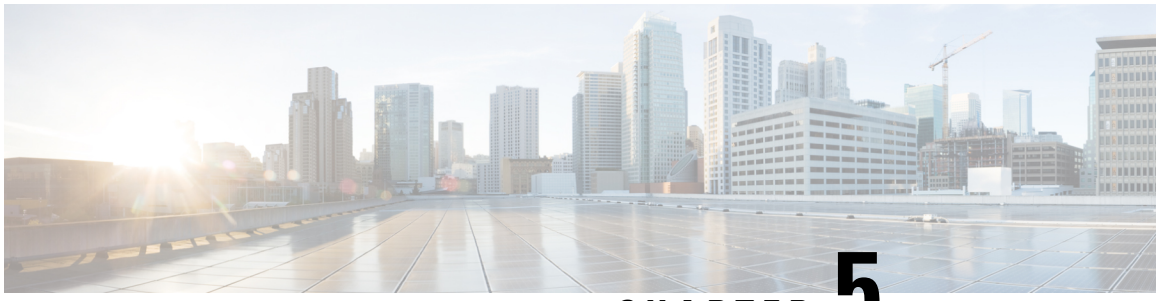
Table 11: show ethernet service instance Field Descriptions

Field	Description
Service Instance ID	Displays the service instance ID.
L2 ACL (inbound):	Displays the ACL name.
Associated Interface:	Displays the interface details of the service instance.
Associated EVC:	Displays the EVC with which the service instance is associated.
CEVlans:	Displays details of the associated VLAN ID.
State:	Displays whether the service instance is in an up or down state.
L2 ACL permit count:	Displays the number of packet frames allowed to pass on the service instance by the ACL.
L2 ACL deny count	Displays the number of packet frames not permitted to pass on the service instance by the ACL.

Example Displaying the Details of Configured Layer 2 ACL

The following sample output displays the details of a configured Layer 2 ACL.

```
Device# show access-lists
Extended IP access list ip-acl
10 permit ip any any
Extended MAC access list mac-acl
permit any any vlan 10
Device#
Device#sh access-lists mac-acl
Extended MAC access list mac-acl
permit any any vlan 10
```



CHAPTER 5

Configuring MAC Address Security on Service Instances and EVC Port Channels

The MAC Address Security on Service Instances and EVC Port Channels feature addresses port security with service instances by providing the capability to control and filter MAC address learning behavior at the granularity of a per-service instance. When a violation requires a shutdown, only the customer who is assigned to a given service instance is affected and--not all customers who are using the port. MAC address limiting is a type of MAC security and is also referred to as a MAC security component or element.

Prerequisites for MAC Address Security on Service Instances and EVC Port Channels

- An understanding of service instances and bridge domains.
- An understanding of the concepts of MAC address limiting and how it is used for MAC security.
- An understanding of how port channels and EtherChannels work in a network.

Restrictions for MAC Security on the RSP3 Module

- Maximum MAC scale supported on the RSP3 module is 200000 entries.
- Maximum number of secure EFPs supported is 255.
- Maximum number of secure MAC entries is 16K.
- Maximum number of MAC addresses supported per BD is 64K.

- Maximum number of secure MAC entries per BD is 10K.
- MAC security is supported only on Layer2 EFPs.
- MAC security on local connect and cross-conect is *not* supported.
- MAC security is *not* supported on trunk EFP.

Information About MAC Address Security on Service Instances and EVC Port Channels

Ethernet Virtual Circuits, Service Instances, and Bridge Domains

An Ethernet virtual circuit (EVC) as defined by the Metro Ethernet Forum is a port-level point-to-point or multipoint-to-multipoint Layer 2 circuit. It is an end-to-end representation of a single instance of a Layer 2 service being offered by a provider to a customer. An EVC embodies the different parameters on which the service is being offered. A service instance is the instantiation of an EVC on a given port.

Support for Ethernet bridging is an important Layer 2 service that is offered on a router as part of an EVC. Ethernet bridging enables the association of a bridge domain with a service instance.

Service instances are configured under a port channel. The traffic carried by service instances is load-balanced across member links. Service instances under a port channel are grouped and each group is associated with one member link. Ingress traffic for a single service instance can arrive on any member of the bundle. All egress traffic for a service instance uses only one of the member links. Load-balancing is achieved by grouping service instances and assigning them to a member link.

For information about the Metro Ethernet Forum standards, see the “Standards” table in the “Additional References” section.

EVCs on Port Channels

An EtherChannel bundles individual Ethernet links into a single logical link that provides the aggregate bandwidth of up to eight physical links. The Ethernet Virtual Connection Services (EVCS) EtherChannel feature provides support for EtherChannels on service instances.

Note The MAC Address Security on EVC Port Channel services is supported only on bridge domains over Ethernet and is not supported on xconnect services.

EVCS uses the concepts of EVCs and service instances.

Load balancing is done on an Ethernet flow point (EFP) basis where a number of EFPs exclusively pass traffic through member links.

MAC Security and MAC Addressing

MAC security is enabled on a service instance by configuring the **mac security** command. Various MAC security elements can be configured or removed regardless of whether the **mac security** command is presently configured, but these configurations become operational only when the **mac security** command is applied.

In this document, the term “secured service instance” is used to describe a service instance on which MAC security is configured. The MAC addresses on a service instance on which MAC security is configured are referred to as “secured MAC addresses.” Secured MAC addresses can be either statically configured (as a permit list) or dynamically learned.

MAC Address Permit List

A permit list is a set of MAC addresses that are permitted on a service instance. Permitted addresses permanently configured into the MAC address table of the service instance.

On a service instance that is a member of a bridge domain, the operator is permitted to configure one or more permitted MAC addresses.

For each permitted address, eligibility tests are performed and after the address passes these tests, it is either:

- Programmed into the MAC address table of the bridge domain, if MAC security is enabled on the service instance or,
- Stored in an area of memory referred to as “MAC table cache” if MAC security is not enabled on the service instance. When MAC security is enabled, the addresses from the MAC table cache are added to the MAC address table as secure addresses.

The eligibility tests performed when a user tries to add a MAC address to the permit list on a service instance are as follows:

- If the address is already a denied address on the service instance, the configuration is rejected with an appropriate error message.
- If the acceptance of this address would increase the secure address count on the service instance beyond the maximum number allowed, an attempt is made to make room by removing an existing address from the MAC address table. The only candidate for removal is a dynamically learned address on the service instance. If sufficient room cannot be made, the configuration is rejected. If the acceptance of this address would increase the secure address count on the bridge domain beyond the maximum number allowed, an attempt is made to make room by removing an existing address from the MAC address table. The only candidate for removal is a dynamically learned address on the service instance. If room cannot be made, the configuration is rejected.

Note Default maximum address is '1' for a service instance.

- If the address is already permitted on another service instance in the same bridge domain, one of the following actions occur:
  - If the conflicting service instance has MAC security configured, the configuration is rejected with an appropriate error message.

- If the conflicting service instance does not have MAC security configured, the configuration is accepted silently. (If the operator attempts to enable MAC security on the conflicting service instance, that attempt fails.)

MAC Address Deny List

A deny list is a set of MAC addresses that are not permitted on a service instance. An attempt to learn a denied MAC address will fail. On a service instance that is a member of a bridge domain, the operator is permitted to configure one or more denied MAC addresses. The arrival of a frame with a source MAC address that is part of a deny list will trigger a violation response.

Before a denied address can be configured, the following test is performed:

- If the address is already configured as a permitted address on the specific service instance or if the address has been learned and saved as a sticky address on the service instance, the configuration is rejected with an appropriate error message.

In all other cases, the configuration of the denied address is accepted. Typical cases include:

- The address is configured as a permitted address on another service instance in the same bridge domain, or the address has been learned and saved as a sticky address on another service instance.
- The address is present in the MAC table of the bridge domain as a dynamically learned address on the specific service instance and is deleted from the MAC table before the configuration is accepted.

Violation Response Configuration

A violation response is a response to a MAC security violation or a failed attempt to dynamically learn a MAC address due to an address violation. MAC security violations are of two types:

Type 1 Violation --The address of the ingress frame cannot be dynamically learned due to a deny list, or because doing so would cause the maximum number of secure addresses to be exceeded (see the *MAC Address Limiting and Learning*).

Type 2 Violation --The address of the ingress frame cannot be dynamically learned because it is already "present" on another secured service instance (see the *MAC Move and MAC Locking*) in the same bridge-domain.

There are three possible sets of actions that can be taken in response to a violation:

1. **Shutdown**
    - The ingress frame is dropped.
    - The service instance on which the offending frame arrived is shut down.
    - The violation count is incremented, and the violating address is recorded for later CLI display.
    - The event and the response are logged to SYSLOG.
2. **Restrict**
    - The ingress frame is dropped.
    - The violation count is incremented, and the violating address is recorded for display.

- The event and the response are logged to SYSLOG.

3. **Protect**
   - The ingress frame is dropped.

Note The ingress frame is dropped silently, without sending any violation report to the SYSLOG.

Note The Restrict and Protect modes are applied on EFP level to discard the traffic. Both the modes are not applied on the Erroneous MAC level.

If a violation response is not configured, the default response mode is shutdown. The violation response can be configured to protect or restrict mode. A "no" form of a violation response, sets the violation response to the default mode of shutdown.

You are allowed to configure the desired response for a Type 1 and Type 2 violations on a service instance. For a Type 1 violation on a bridge domain (that is, if the learn attempt conforms to the policy configured on the service instance, but violates the policy configured on the bridge domain), the response is always "Protect." This is not configurable.

In shutdown mode, the service instance is put into the error disabled state immediate, an SNMP trap notification is transmitted, and a message is sent to the console and SYSLOG as shown below:

```
%ETHER_SERVICE-6-ERR_DISABLED:
Mac security violation - shutdown service instance 100 on interface gig 0/0/0
```

To bring a service instance out of the error-disabled state, use **errdisable recovery cause mac-security** command to set the auto recovery timer, or re-enable it using the EXEC command **clear ethernet service instance id** *id* **interface** *type number* **errdisable**.

In Restrict mode, the violation report is sent to SYSLOG at level LOG_WARNING.

Support for the different types of violation responses depends on the capabilities of the platform. The desired violation response can be configured on the service instance. The configured violation response does not take effect unless and until MAC security is enabled using the **mac security** command.

MAC Address Aging Configuration

A specific time scheduler can be set to age out secured MAC addresses that are dynamically learned or statically configured on both service instances and bridge domains, thus freeing up unused addresses from the MAC address table for other active subscribers.

The set of rules applied to age out secured MAC addresses is called secure aging. By default, the entries in the MAC address table of a secured service instance are never aged out. This includes permitted addresses and dynamically learned addresses.

The **mac security aging time** *aging-time* command sets the aging time of the addresses in the MAC address table to <*n* > minutes. By default, this affects only dynamically learned (not including sticky) addresses--permitted addresses and sticky addresses are not affected by the application of this command.

By default, the aging time <n> configured via the **mac security aging time** *aging-time* command is an absolute time. That is, the age of the MAC address is measured from the instant that it was first encountered on the service instance. This interpretation can be modified by using the **mac security aging time** *aging-time* **inactivity** command, which specifies that the age <n> be measured from the instant that the MAC address was last encountered on the service instance.

The **mac security aging static**and **mac security aging sticky** commands specify that the **mac security aging time**aging-time command must be applicable to permitted and sticky MAC addresses, respectively. In the case of permitted MAC addresses, the absolute aging time is measured from the time the address is entered into the MAC address table (for example, when it is configured or whenever the **mac security** command is entered--whichever is later).

If the **mac security aging time** command is not configured, the **mac security aging static** command has no effect.

Sticky MAC Address Configurations

The ability to make dynamically learned MAC addresses on secured service instances permanent even after interface transitions or device reloads can be set up and configured. A dynamically learned MAC address that is made permanent on a secured service instance is called a “sticky MAC address”. The **mac security sticky** command is used to enable the sticky MAC addressing feature on a service instance.

With the “sticky” feature enabled on a secured service instance, MAC addresses learned dynamically on the service instance are kept persistent across service instance line transitions and device reloads.

The sticky feature has no effect on statically configured MAC addresses. The sticky addresses are saved in the running configuration. Before the device is reloaded, it is the responsibility of the user to save the running configuration to the startup configuration. Doing this will ensure that when the device comes on, all the MAC addresses learned dynamically previously are immediately populated into the MAC address table.

The **mac security sticky address** *mac-address* command can configure a specific MAC address as a sticky MAC address. The use of this command is not recommended for the user because configuring a MAC address as a static address does the same thing. When sticky MAC addressing is enabled by the **mac security sticky** command, the dynamically learned addresses are marked as sticky and a **mac security sticky address** *mac-address* command is automatically generated and saved in the running configuration for each learned MAC address on the service instances.

Aging for Sticky Addresses

MAC addresses learned on a service instance that has the sticky behavior enabled are subject to aging as configured by the **mac security aging time** and **mac security aging sticky** commands. In other words, for the purpose of aging functionality, sticky addresses are treated the same as dynamically learned addresses.

Transitions

This section contains a description of the expected behavior of the different MAC security elements when various triggers are applied; for example, configuration changes or link state transitions.

MAC Security Enabled on a Service Instance

When MAC security is enabled on a service instance, all existing MAC table entries for the service instance are purged. Then, permitted MAC address entries and sticky addresses are added to the MAC table, subject to the prevailing MAC address limiting constraints on the bridge domain.

If MAC address limits are exceeded, any MAC address that fails to get added is reported via an error message to the console, the attempt to enable MAC security on the service instance fails, and the already added permitted entries are backed out or removed.

The aging timer for all entries is updated according to the secure aging rules.

MAC Security Disabled on a Service Instance

The existing MAC address table entries for this service instance are purged.

Service Instance Moved to a New Bridge Domain

This transition sequence applies to all service instances, whether or not they have MAC security configured. All the MAC addresses on this service instance in the MAC address table of the old bridge domain are removed. The count of dynamically learned addresses in the old bridge domain is decremented. Then, all the MAC security commands are permanently erased from the service instance.

Service Instance Removed from a Bridge Domain

All the MAC addresses in the MAC address table that attributable to this service instance are removed, and the count of dynamically learned addresses in the bridge domain is decremented. Since MAC security is applicable only on service instances that are members of a bridge domain, removing a service instance from a bridge domain causes all the MAC security commands to be erased permanently.

Service Instance Shut Down Due to Violation

All dynamically learned MAC addresses in the MAC address table are removed, and all the other MAC security state values are left unchanged. The only change is that no traffic is forwarded, and therefore no learning can take place.

Interface Service Instance Down Linecard OIR Removed

The MAC tables of all the affected bridge domains are cleared of all the entries attributable to the service instances that are down.

Interface Service Instance Re-activated Linecard OIR Inserted

The static and sticky address entries in the MAC tables of the affected bridge domains are re-created to the service instances that are activated.

MAC Address Limit Decreased

When the value of the MAC address limit on the service instance is changed initially, a sanity check is performed to ensure that the new value of <n> is greater than or equal to the number of permitted entries. If not, the command is rejected. The MAC table is scanned for addresses that are attributable to this service instance, and dynamically learned MAC addresses are removed when the new MAC address limit is less than the old MAC address limit.

When the value of <n> on a bridge domain is changed initially, a sanity check is performed to ensure that the new value of <n> is greater than or equal to the sum of the number of permitted entries on all the secured service instances on the bridge domain. If this sanity test fails, the command is rejected. The bridge domain MAC address table (regardless of service instance) is scanned for dynamically learned (or sticky) addresses. All dynamically learned addresses are removed when the new MAC address limit is less than the old MAC address limit.

Sticky Addresses Added or Removed on a Service Instance

Existing dynamically learned MAC addresses remain unchanged. All new addresses learned become “sticky” addresses.

Disabling sticky addresses causes all sticky secure MAC addresses on the service instance to be removed from the MAC address table. All new addresses learned become dynamic addresses on the service instance and are subject to aging.

How to Configure MAC Address Limiting on Service Instances Bridge Domains and EVC Port Channels

Enabling MAC Security on a Service Instance

Perform this task to enable MAC address security on a service instance.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet2/0/1
```

Specifies the interface type and number, and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used in order to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 200
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 8 **end**

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Enabling MAC Security on an EVC Port Channel

Before you begin

Note

- Bridge-domain, xconnect, and Ethernet virtual circuits (EVCs) are allowed only over the port channel interface and the main interface.
- If you configure a physical port as part of a channel group, you cannot configure EVCs under that physical port.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface port-channel** *channel-group*

Example:

```
Device(config)# interface port-channel 2
```

Specifies the port channel group number and enters interface configuration mode.

- Acceptable values are integers from 1 to 64.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used in order to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 200
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 8 end

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Configuring a MAC Address Permit List

Perform this task to configure permitted MAC addresses on a service instance that is a member of a bridge domain.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet2/0/1
```

Specifies the interface type and number, and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance (an instance of an EVC) on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used for mapping ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 200
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 **mac security address permit** *mac-address*

Example:

```
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaaa
```

Adds the specified MAC address as a permit MAC address for the service instance.

Step 8 **mac security address permit** *mac-address*

Example:

```
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaab
```

Adds the specified MAC address as a permitted MAC address for the service instance.

Step 9 **mac security address permit** *mac-address*

Example:

```
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaac
```

Adds the specified MAC address as a permitted MAC address for the service instance.

Step 10 **mac security address permit** *mac-address*

Example:

```
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaad
```

Adds the specified MAC address as a permitted MAC address for the service instance.

Step 11 **mac security address permit** *mac-address*

Example:

```
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaae
```

Adds the specified MAC address as a permitted MAC address for the service instance.

Step 12 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 13 **end**

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Configuring a MAC Address Deny List

Perform this task to configure a list of MAC addresses that are not allowed on a service instance that is a member of a bridge domain.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet2/0/1
```

Specifies the interface type and number, and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance (an instance of an EVC) on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used in order to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 200
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 **mac security address deny** *mac-address*

Example:

```
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaaa
```

Adds the specified MAC address as a denied MAC address for the service instance.

Step 8 **mac security address deny** *mac-address*

Example:

```
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaab
```

Adds the specified MAC address as a denied MAC address for the service instance.

Step 9 **mac security address deny** *mac-address*

Example:

```
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaac
```

Adds the specified MAC address as a denied MAC address for the service instance.

Step 10 **mac security address deny** *mac-address*

Example:

```
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaad
```

Adds the specified MAC address as a denied MAC address for the service instance.

Step 11 **mac security address deny** *mac-address*

Example:

```
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaae
```

Adds the specified MAC address as a denied MAC address for the service instance.

Step 12 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 13 **end**

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Configuring MAC Address Security on a Service Instance

Perform this task to configure an upper limit for the number of secured MAC addresses allowed on a service instance. This number includes addresses added as part of a permit list as well as dynamically learned MAC addresses. If the upper limit is decreased, all learned MAC entries are removed.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet2/0/1
```

Specifies the interface type and number, and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance (an instance of an EVC) on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 200
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 **mac security maximum addresses** *maximum-addresses*

Example:

```
Device(config-if-srv)# mac security maximum addresses 500
```

Sets the maximum number of secure addresses permitted on the service instance.

Note Default value for a service instance is '1'.

Step 8 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 9 **end**

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Configuring a MAC Address Violation

Perform this task to specify the expected behavior of a device when an attempt to dynamically learn a MAC address fails because the configured MAC security policy on the service instance was violated.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet2/0/1
```

Specifies the interface type and number, and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance (an instance of an EVC) on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 100
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 Do one of the following:

- **mac security violation restrict**
- **mac security violation protect**

Example:

```
Device(config-if-srv)# mac security violation restrict
```

Example:

```
Device(config-if-srv)# mac security violation protect
```

Sets the violation mode (for Type 1 and 2 violations) to restrict.

or

Sets the violation mode (for Type 1 and 2 violations) to protect.

- If a MAC security violation response is not specified, by default, the violation mode is shutdown.

Step 8 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 9 **end**

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Configuring MAC Address Aging

Perform this task to configure the aging of secured MAC addresses under MAC security. Secured MAC addresses are not subject to the normal aging of MAC table entries. If aging is not configured, secured MAC addresses are never aged out.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet2/0/1
```

Specifies the interface type and number, and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance (an instance of an EVC) on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used in order to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 200
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 **mac security aging time** *aging-time* **[inactivity]**

Example:

```
Device(config-if-srv)# mac security aging time 200 inactivity
```

Sets the aging time for secure addresses, in minutes. The optional **inactivity** keyword specifies that the aging out of addresses is based on inactivity of the sending hosts (as opposed to absolute aging).

Step 8 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 9 **end**

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Configuring a Sticky MAC Address

If sticky MAC addressing is configured on a secured service instance, MAC addresses that are learned dynamically on the service instance are retained during a link-down condition. Perform this task to configure sticky MAC addresses on a service instance.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface gigabitethernet2/0/1
```

Specifies the interface type and number, and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet**

Example:

```
Device(config-if)# service instance 100 ethernet
```

Creates a service instance (an instance of an EVC) on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
Device(config-if-srv)# encapsulation dot1q 100
```

Defines the matching criteria to be used to map ingress dot1q frames on an interface to the appropriate service instance.

Step 6 **bridge-domain** *bridge-id*

Example:

```
Device(config-if-srv)# bridge-domain 200
```

Binds the service instance to a bridge- domain instance where *bridge-id* is the identifier for the bridge- domain instance.

Step 7 **mac security sticky address** *mac-address*

Example:

```
Device(config-if-srv)# mac security sticky address 1111.2222.3333
```

Sets up a MAC address to be declared as a sticky MAC address on the service instance.

Step 8 **mac security**

Example:

```
Device(config-if-srv)# mac security
```

Enables MAC security on the service instance.

Step 9 **end**

Example:

```
Device(config-if-srv)# end
```

Returns to user EXEC mode.

Displaying the MAC Security Status of a Specific Service Instance

Perform this task to display the MAC security status of a service instance.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ethernet service instance id** *id* **interface** *type* *number* **mac security**

Example:

```
Device# show ethernet service instance id 100 interface gigabitethernet1/1 mac security
```

Displays the MAC security status of a specific service instance.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Displaying the Service Instances with MAC Security Enabled

Perform this task to display all the service instances with MAC security enabled.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ethernet service instance mac security**

Example:

```
Device# show ethernet service instance mac security
```

Displays all the service instances with MAC security enabled.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Displaying the Service Instances with MAC Security Enabled on a Specific Bridge Domain

Perform this task to display the service instances on a specific bridge domain that have MAC security enabled.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show bridge-domain** *id* **mac security**

Example:

```
Device# show bridge-domain 100 mac security
```

Displays all the service instances with MAC security enabled on a specific bridge domain.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Showing the MAC Addresses of All Secured Service Instances

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ethernet service instance mac security address**

Example:

```
Device# show ethernet service instance mac security address
```

Displays the secured addresses on all the service instances.

Step 3 **show mac address-table secure**

Example:

```
Device# show mac address-table secure
```

Displays the secure MAC address on the service instances.

Step 4 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Showing the MAC Addresses of a Specific Service Instance

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ethernet service instance id** *id* **interface** *type number* **mac security address**

Example:

```
Device# show ethernet service instance id 200 interface GigabitEthernet 1/0 mac security
address
```

Displays the addresses of a specific service instance.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Showing the MAC Addresses of All Service Instances on a Specific Bridge Domain

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show bridge-domain** *id* **mac security address**

Example:

```
Device# show bridge-domain 100 mac security address
```

Displays the secured addresses of all the service instances on a specified bridge domain.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Showing the MAC Security Statistics of a Specific Service Instance

This section describes how to display the MAC security statistics of a specific service instance.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ethernet service instance id** *id* **interface** *type number* **mac security statistics**

Example:

```
Device# show ethernet service instance id 100 interface gigabitethernet1/1 mac security
statistics
```

Displays the MAC security statistics of a specific service instance.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Showing the MAC Security Statistics of All Service Instances on a Specific Bridge Domain

Perform this task to display the MAC security statistics of all the service instances on a specific bridge domain.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show bridge-domain** *bridge-id* **mac security statistics**

Example:

```
Device# show bridge-domain 100 mac security statistics
```

Displays the MAC security statistics of all service instances that belong to a specific bridge domain.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Showing the Last Violation Recorded on Each Service Instance on a Specific Bridge Domain

Perform this task to display the last violation recorded on each service instance on a specific bridge domain. Service instances on which there have been no violations are excluded from the output.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show bridge-domain** *bridge-id* **mac security last violation**

Example:

```
Device# show bridge-domain 100 mac security last violation
```

Displays information about the last violation recorded on each of the service instances that belong to the bridge domain.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Clearing All Dynamically Learned Secure MAC Addresses on a Service Instance

Perform this task to clear all dynamically learned Secure MAC addresses on a service instance.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **clear ethernet service instance id** *id* **interface** *type number* **mac table**

Example:

```
Device# clear ethernet service instance id 100 interface gigabitethernet0/0/1 mac table
```

Clears all the dynamically learned Secure MAC addresses on the specified service instance.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Clearing All Dynamically Learned MAC Addresses on a Bridge Domain

Perform this task to clear all dynamically learned MAC addresses on a bridge domain.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **clear bridge-domain** *bridge-id* **mac table**

Example:

```
Device# clear bridge-domain 100 mac table
```

Clears all dynamically learned MAC addresses on the specified bridge domain.

Step 3 **end**

Example:

```
Device# end
```

Returns to user EXEC mode.

Bringing a Specific Service Instance Out of the Error-Disabled State

Perform this task to bring a specific service instance out of the error-disabled state.

Note The **clear ethernet service instance id** *id* **interface** type *number* **errdisable**command can also be used to bring a service instance out of an error disabled state. For more information about this command, see the *Cisco IOS Carrier Ethernet Command Reference*.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet2/0/1`	Specifies the interface type and number, and enters interface configuration mode.
Step 4	**service instance** *id* **ethernet** **Example:** `Device(config-if)# service instance 100 ethernet`	Creates a service instance (an instance of an EVC) on an interface and enters service instance configuration mode.
Step 5	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 100`	Defines the matching criteria to be used to map ingress dot1q frames on an interface to the appropriate service instance.
Step 6	**bridge-domain** *bridge-id* **Example:** `Device(config-if-srv)# bridge-domain 200`	Binds the service instance to a bridge-domain instance where *bridge-id* is the identifier for the bridge-domain instance.
Step 7	**mac security** **Example:** `Device(config-if-srv)# mac security`	Enables MAC security on the service instance.
Step 8	**errdisable recovery cause mac-security** *interval* **Example:** `Device(config-if-srv)# errdisable recovery cause mac-security 50`	Brings a specific service instance out of an error-disabled state and specifies a time interval to recover.
Step 9	**end** **Example:** `Device(config-if-srv)# end`	Returns to user EXEC mode.

Configuration Examples for MAC Address Limiting on Service Instances and Bridge Domains and EVC Port Channels

Example Enabling MAC Security on a Service Instance

The following example shows how to enable MAC security on a service instance:

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 3/0/1
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

Example Enabling MAC Security on an EVC Port Channel

The following example shows how to enable MAC Security on an EVC port channel:

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

Example Configuring a MAC Address Permit List

The following example shows how to configure a MAC address permit list:

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 3/0/1
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaaa
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaab
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaac
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaad
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaae
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 3/0/1
Device(config-if)# service instance 100 ethernet
```

```
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security maximum addresses 5
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaaa
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaab
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaac
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaad
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaae
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

Example Configuring a MAC Address Deny List

The following example shows how to configure a MAC address deny list:

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 3/0/1
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaaa
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaab
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaac
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaad
Device(config-if-srv)# mac security address deny a2aa.aaaa.aaae
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

Example Configuring a MAC Address Security on a Service Instance

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 3/0/1
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security maximum addresses 10
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

Example Configuring a MAC Address Violation Response

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 3/0/1
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security address permit a2aa.aaaa.aaaa
Device(config-if-srv)# mac security violation protect
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

Example Configuring MAC Address Aging

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 4/0/1
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security aging time 10
Device(config-if-srv)# mac security
Device(config-if-srv)# end
```

Example Configuring a Sticky MAC Address

```
Device> enable
Device# configure terminal
Device(config)# interface gigabitethernet 3/0/1
Device(config-if)# service instance 100 ethernet
Device(config-if-srv)# encapsulation dot1Q 100
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# mac security sticky address 1111.2222.3333
Device(config-if-srv)# mac security
```

Example Displaying the MAC Addresses on a Specific Secure Service Instance

```
Device# show ethernet service instance id 1879665131 interface gigabitethernet 0/2 mac
security address
MAC Address       Type      Rem. Age(min)
0001.0001.0001    static    100
0001.0001.0002    static    100
0001.0001.aaaa    dynamic   100
0001.0001.aaab    dynamic   100
```

```
Device# show ethernet service instance id 10 inter gig 0/0/3 mac security
address Bridge-domain 10

MAC Address       Type
0000.00ac.ef02    sticky
0000.00ac.ef03    sticky
0001.0001.aaaa    dynamic
0001.0001.aaab    dynamic
```

Example Displaying the Last Violation on a Specific Service Instance

```
Device# show ethernet service instance id 1879665131 interface gigabitethernet 0/2 mac
security last violation
At: Apr 4 06:57:25.971
Source address: ae4e.b7b5.79ae
Reason: Denied address
```

```
Device# show bridge-domain 100 mac security last violation Te0/0/3 ServInst 200
Last violation at: 15:54:25 IST Fri Jun 5 2015
Source MAC address: 0000.1111.1111
```

```
Reason: Re-learn attempt
Total violation count: 321
```

Example Displaying the MAC Security Status of a Specific Service Instance

```
Device# show ethernet service instance id 1879665131 interface Ethernet0/2 mac security
MAC Security: enabled
```

```
Device# show ethernet service instance id 100 interface te0/0/3 mac security
Bridge-domain 100
MAC Security enabled: yes
```

Example Displaying the MAC Addresses of All Secured Service Instances

```
Device# show ethernet service instance mac security address
Port                    Bridge-domain    MAC Address       Type      Rem. Age(min)
Gi1/0/0 ServInst 1      10               0001.0001.0001    static    82
Gi1/0/0 ServInst 1      10               0001.0001.0002    static    82
Gi1/0/0 ServInst 1      10               0001.0001.aaaa    dynamic   82
Gi1/0/0 ServInst 1      10               0001.0001.aaab    dynamic   82
Gi1/0/0 ServInst 2      10               0002.0002.0002    static    -
Gi1/0/0 ServInst 2      10               0002.0002.0003    static    -
Gi1/0/0 ServInst 2      10               0002.0002.0004    static    -
Gi1/0/0 ServInst 2      10               0002.0002.aaaa    dynamic   -
Gi1/0/0 ServInst 2      10               0002.0002.bbbb    dynamic   -
Gi1/0/0 ServInst 2      10               0002.0002.cccc    dynamic   -
Gi3/0/5 ServInst 10     30               0003.0003.0001    static    200
Gi3/0/5 ServInst 10     30               0003.0003.0002    static    200
```

```
Device# show ethernet service instance mac security address
Port                    Bridge-domain    MAC Address       Type
Gi0/0/3 ServInst 10     10  0000.00ac.ef02 sticky
Gi0/0/3 ServInst 10     10   0000.00ac.ef03 sticky
Gi0/0/3 ServInst 10     10   0000.00ac.ef04  dynamic
Gi0/0/3 ServInst 10     10   0000.00ac.ef05  dynamic
Gi0/0/3 ServInst 10     10   0000.00ac.ef06  sticky
Gi0/0/3 ServInst 10     10   0000.00ac.ef07  dynamic
Gi0/0/3 ServInst 10     10   0000.00ac.ef08  dynamic
Gi0/0/3 ServInst 10     10   0000.00ac.ef09  dynamic
Gi0/0/3 ServInst 10     10   0000.00ac.ef0a  dynamic
Gi0/0/3 ServInst 10     10   0000.00ac.ef0b  dynamic
```

Example Displaying the MAC Security Statistics of All Service Instances

In the following example, the numbers of allowed and actual secured addresses recorded on the service instance are displayed.

```
Device# show ethernet service instance mac security statistics
Ethernet0/0 service instance 890597333 (bridge-domain 730)
Secure addresses: 3
Address limit: 7
Ethernet0/0 service instance 1559665780 (bridge-domain 1249)
Secure addresses: 8
Address limit: 8
Ethernet0/0 service instance 1877043343 (bridge-domain 1155)
```

```
Secure addresses: 0
Address limit: 8
Ethernet0/1 service instance 127771402 (bridge-domain 730)
Secure addresses: 12
Address limit: 12
Ethernet0/1 service instance 183598286 (bridge-domain 730)
Secure addresses: 1
Address limit: 1
Ethernet0/1 service instance 433365207 (bridge-domain 1249)
Secure addresses: 0
Address limit: 1
Ethernet0/1 service instance 858688453 (bridge-domain 1328)
Secure addresses: 0
Address limit: 2
```

```
Device#  show ethernet serv instance mac security statistics
Te0/0/3 ServInst 100 (bridge-domain 100)
Current secure addresses: 1
Permitted addresses: 10
Te0/0/3 ServInst 200 (bridge-domain 100)
Current secure addresses: 0
Permitted addresses: 1
Te0/0/3 ServInst 300 (bridge-domain 100)
Current secure addresses: 0
Permitted addresses: 1
```

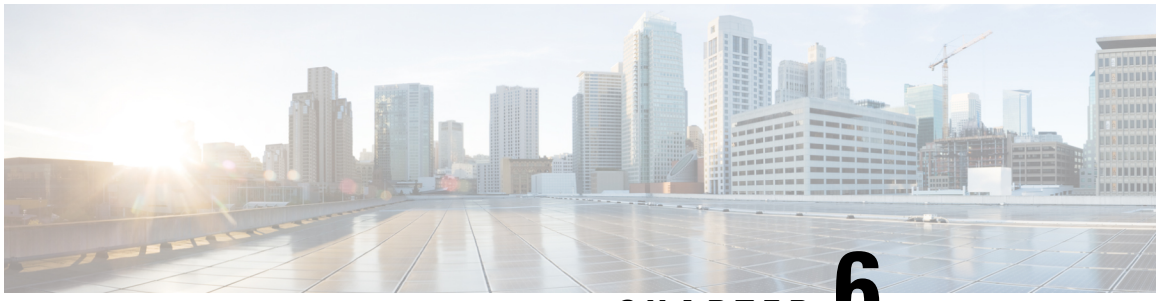
Example: Displaying the MAC Addresses on All Service Instances for a Bridge Domain

```
Router# show bridge-domain 730 mac security address
Port                  MAC Address       Type      Rem. Age(min)
Gi1/0/0 ServInst 1    0001.0001.0001    static    74
Gi1/0/0 ServInst 1    0001.0001.0002    static    74
Gi1/0/0 ServInst 1    0001.0001.aaaa    dynamic   74
Gi1/0/0 ServInst 1    0001.0001.aaab    dynamic   74
Gi1/0/0 ServInst 2    0002.0002.0002    static    -
Gi1/0/0 ServInst 2    0002.0002.0003    static    -
Gi1/0/0 ServInst 2    0002.0002.0004    static    -
Gi1/0/0 ServInst 2    0002.0002.aaaa    dynamic   -
Gi1/0/0 ServInst 2    0002.0002.bbbb    dynamic   -
Gi1/0/0 ServInst 2    0002.0002.cccc    dynamic   -
```

```
Router# show bridge-domain 10 mac security address
Port                  MAC Address       Type
Gi0/0/3 ServInst 10   0000.00ac.ef02    sticky
Gi0/0/3 ServInst 10   0000.00ac.ef03    sticky
Gi0/0/3 ServInst 10   0000.00ac.ef04    dynamic
Gi0/0/3 ServInst 10   0000.00ac.ef05    dynamic
Gi0/0/3 ServInst 10   0000.00ac.ef06    sticky
Gi0/0/3 ServInst 10   0000.00ac.ef07    dynamic
Gi0/0/3 ServInst 10   0000.00ac.ef08    dynamic
Gi0/0/3 ServInst 10   0000.00ac.ef09    dynamic
Gi0/0/3 ServInst 10   0000.00ac.ef0a    dynamic
Gi0/0/3 ServInst 10   0000.00ac.ef0b    dynamic
```

Example Displaying the Secured Service Instances for a Specific Bridge Domain

```
Router# show bridge-domain 730 mac security
Gi1/0/0 ServInst 1
MAC Security enabled: yes
Gi1/0/0 ServInst 2
MAC Security enabled: yes

Router# show bridge-domain 10 mac security
Gi0/0/3 ServInst 10
MAC Security enabled: yes
```



CHAPTER 6

Static MAC Address Support on Service Instances

The Multicast and Unicast static MAC address support on Service Instances feature supports configuration of a static MAC address on a pseudoport. Use of a static MAC address for Broadband Network Gateway (BNG) upstream traffic enables traffic forwarding while conserving MAC table resources and limiting the traffic flood by creating multicast groups.

Prerequsites for Static MAC Address Support on Service Instances

- Knowledge of both port and bridge domain limitations.
- Knowledge of service instances.

Restrictions for Static MAC Address Support on Service Instances

- Multicast static MAC addresses are not allowed in MAC address security configurations.
- Static MAC addresses are programmed only on switch processors (both active and standby).
- Static MAC configuration is *not* allowed at secure service instance.
- Static MAC addresses are programmed only on switch processors (both active and standby).
- The Static MAC address on Pseudowires is *not* supported on the Cisco ASR 900 Series Routers.

- Static MAC address configuration is *not* supported on Trunk EFP.

Information about Static MAC Address Support on Service Instances

Static MAC address configuration on service instances eliminates the need for MAC address learning, which is required for traffic forwarding. In the upstream direction, without MAC address learning, MAC address table resources can be conserved and network resources optimized.

When a bridge domain ID is either changed or deleted for a service instance, all static MAC addresses are removed.

When a service instance is deleted, all static MAC addresses on that pseudoport are removed.

Benefits of Static MAC Address Support on Service Instances

- Facilitates optimization of network resources
- Conserves MAC table resources when used for upstream traffic

Configuring a Static MAC Address on a Service Instance

Perform this task to manually configure a static MAC address on a service instance.

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Router(config)# interface Ethernet 1/0GigabitEthernet
 0/2/1
```

Configures an interface type and enters interface configuration mode.

Step 4 **service instance** *id* **ethernet** [*evc-id*]

Example:

```
Router(config-if)# service instance 1 ethernet
```

Configures an Ethernet service instance on an interface and enters service instance configuration mode.

Step 5 **encapsulation dot1q** *vlan-id* [**,** *vlan-id*[**-** *vlan-id*]] [**native**]

Example:

```
Router(config-if-srv)# encapsulation dot1q 100
```

Enables IEEE 802.1Q encapsulation of traffic on a specified subinterface in a VLAN.

Step 6 **bridge-domain** *bridge-id* [**split-horizon**[**group** *group-id*]]

Example:

```
Router(config-if-srv)# bridge-domain 100
```

Binds a service instance to a bridge domain instance.

Step 7 **mac static address** *mac-addr* [**auto-learn**] [**disable-snooping**]

Example:

```
Router(config-if-srv)# mac static address 0000.bbbb.cccc
```

Configures a static MAC address.

Step 8 **exit**

Example:

```
Router(config-if-srv)# exit
```

Returns the CLI to privileged EXEC mode.

Example for Configuring a Static MAC Address on a Service Instance

```
Router> enable
Router# configure terminal
Router(config)# interface GigabitEthernet 0/2/1
Router(config-if)# service instance 1 ethernet
Router(config-if-srv)# encapsulation dot1q 100
Router(config-if-srv)# bridge-domain 100
Router(config-if-srv)# mac static address 0000.bbbb.cccc
Router(config-if-srv)# exit
```

Verifying Configured Static MAC Addresses on a Service Instance

Use one or more of the following commands to verify the configured static MAC address on a service instance:

- **show bridge-domain**
- **show mac address-table**

Example: Verifying Configured Static MAC Addresses on a Service Instance

show bridge-domain

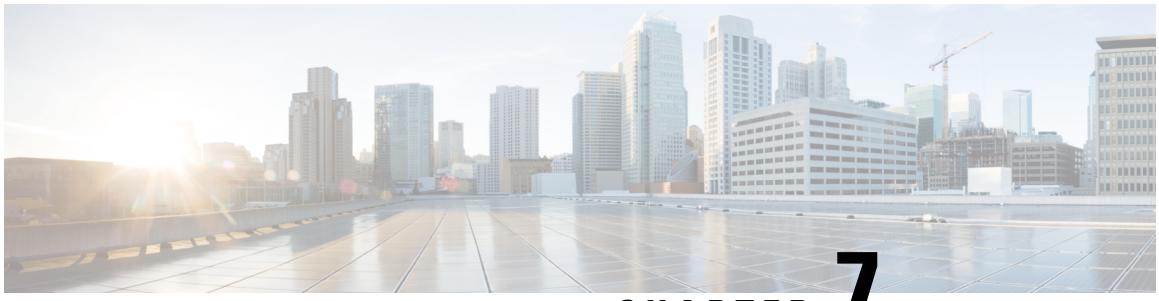
The sample output for the **show bridge-domain** command:

```
Router# show bridge-domain 10 mac static address

Bridge-Domain ID : 10
Static MAC count : System : 1, bridge-domain : 1

Port                                    Address          Action
Gi0/3/7 ServInst 10                     aaa1.123c.bc32
```

show mac address-table

The sample output for the **show mac address-table** command:

```
Router# show mac address-table bdomain 10

   Nile Mac Address Entries

   BD    mac addr        type     ports

-------------------------------------------------------------------------------------------------

   10    aaa1.123c.bc32  STATIC   Gi0/3/7.Efp10
```



CHAPTER 7

MAC Limiting

This document describes how to configure MAC limiting.

Information About Global MAC Address Limiting on Bridge Domain

Table 12: Feature History

Feature Name	Release Information	Description
Mac Address Limiting Per Bridge Domain	Cisco IOS XE Cupertino 17.8.1	This feature restricts the number of MAC addresses that the router learns in a bridge-domain on an EFP or trunk EFP to a specified number. Use the feature to enable warning and limit actions when a violation occurs.

MAC address limiting per bridge-domain restricts the number of MAC addresses that the router learns in a bridge-domain on an EFP or trunk EFP to a specified number.

Note For the RSP1 and RSP2 modules, the local connect feature is not supported on the Cisco router. However, to simulate a local connect scenario, configure the connecting EFPs on the same bridge domain and disable the mac-learning on the bridge domain by setting the MAC limit to 0. Use the **mac-address-table limit bdomain** *num* **maximum** *0* **action limit** command to disable mac-learning on bridge-domain.

When the total number of MAC addresses (dynamic MAC addresses) in a bridge-domain exceeds the maximum number, then the router takes a violation action. The router either restricts further learning on bridge-domain by itself with a syslog or just intimate the user through a syslog to take further action.

You can enable the following actions when violation occurs:

- Warning—The violation is logged as a syslog message and no further action is taken. There is one syslog message received, when the MAC count exceeds the configured limit (exceed notification) and no more syslog messages are received for the bridge-domain (bdomain) unless the violation is no longer valid (drop notification). When you select the warning action, the further learning of new MAC addresses and forwarding of traffic continue to happen irrespective of violation.
- Limit—When the Limit option is selected as an action for violation, the MAC learning on the bdomain is disabled when violation occurs. No new MAC addresses are learnt on the bdomain until the recovery mechanism gets started. Even though new MAC addresses are not learned but frames are still flooded in the system. If user needs to stop flooding, then a sub action flood can also be used along with limit action.

Note The threshold value must be 80% of the maximum value configured for the recovery mechanism.

- Flood—The flood sub action allows the user to disable unknown unicast flooding on a given bdomain. This flood sub action is initiated only when the limit action is configured and violation has occurred. Unknown unicast flooding is disabled only for the interval necessary to limit the entries. Using this option, improves the performance and the flooding is re-enabled when the total number of MAC entries are dropped below the threshold value.
- Shutdown—When the shutdown action is selected, a syslog message is generated and the particular bdomain on which violation occurred is disabled. Hence, all the learning and forwarding of traffic are stopped on the bdomain. The bdomain remain in such state until the feature is explicitly disabled through CLI.

Note **Warning** is the default action when no action is configured.

Note The functionality of automatic error recovery is *not* supported on the Cisco ASR 900 RSP2 module.

For the limit and warning actions, the recovery mechanism is initiated when the total MAC limit count drops to equal or below a threshold value. The threshold value is dependent on the maximum limit configured on bridge domain (the threshold value is 80% of the limit value). The recovery mechanism reverts the action taken during violation. For example, if the MAC address learning is disabled as a violation action, then it will be re-enabled.

If no maximum value or action option in specified through the **mac address-table limit bdomain id maximum num action** command, then the default action (warning) and a default maximum value of 500 is configured.

Note For a MAC limit of 0 with the action limit, limit flood, the violation action occurs when the user configures it irrespective of MAC address learning on the bridge domain. The recovery mechanism is to disable the feature through the **no mac address-table limit bdomain id** command.

Restrictions and Usage Guidelines for the RSP1 and RSP2 Modules

MAC limiting is supported on the following interface types:

- You can apply MAC limiting only to bridge-domains.
- MAC limiting is supported for dynamic MAC addresses.

Restrictions for MAC Limiting for RSP3 Module

- Bridge domain MAC limit and EFP MAC Security are not supported together on a bridge domain.
- The change in split horizon group configuration is not supported on the bridge domain if the MAC limit is already configured on that domain.
- A maximum number of four unique MAC limit values can be configured at any time. Many bridge domains can use the same values but it cannot be shared with a bridge domain interface. If the bridge domain interface is added to the existing bridge domain MAC limit configuration, then the configuration should be removed and added again.
- On a Trunk EFP, if the violation is noticed on atleast one of the bridge domains, then the violation action applies to the whole Trunk EFP. If one bridge domain has the action limit, the limit flood or the shutdown action exceeds, then the whole Trunk EFP's MAC learning is disabled.
- The allowed MAC limit range is from 0 through 0xFFFD.
- The MAC limit on the bridge domain interface needs to be configured to a value higher than the actual maximum limit value that is expected. This is because an internal static MAC is added if the bridge domain interface has an IP configured or the corresponding bridge domain is a part of L2VPN. This will be taken into account for MAC limit.
- The action warning is applied based on the software learning and a delay of approximately 1 minute is observed while generating syslog on a normal bridge domain.
- The delay in the drop notification in based on the software again and the delay is approximately 1 minute for the syslog generation.
- In case of MAC limit 0, static MACs are allowed to be added even after the limit exceeds, only if the bridge domain is UP.

Configuring MAC Limiting

Procedure

Step 1 **configure terminal**

Enter global configuration mode.

Step 2 **mac-address-table limit** *bdomain id maximum num action {warning | limit | shutdown} [flood]*

Sets the specific limit and any optional actions to be imposed at the bridge-domain level.

The default **maximum** value is 500.

Step 3 **end**

Return to privileged EXEC mode.

Step 4 **show mac-address-table limit bdomain** *bdomain id*

Displays the information about the MAC-address table.

Step 5 **copy running-config startup-config**

(Optional) Save your entries in the configuration file.

Example of Enabling Per-Bridge-Domain MAC Limiting

This example shows how to enable per-bridge-domain MAC limiting.

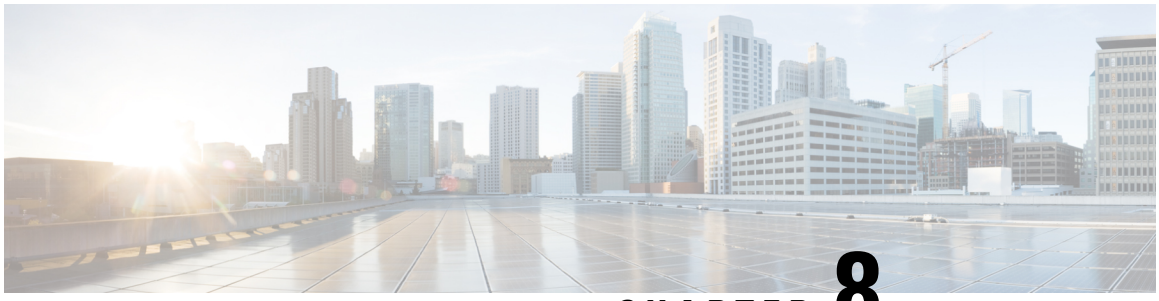
```
Router# enable
Router# configure terminal
Router(config)# mac-address-table limit bdomain 10 maximum 100 action limit flood
Router(config)# end
```

Verifying the MAC Limiting on Bridge Domain

Use the **show mac address-table limit** command to verify the information related to configured MAC limit per bridge domain.

This example shows how to display the information related to configured MAC limit per bridge domain.

```
Router#show mac address-table limit bdomain 10
  bdomain       action      flood       maximum     Total entries     Current state
-------------+----------+------------+------------+---------------+---------------
   10           limit      Disable        100            0             Within Limit
```



CHAPTER 8

WAN MACsec and MKA Support Enhancements

The WAN MACsec and MACsec Key Agreement protocol (MKA) features introduce MACsec support on WAN, and uplink support and pre-shared key support for the MKA.

Table 13: Feature History

Feature Name	Release	Description
MACsec support with PTP for 1GE NCS4200-1T16G-PS Interface Module	Cisco IOS XE Dublin 17.12.1	You can now configure MACsec support with Precision Time Protocol (PTP) packets for mitigating security vulnerabilities on a router.
MACsec Support with SyncE for 1GE and 10GE NCS4200-1T16G-PS Interface Module	Cisco IOS XE Dublin 17.10.1	You can now configure MACSec encryption on Synchronized Ethernet (SyncE) interfaces that send and receive Ethernet Synchronization Message Channel (ESMC) packets. The MACSec header is added to the ESMC packets to secure data on the physical media. Also, MACSec encryption prevents the higher-layer protocols' traffic from being compromised.
802.1AE WAN MACsec Enhancement for 1GE and 10GE NCS4200-1T16G-PS	Cisco IOS XE Cupertino 17.8.1	The 802.1AE WAN MACsec supports 10GE physical layer (PHY) interfaces for NCS4200-1T16G-PS interface module. From this release, full HA, Power on Self Test (POST) and double tag support are available on NCS4200-1T16G-PS interface module. The following new command is introduced: **show macsec post**
802.1AE WAN MACsec for 1GE and 10GE NCS4200-1T16G-PS	Cisco IOS XE Bengaluru 17.6.1	The WAN MACsec and MKA feature introduce MACsec support on WAN and uplink support and pre-shared key support for the MACsec Key Agreement protocol (MKA). The WAN MACsec supports 1GE and 10GE interfaces for NCS4200-1T16G-PS interface module.

Feature Name	Release	Description
MAC Security	Cisco IOS XE Bengaluru 17.5.1	The MACsec and MACsec Key Agreement protocol (MKA) features are introduced on the router interface (main interface) with pre-shared key support for the MKA. This feature is supported on the Cisco RSP3 module.

Prerequisites for WAN MACsec and MKA Support Enhancements

- Layer 2 transparent Ethernet Services must be present.
- The service provider network must provide a MACsec Layer 2 Control Protocol transparency such as, Extensible Authentication Protocol over LAN (EAPoL).

Restrictions for WAN MACsec and MKA Support Enhancements

- MACSec encryption of SyncE packets is not supported for those that are tagged with EFP service instances.
- MACsec with PTP is not supported until Cisco IOS XE Dublin 17.11.1.

  Starting with Cisco IOS XE Dublin 17.12.1, the router supports MACsec and PTP with G8275.1 profile only in different 1G interfaces on NCS4200-1T16G-PS Interface Module. You cannot configure different 1G on the same interface.
- For Ethernet service instance on the customer edge (CE) copper link to work, the Ethernet service instance, where xconnect is present in the provider edge (PE) must have the **remote link failure notification** command that is configured on the link.

  Following is a sample configuration.

```
interface TenGigabitEthernet x/y/z
  description ### X connect 1 <name> ###
  mtu 9216
  no ip address
  negotiation auto
  no keepalive
  service instance 10 ethernet
  encapsulation default
  l2protocol tunnel cdp stp vtp pagp dot1x lldp lacp udld loam esmc elmi ptppd mmrp
mvrp
  xconnect x.x.x.x 10 encapsulation mpls
  remote link failure notification
```

- On Cisco NCS 4206 and 4216 Series Aggregation Services Routers, MACsec doesn't support AAA accounting.
- MACsec is supported up to line rate on each interface. However, the forwarding capability may be limited by the maximum system forwarding capability.
- MACsec is supported on PE label on Cisco NCS 4206 and 4216 routers of NCS4200-1T16G-PS interface module.
- MACsec configuration on Ether Channel (Link bundling) isn't supported.
- Any interface that is configured with MACsec can't be part of Ether Channel.
- If the MKA session is torn down because of key unwrap failure, re-configure the pre-shared key-based MKA session using MACsec configuration commands on the respective interfaces to bring up the MKA session.
- MACsec-configured on physical interface with Ethernet Virtual Circuits (EVC) isn't supported. The EAPoL frames get dropped in such cases.
- On Cisco NCS 4206 and 4216 Series Aggregation Services Routers, the following table lists the Gigabit Ethernet interface and the maximum number of peers that are supported per interface.

Gigabit Ethernet Interface	Peers per Interface
1G	8
10G	32

- When `macsec dot1q-in-clear` is enabled, the native VLAN isn't supported.

Information About WAN MACsec and MKA Support Enhancements

MACsec and MKA Overview

MACsec is an IEEE 802.1AE standards based Layer 2 hop-by-hop encryption that provides data confidentiality and integrity for media access independent protocols.

MACsec, provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. The MACsec Key Agreement (MKA) protocol provides the required session keys and manages the required encryption keys. Only host facing links (links between network access devices and endpoint devices such as a PC or IP phone) can be secured using MACsec.

The 802.1AE encryption with MACsec Key Agreement (MKA) is supported on downlink ports for encryption between the routers or switches and host devices.

MACsec encrypts the entire data except for the Source and Destination MAC addresses of an Ethernet packet.

To provide MACsec services over the LAN or Metro Ethernet, service providers offer Layer 2 transparent services such as E-Line or E-LAN using various transport layer protocols such as Ethernet over Multiprotocol Label Switching (EoMPLS) and L2TPv3.

The EAP framework implements MKA as a newly defined EAP-over-LAN (EAPOL) packet. EAP authentication produces a master session key (MSK) shared by both partners in the data exchange. Entering the EAP session ID generates a secure connectivity association key name (CKN). Because the switch is the authenticator, it is also the key server, generating a random 128-bit secure association key (SAK), which it sends it to the client partner. The client is never a key server and can only interact with a single MKA entity, the key server. After key derivation and generation, the switch sends periodic transports to the partner at a default interval of 2 seconds.

The packet body in an EAP-over-LAN (EAPOL) Protocol Data Unit (PDU) is referred to as a MACsec Key Agreement PDU (MKPDU). When no MKPDU is received from a participants after 3 hearbeats (each hearbeat is of 2 seconds), peers are deleted from the live peer list For example, if a client disconnects, the participant on the switch continues to operate MKA until 3 heartbeats have elapsed after the last MKPDU is received from the client.

The MKA feature support provides tunneling information such as VLAN tag (802.1Q tag) in the clear so that the service provider can provide service multiplexing such that multiple point to point or multipoint services can co-exist on a single physical interface and differentiated based on the now visible VLAN ID.

In addition to service multiplexing, VLAN tag in the clear also enables service providers to provide quality of service (QoS) to the encrypted Ethernet packet across the SP network based on the 802.1P (CoS) field that is now visible as part of the 802.1Q tag.

Starting with Cisco IOS XE Release 17.8.1, full HA, Power on Self Test (POST) and double tag support are available on NCS4200-1T16G-PS interface module. The POST tests the hardware to verify that all components of the device are operational and present. In the double tagging (qinq tag) method, the VLAN tag simply adds another tag to the tagged packets that enter the network. The purpose is to expand the VLAN space by tagging the tagged packets, thus producing a “double-tagged” frame. The expanded VLAN space allows the service provider to provide certain services, such as Internet access on specific VLANs for specific customers, and yet still allows the service provider to provide other types of services for their other customers on other VLANs. The Single Sign-On (SSO) and IM Online Insertion and Removal (OIR) triggers preserve MKA sessions.

Starting from release 17.12.1, Cisco IOS XE supports MACSec with PTP with G8275.1 profile on different 1G interfaces within the same NCS4200-1T16G-PS Interface Module. It does not support MACsec with PTP with G8275.1 on the same interface.

This limitation is applicable only when MACsec is configured on 1G port and PTP on 1G port with G8275.1 profile.

Table 14: MACsec and PTP Support on NCS4200-1T16G-PS

MACsec Support with PTP	Same Interface	Different Interface
1G—MACsec and PTP with G8275.1 profile	Not supported	Supported
1G—MACsec and PTP with G8275.2 and G8265.1 profiles	Not supported	Not supported

Benefits of WAN MACsec and MKA Support Enhancements

- Support for both Copper SFP 10M and 100M speed for MACsec.
- Support for point-to-point (P2P) deployment models.

- Support for 128 bit and 256 bit Advanced Encryption Standard–Galois Counter Mode (AES-GCM) encryption for data packets.
- Support for 128 bit and 256 bit Advanced Encryption Standard-Cipher-based Message Authentication Code (AEC-CMAC) encryption for control packets.
- Support for VLAN tag in the clear option to enable Carrier Ethernet Service Multiplexing.
- Support for coexisting of MACsec and Non-MACsec subinterfaces.
- Support for configurable Extensible Authentication Protocol over LAN (EAPOL) destination address.
- Support for configurable option to change the EAPOL Ethernet type.
- Support for configurable replay protection window size to accommodate packet reordering in the service provider network.
- Support for MACsec stateless switchover. During route processor (RP) switchover on dual RP setup, there is a teardown of existing MACsec session and the session is re-negotiated/reinitiated automatically (stateless switchover). During this process, some traffic drop might occur for a few seconds.

Best Practices for Implementing WAN MACsec and MKA Support Enhancements

- Ensure basic Layer 2 Ethernet connectivity is established and verified before attempting to enable MACsec. Basic ping between the customer edge devices must work.
- When you are configuring MACsec for the first time, ensure that you have out of band connectivity to the remote site to avoid locking yourself out after enabling MACsec, if the session fails to establish.
- We recommend that you configure an interface MTU, adjusting it for MACsec overhead, for example, 32 bytes. Although MACsec encryption and decryption occurs at the physical level and MTU size does not effect the source or destination router, it may effect the intermediate service provider router. Configuring an MTU value at the interface allows for MTU negotiation that includes MACsec overhead.

MKA Policy Inheritance

On WAN routers, MKA policy is inherited and also it has a default value. When a new session is started, the following rules apply:

- If an MKA policy is configured on a subinterface, it will be applied when an MKA session is started.
- If a MKA policy is not configured on a subinterface or physical interface, default policy is applied at session start.

Key Lifetime and Hitless Key Rollover

A MACsec key chain can have multiple pre-shared keys (PSK) each configured with a key ID and an optional lifetime. A key lifetime specifies when the key expires. In the absence of a lifetime configuration, the default lifetime is unlimited. When a lifetime is configured, MKA rolls over to the next configured pre-shared key in the key chain after the lifetime is expired. Time zone of the key can be local or UTC. Default time zone is UTC.

Use the **key chain** *name* **macsec** command to configure the MACsec key chain.

The key rolls over to the next key within the same key chain by configuring a second key in the key chain and configuring a lifetime for the first key. When the lifetime of the first key expires, it automatically rolls over to the next key in the list. If the same key is configured on both sides of the link at the same time, then the key rollover is hitless, that is, key rolls over without traffic interruption.

Note The lifetime of the keys needs to be overlapped in order to achieve hitless key rollover.

Encryption Algorithms for Protocol Packets

Cryptographic Algorithm selection for MKA control protocol packets encryption is as follows:

- Cryptographic Algorithm to encrypt MKA control protocol packets is configured as part of the key chain. There can be only one cryptographic algorithm configured per key chain.
- A key server uses the configured MKA cryptographic algorithm from the key chain that is used.
- All nonkey servers must use the same cryptographic algorithm as the key server.

If an MKA cryptographic algorithm is not configured, a default cryptographic algorithm of AES-CMAC-128 (Cipher-based Message Authentication Code with 128-bit Advanced Encryption Standard) is used.

Encryption algorithm for Data packets:

```
mka policy p1
macsec-cipher-suite [gcm-aes-128 | gcm-aes-256
```

Encryption algorithm for MKA Control packets

```
key chain <name> macsec
key 01
key-string <Hex string>
cryptographic-algorithm [aes-256-cmac  | aes-128-cmac]
```

Replay Protection Window Size

Replay protection is a feature provided by MACsec to counter replay attacks. Each encrypted packet is assigned a unique sequence number and the sequence is verified at the remote end. Frames transmitted through a Metro Ethernet service provider network are highly susceptible to reordering due to prioritization and load balancing mechanisms used within the network.

A replay window is necessary to support use of MACsec over provider networks that reorder frames. Frames within the window can be received out of order, but are not replay protected. The default window size is set to 64. Use the **macsec replay-protection window-size** command to change the replay window size. The range for window size is 0 to 4294967295.

The replay protection window may be set to zero to enforce strict reception ordering and replay protection.

WAN MACsec on Interface Module

The interface module NCS4200-1T16G-PS is supported on Cisco NCS 4206 and 4216 routers on 1GE and 10GE interfaces.

OIR Support

When the interface module is operationally inserted or removed (OIR), the configuration associated with that interface is preserved such that if the interface is ever reinserted into the system it appears with the same configuration. However, on Cisco NCS routers the following limitations apply for MACsec and MKA sessions:

- In some scale scenarios, after OIR MKA/MACsec session may be lost.
- MKA/MACsec session may be reestablished after OIR.

How to Configure WAN MACsec and MKA Support Enhancements

Configuring MKA

The MACsec Key Agreement (MKA) enables configuration and control of keying parameters. Perform the following task to configure MKA.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **mka policy** *policy-name*

Example:

```
Device(config)# mka policy MKAPolicy
```

Configures an MKA policy.

Step 4 **include-icv-indicator**

Example:

```
Device(config-mka-policy)# include-icv-indicator
```

(Optional) Include ICV indicator in MKPDU.

Step 5 **key-server priority** *key-server-priority*

Example:

```
Device(config-mka-policy)# key-server priority 200
```

(Optional) Configures MKA key server priority.

Step 6 **macsec-cipher-suite** {**gcm-aes-128** | **gcm-aes-256**}

Example:

```
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128 gcm-aes-256
```

(Optional) Configures cipher suite(s) for secure association key (SAK) derivation. Each of the cipher suite options can be repeated only once, but they can be used in any order.

Step 7 **sak-rekey interval** *interval*

Example:

```
Device(config-mka-policy)# sak-rekey interval 30
```

(Optional) Sets the SAK rekey interval (in seconds). The range is from 30 to 65535, and the default value is 0. The SAK rekey timer does not start by default until it is configured.

- To stop the SAK rekey timer, use the **no sak-rekey interval** command under the defined MKA policy.

Step 8 **confidentiality-offset 30**

Example:

```
Device(config-mka-policy)# confidentiality-offset 30
```

(Optional) Configures confidentiality offset for MACsec operation.

Step 9 **use-updated-eth-header**

(Optional) Enables interoperability with Cisco routers, and any port on a device that includes the updated ethernet header in MKPDUs for ICV calculation. This updated ethernet header is non-standard. Enabling this option ensures that an MKA session between the devices can be set up.

Before this fix, devices such as Cisco routers did not include the ethernet header for ICV calculation. With this command, an MKA session can be successfully established between your devices.

Step 10 **end**

Example:

```
Device(config-mka-policy)# end
```

Returns to privileged EXEC mode.

Example

You can use the **show mka policy** command to verify the configuration. Here's a sample output of the **show** command. If you do not want to include icv-indicator in MKPDUs, use the **no include-icv-indicator** command in the MKA policy.

```
MKA Policy Summary...

Codes : CO - Confidentiality Offset, ICVIND - Include ICV-Indicator,
        SAKR OLPL - SAK-Rekey On-Live-Peer-Loss,
        DP - Delay Protect, KS Prio - Key Server Priority

Policy            KS   DP    CO SAKR  ICVIND Cipher          Interfaces
Name              Prio          OLPL         Suite(s)        Applied
============================================================================
*DEFAULT POLICY*  0    FALSE 0  FALSE TRUE   GCM-AES-128     N/A
                                              GCM-AES-256

confid50          0    FALSE 50 FALSE TRUE   GCM-AES-128
                                              GCM-AES-256

icv               0    FALSE 0  FALSE TRUE   GCM-AES-128     Te3/0/9
                                              GCM-AES-256

k10               0    FALSE 0  FALSE TRUE   GCM-AES-128
                                              GCM-AES-256
```

Configuring MKA Pre-shared Key

Perform the following task to configure MACsec Key Agreement (MKA) pre-shared key.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **key chain** *key-chain-name* [**macsec**]

Example:

```
Device(config)# Key chain keychain1 macsec
```

Configures a key chain and enters keychain configuration mode.

Step 4 **key** *hex-string*

Example:

```
Device(config-keychain)# key 9ABCD
```

Configures a key and enters keychain key configuration mode.

Step 5 **cryptographic-algorithm** {**aes-128-cmac** | **aes-256-cmac**}

Example:

```
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
```

Set cryptographic authentication algorithm.

Step 6 **key-string** {[**0** | **6**] *pwd-string* | **7** | *pwd-string*}

Example:

```
Device(config-keychain-key)# key-string 0 pwd
```

Sets the password for a key string.

Step 7 **lifetime local** {{*day month year* **duration** *seconds*}

Example:

```
Device(config-keychain-key)# lifetime local 16:00:00 Nov 9 2014 duration 6000
```

Sets the lifetime for a key string. If you want infinite, then skip the configuration as the default value is infinite only.

The range you can specify for the duration is 1–864000 seconds.

Step 8 **end**

Example:

```
Device(config-keychain-key)# end
```

Returns to privileged EXEC mode.

Configuring MACsec and MKA on Interfaces

Perform the following task configure MACsec and MKA on an interface.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface GigabitEthernet 0/0/0
```

Enters interface configuration mode.

Step 4 **mka policy** *policy-name*

Example:

```
Device(config-if)# mka policy MKAPolicy
```

Configures an MKA policy

Step 5 **mka pre-shared-key key-chain** *key-chain-name*

Example:

```
Device(config-if)# mka pre-shared-key key-chain key-chain-name
```

Configures an MKA pre-shared-key key-chain keychain1

Note The MKA Pre-shared key can be configured on either physical interface or subinterfaces and not on both physical and subinterfaces.

Step 6 **macsec**

Example:

```
Device(config-if)# macsec
```

Configures MACsec for the EAPOL frame ethernet type.

Step 7 **macsec replay-protection window-size**

Example:

```
Device(config-if)# macsec replay-protection window-size 10
```

Sets the MACsec window size for replay protection.

Step 8 **end**

Example:

```
Device(config-if)# end
```

Returns to privileged EXEC mode.

Configure WAN MACsec for QINQ Clear Tag

To configure WAN MACsec for QINQ clear tag:

```
configure terminal
!
```

```
key chain k10 macsec
key 10  cryptographic-algorithm aes-256-cmac
key-string  12345678901234567890123456789012123456789012345678901234567890123456789012!
mka policy p2
key-server priority 223
macsec-cipher-suite gcm-aes-256
!
interface TengigabitEthernet0/3/16
no ip address
ip mtu 1468
macsec dot1q-in-clear 2
service instance 200 ethernet
encapsulation dot1q 200 second-dot1q 201
rewrite ingress tag pop 2 symmetric
mka pre-shared-key key-chain k10
mka policy p2
macsec
bridge-domain 200
end
```

Verify POST Configuration

To verify the macsec Power on Self Test (POST) configuration, use the **show macsec post** command in privileged EXEC mode.

To verify the macsec Power on Self Test (POST) configuration:

```
 MACsec Capable Interface                          POST Result
------------------------------------------------------------------
 GigabitEthernet0/1/0                                 PASS
 GigabitEthernet0/1/2                                 PASS
 GigabitEthernet0/1/4                                 PASS
 GigabitEthernet0/1/6                                 PASS
 GigabitEthernet0/1/8                                 NONE
 GigabitEthernet0/1/10                               NONE
 GigabitEthernet0/1/12                               NONE
 GigabitEthernet0/1/14                               NONE
 TenGigabitEthernet0/1/16                           PASS
 GigabitEthernet0/2/0                                  PASS
GigabitEthernet0/2/2                                 PASS
 GigabitEthernet0/2/4                                  PASS
 GigabitEthernet0/2/6                                 NONE
```

MKA-PSK: CKN Behavior Change

For MKA-PSK sessions, instead of fixed 32 bytes, the Connectivity Association Key name (CKN) uses the same string as the CKN, which is configured as the hex-string for the key.

Example Configuration:

```
configure terminal
key chain abc macsec
  key 11
    cryptographic-algorithm aes-128-cmac
    key-string 12345678901234567890123456789013
    lifetime local 12:21:00 Sep 9 2015 infinite

 end
```

For the previous example, following is the **show** command output for the **show mka session** command:

```
Device# show mka session

           Total MKA Sessions....... 1
           Secured Sessions... 1
           Pending Sessions... 0

=============================================================================================

           Interface      Local-TxSCI         Policy-Name      Inherited         Key-Server

           Port-ID        Peer-RxSCI          MACsec-Peers     Status            CKN


=============================================================================================

           Et0/0          aabb.cc00.6600/0002     icv             NO                NO

           2              aabb.cc00.6500/0002 1                Secured           11    *Note
 that the CKN key-string is exactly the same that has been configured for the key as
hex-string.*
```

In the case of interoperability between two images -- one having the CKN behavior change, and one without the CKN behavior change, then the hex-string for the key must be a 64-character hex-string with zero padded to work on a device that has an image with the CKN behavior change. See the following example:

Configuration without CKN key-string behavior change:

```
config t
  key chain abc macsec
   key 11
   cryptographic-algorithm aes-128-cmac
   key-string 12345678901234567890123456789013
   lifetime local 12:21:00 Sep 9 2015 1-864000
```

Configuration with CKN key-string behavior change:

```
config t
  key chain abc macsec
   key 11000000000000000000000000000000000000000000000000000000000000000000
   cryptographic-algorithm aes-128-cmac
   key-string 12345678901234567890123456789013
   lifetime local 12:21:00 Sep 9 2015 1-864000
```

Configuring an Option to Change the EAPoL Ethernet Type

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type number*

Example:

```
Device(config)# interface GigabitEthernet 0/0/1
```

Enters interface configuration mode.

Step 4 **macsec** *eth-type*

Example:

```
Device(config)# macsec
```

Configures the Ethernet type for the Extensible Authentication Protocol (EAPoL) Frame.

Step 5 **eapol** *eth-type*

Example:

```
Device(config-if)# eapol eth-type 0xB860
```

Configures an ethernet type (Hexadecimal) for the EAPoL Frame on the interface.

Step 6 **exit**

Example:

```
Device(config-if)# exit
```

Exits interface configuration mode and returns to global configuration mode.

Configuration Examples for MACsec and MKA

Example: Point-to-point, CE to CE Connectivity Using EPL Service

The following is the sample configuration for point-to-point, Customer Edge to Customer Edge connectivity using Ethernet Private Line (EPL) using port-based service.

```
!Customer Edge 1
key chain k1 macsec*
  key 01
  key-string 12345678901234567890123456789012
interface GigabitEthernet0/0/4
  ip address 10.3.1.1 255.255.255.0
  mka pre-shared-key key-chain k1*
  macsec*

!Customer Edge 2
key chain k1 macsec*
```

```
  key 01
  key-string 12345678901234567890123456789012
interface GigabitEthernet0/0/4
  ip address 10.3.1.2 255.255.255.0
  mka pre-shared-key key-chain k1*
  macsec*
```

Example: Point-to-point, CE to CE Connectivity Using EVPL Service

Example: Performing Maintenance Tasks Without Impacting Traffic

The following are sample configurations of performance maintenance tasks that do not impact traffic:

Changing a Pre-Shared Key (CAK Rollover)

The following is sample configuration for changing a pre-shared key:

Note Keys can be configured to automatically roll over to the next key by configuring a lifetime on both routers.

```
!From
key chain k1 macsec*
  key 01
  key-string 12345678901234567890123456789012

!To
key chain k1 macsec*
  key 01
   key-string 12345678901234567890123456789012
   lifetime local 10:30:00 Oct 30 2014 11:30:00 Oct 30 2014
  key 02
   key-string 11145678901234567890123456789012
```

Changing a Key Chain (Keychain Rollover)

The following is the sample configuration for changing a key chain—Keychain Rollover

```
! From
key chain k1 macsec*
  key 01
  key-string 12345678901234567890123456789012
interface TenGigabitEthernet0/0/0
 mka pre-shared-key key-chain k1

! To
key chain k1 macsec
  key 01
   key-string 12345678901234567890123456789012
key chain k2 macsec
  key 02
   key-string abcdef0987654321abcdef0987654321
interface TenGigabitEthernet0/0/0
 mka pre-shared-key key-chain k2
```

Note The defined key ID, under any key chain, should be a unique value on the device.

A router can be become a key server by configuring a lower priority than other peer routers that participate in the same session. Configure a key server priority so that the key server selection is deterministic. For example, in a Hub and Spoke scenario, the most ideal place for a key server is the Hub site router.

```
!Hub Site (Key Server):
mka policy p1
key-server priority 0
!0 is the default.

interface TenGigabitEthernet0/0/0
 mka pre-shared-key key-chain k1
mka policy p1

!Spoke Sites (non-Key Servers):
mka policy p1
key-server priority 1

interface TenGigabitEthernet0/0/0
 mka pre-shared-key key-chain k1
mka policy p1
```

The following is sample configuration for changing Cipher Suite to encrypt data traffic:

```
mka policy p1
 macsec-cipher-suite gcm-aes-128
interface GigabitEthernet0/0/1
 mka policy p1

!Alternate configuration

mka policy p1
 macsec-cipher-suite gcm-aes-256
interface GigabitEthernet0/0/1
 mka policy p1
```

```
key chain k3 macsec
 key 01
   key-string abcdef0987654321abcdef0987654321
   cryptographic-algorithm aes-128-cmac
interface TenGigabitEthernet0/0/0
 mka pre-shared-key key-chain k3

!Alternate configuration:

key chain k3 macsec
 key 01
   key-string abcdef0987654321abcdef0987654321
   cryptographic-algorithm aes-256-cmac
interface TenGigabitEthernet0/0/0
 mka pre-shared-key key-chain k3
```

EAPOL Destination MAC address can be changed from physical interface configuration mode or subinterface configuration mode and is automatically inherited by the subinterfaces, if configured at the physical interface level. To override the inherited value, configure the MAC address at the subinterface mode. Default EAPOL destination MAC address is 01:80:c2:00:00:03.

```
interface TenGigabitEthernet0/0/0
eapol destination-address <H.H.H>

!Alternate configuration

interface TenGigabitEthernet0/0/0
 bridge-group-address

!Alternate configuration

interface TenGigabitEthernet0/0/0
 lldp-multicast-address>

mka policy p1
 confidentiality-offset 30
interface GigabitEthernet0/0/1
 mka policy p1
```

Example: Performing Maintenance Tasks—Traffic Impacting

Changing a Replay Protection Window Size

Replay protection window can be changed from physical interface configuration mode or subinterface configuration mode and is automatically inherited by the subinterfaces if configured at the physical interface level. If you need to override the inherited value, configure it at the subinterface mode. However, MACsec on subinterfaces is not supported. The default replay protection window size is 64.

```
interface TenGigabitEthernet0/0/0
macsec replay-protection window-size 10

interface TenGigabitEthernet0/0/0
macsec replay-protection window-size 5
```

Enabling or Disabling VLAN (dot1q) Tag in the Clear Option

The **macsec dot1q-in-clear** command can only be configured on physical interface, and the setting is automatically inherited by the subinterfaces.

```
interface GigabitEthernet0/0/1
macsec dot1q-in-clear 1
```

The **macsec access-control** [**must-secure** | **should-secure**] command can only be configured on physical interface, and the setting is automatically inherited by the subinterfaces.

```
interface GigabitEthernet0/0/1
macsec access-control must-secure/should-secure
```

Example: Configuring SyncE and MACSec

The process of enabling MACSec encryption for Synchronized Ethernet (SyncE) includes configurations to be performed in the global and interface configuration modes. MACSec is first configured in the global mode on the ingress and egress routers. Thereafter, MACSec and SyncE are configured on the required interfaces of the ingress and egress routers.

SyncE with Ethernet Synchronization Message Channel (ESMC) ensures network clock synchronization for successful network communication. The MACSec header is added to the ESMC packets. The ESMC MACSec header is used by SyncE interfaces to secure data on the physical media. See Configuring Synchronous Ethernet ESMC and SSM for more information.

This example shows how to configure MACSec in the global configuration mode:

At the ingress router:

```
Router(config)#key chain k10 macsec
Router(config-keychain-macsec)#key 10
Router(config-keychain-macsec-key)#cryptographic-algorithm aes-256-cmac
Router(config-keychain-macsec-key)#key-string
12345678901234567890123456789012123456789012345678901234567890123456789012
Router(config-keychain-macsec-key)#end
```

This example shows how to configure MACSec and SyncE on the GigabitEthernet0/5/0 interface:

```
Router(config)#interface GigabitEthernet0/5/0
Router(config-if)#mka pre-shared-key key-chain k10
Router(config-if)#macsec
MACSEC changed IP MTU on Gi0/5/0 from 1500 to 1468
Router(config-if)#no shutdown
Router(config-if)#synchronous mode
```

At the egress router:

This example shows how to configure MACSec in the global configuration mode:

```
Router(config)#key chain k10 macsec
Router(config-keychain-macsec)#key 10
Router(config-keychain-macsec-key)#cryptographic-algorithm aes-256-cmac
Router(config-keychain-macsec-key)#key-string
12345678901234567890123456789012123456789012345678901234567890123456789012
Router(config-keychain-macsec-key)#end
```

This example shows how to configure MACSec and SyncE on the GigabitEthernet0/2/0 interface:

```
Router(config)#interface GigabitEthernet0/2/0
Router(config-if)#mka pre-shared-key key-chain k10
Router(config-if)#macsec
MACSEC changed IP MTU on Gi0/5/0 from 1500 to 1468
Router(config-if)#no shutdown
Router(config-if)#synchronous mode
```

Verification

At the ingress router:

This example shows how to view the network clock synchronization details. The Gi0/2/0 interface denotes the egress router interface. Observe that SyncE and ESMC are enabled on the GigabitEthernet0/5/0 interface.

```
show network-clock synchronization detail
Symbols:     En - Enable, Dis - Disable, Adis - Admin Disable
             NA - Not Applicable
             *  - Synchronization source selected
             #  - Synchronization source force selected
             &  - Synchronization source manually switched
```

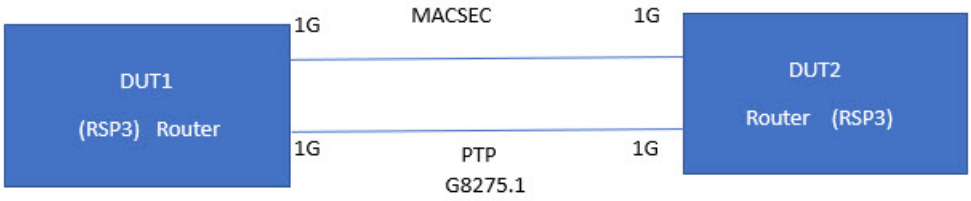
```
Automatic selection process : Enable
Equipment Clock : 2048 (EEC-Option1)
Clock State : Frequency Locked
Clock Mode : QL-Enable
ESMC : Enabled
SSM Option : 1
T0 : GigabitEthernet0/5/0
Hold-off (global) : 300 ms
Wait-to-restore (global) : 0 sec
Tsm Delay : 180 ms
Revertive : Yes
Force Switch: FALSE
Manual Switch: FALSE
Number of synchronization sources: 1
Squelch Threshold: QL-SEC
sm(netsync NETCLK_QL_ENABLE), running yes, state 1A
Last transition recorded: (begin)-> 2A (ql_mode_enable)-> 1A (src_added)-> 1A (sf_change)->
 1A (ql_change)-> 1A


Nominated Interfaces

 Interface            SigType     Mode/QL      Prio  QL_IN  ESMC Tx  ESMC Rx
 Internal             NA          NA/Dis       251   QL-SEC    NA        NA
*Gi0/2/0      NA             Sync/En          1      QL-PRC      -            -

Interface:
-----------------------------------------------
Local Interface: Internal
Signal Type: NA
Mode: NA(Ql-enabled)
SSM Tx: DISABLED
SSM Rx: DISABLED
Priority: 251
QL Receive: QL-SEC
QL Receive Configured: -
QL Receive Overrided: -
QL Transmit: -
QL Transmit Configured: -
Hold-off: 0
Wait-to-restore: 0
Lock Out: FALSE
Signal Fail: FALSE
Alarms: FALSE
Active Alarms: None
Slot Disabled: FALSE
SNMP input source index: 1
SNMP parent list index: 0
Description: None

Local Interface: Gi0/5/0
Signal Type: NA
Mode: Synchronous(Ql-enabled)
ESMC Tx: ENABLED
ESMC Rx: ENABLED
Priority: 1
QL Receive: QL-PRC
QL Receive Configured: -
QL Receive Overrided: -
QL Transmit: QL-DNU
QL Transmit Configured: -
```

```
Hold-off: 300
Wait-to-restore: 0
Lock Out: FALSE
Signal Fail: FALSE
Alarms: FALSE
Active Alarms :  None
Slot Disabled: FALSE
SNMP input source index: 6
SNMP parent list index: 0
Description: None
Router#
--- 00:57:29 ---
show esmc detail | sec GigabitEthernet0/5/0
Interface: GigabitEthernet0/5/0
  Administrative configurations:
    Mode: Synchronous
    ESMC TX: Enable
    ESMC RX: Enable
    QL TX: -
    QL RX: -
  Operational status:
    Port status: UP
    QL Receive: QL-PRC
    QL Transmit: QL-DNU
    QL rx overrided: -
    ESMC Information rate: 1 packet/second
    ESMC Expiry: 5 second
    ESMC Tx Timer: Running
    ESMC Rx Timer: Running
    ESMC Tx interval count: 1
    ESMC INFO pkts in: 3034
    ESMC INFO pkts out: 3058
    ESMC EVENT pkts in: 4
    ESMC EVENT pkts out: 7
```

Example: Configuring MACsec and PTP

Scenario 1—Sample Topology for MACsec and PTP on Different 1G Interfaces

The MACsec is configured first at the global level and then at the interface level. PTP is configured first in synchronous mode at the interface level and then at the global level.



Configuring MACsec

```
Router#configure terminal
```

```
Router(config)#key chain k10 macsec
Router(config)#key 10
Router(config)#cryptographic-algorithm aes-256-cmac
Router(config)#key-string 12345678901234567890123456789012123456789012345678901234567890123456789012
Router(config)#interface GigabitEthernet 0/5/6
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#no shutdown
Router(config-if)#mka pre-shared-key key-chain k10
Router(config-if)#macsec
Router(config-if)#end
```

Verifying MACsec Configuration

```
Router#show mka sess inter Gi0/5/6

Summary of All Currently Active MKA Sessions on Interface GigabitEthernet0/5/6...
===========================================================================================
Interface       Local-TxSCI          Policy-Name       Inherited          Key-Server

Port-ID         Peer-RxSCI           MACsec-Peers      Status             CKN

===========================================================================================
Gi0/5/6        00c1.b126.1c22/0028  *DEFAULT POLICY*  NO                 NO

40              0022.bddd.d97d/0033  1                 Secured            10

Router#
```

Configuring PTP (G8275.1)

```
Router#conf ter
Router(config)#interface GigabitEthernet0/5/4
Router(config-if)#synchronous mode
Router(config-if)#exit
Router(config)#network-clock revertive
Router(config)#network-clock synchronization automatic
Router(config)#network-clock synchronization mode QL-enabled
Router(config)#network-clock input source 1 External R0 10m
Router(config)#network-clock wait-to-restore 10 global
Router(config)#ptp clock ordinary domain 24  profile g8275.1
Router(config-ptp-clk)#tod R0 cisco
Router(config-ptp-clk)#input 1pps R0
Router(config-ptp-clk)#clock-port master master
Router(config-ptp-port)#transport ethernet multicast interface Gi0/3/4
Router(config-ptp-port)#end
```

Verifying PTP (G8275.1) Configuration

```
Router#show ptp clock  running

                    PTP Ordinary Clock [Domain 24]

         State          Ports          Pkts sent      Pkts rcvd      Redundancy Mode
         FREQ_LOCKED    1   1345           528              Hot standby

                               PORT SUMMARY

                     PTP Master
Name   Tx Mode      Role         Transport    State         Sessions     Port Addr
master mcast        master       Ethernet     Master        1            -
Router#
```