Configuring Control Plane Policing

Restrictions for CoPP

Restrictions for control plane policing (CoPP) include the following:

- Only ingress CoPP is supported. The **system-cpp-policy** policy-map is available on the control plane interface, and only in the ingress direction.
- Only the **system-cpp-policy** policy-map can be installed on the control plane interface.
- The **system-cpp-policy** policy-map and the 17 system-defined classes cannot be modified or deleted.
- Only the **police** action is allowed under the **system-cpp-policy** policy-map. The police rate for system-defined classes must be configured only in packets per second (pps); for user-defined class maps this must be configured only in bits per second (bps).
- We recommend not disabling the policer for a system-defined class map, that is, do not configure the **no police rate** *rate* **pps** command. Doing so affects the overall system health in case of high traffic towards the CPU. Further, even if you disable the policer rate for a system-defined class map, the system automatically reverts to the default policer rate after system bootup in order to protect the system bring-up process.
- When setting the policer rate, note that a clock frequency limitation causes differences in the `default rate` and the `set rate` values displayed for some classes (even if you set the default rate for all classes). See the *User-Configurable Aspects of CoPP* and *Example: Setting the Default Policer Rates for All CPU Queues* topics in this chapter for more information.
- Removing the policer rate configuration, disables CoPP on all affected queues.
- The **show run** command does not display information about classes configured under `system-cpp policy`, when they are left at default values. Use the **show policy-map system-cpp-policy** or the **show policy-map control-plane** commands instead.

You can continue use the **show run** command to display information about custom policies.

- A protocol with a huge number of CPU-bound packets may impact other protocols in the same class, as some of these protocols share the same policer. For example, Address Resolution Protocol (ARP) shares 4000 hardware policers with an array of host protocols like Telnet, Internet Control Message Protocol (ICMP), SSH, FTP, and SNMP in the system-cpp-police-forus class. If there is an ARP poisoning or an ICMP attack, hardware policers start throttling any incoming traffic that exceeds 4000 packets per second to protect the CPU and the overall integrity of the system. As a result, ARP and ICMP host protocols are dropped, along with any other host protocols that share the same class.
- Starting from Cisco IOS XE Fuji 16.8.1a, the creation of user-defined class-maps is not supported.

Information About CoPP

This chapter describes how control plane policing (CoPP) works on your device and how to configure it.

CoPP Overview

The CoPP feature improves security on your device protecting the CPU from unnecessary traffic and DoS attacks. It can also protect control and management traffic from traffic drops caused by high volumes of other, lower priority traffic.

Your device is typically segmented into three planes of operation, each with its own objective:

- The data plane, to forward data packets.
- The control plane, to route data correctly.
- The management plane, to manage network elements.

You can use CoPP to protect most of the CPU-bound traffic and ensure routing stability, reachability, and packet delivery. Most importantly, you can use CoPP to protect the CPU from a DoS attack.

CoPP uses the modular QoS command-line interface (MQC) and CPU queues to achieve these objectives. Different types of control plane traffic are grouped together based on certain criteria, and assigned to a CPU queue. You can manage these CPU queues by configuring dedicated policers in hardware. For example, you can modify the policer rate for certain CPU queues (traffic-type), or you can disable the policer for a certain type of traffic.

Although the policers are configured in hardware, CoPP does not affect CPU performance or the performance of the data plane. But since it limits the number of packets going to CPU, the CPU load is controlled. This means that services waiting for packets from hardware may see a more controlled rate of incoming packets (the rate being user-configurable).

System-Defined Aspects of CoPP

When you power-up the device for the first time, the system automatically performs the following tasks:

- Looks for policy-map **system-cpp-policy**. If not found, the system creates and installs it on the control-plane.
- Creates seventeen class-maps under **system-cpp-policy**.

The next time you power-up the device, the system detects the policy and class maps that have already been created.

- Enables all CPU queues by default, with their respective default rate. The default rates are indicated in the table System-Defined Values for CoPP.

The following table lists the class-maps that the system creates when you load the device. It lists the policer that corresponds to each class-map and one or more CPU queues that are grouped under each class-map. There is a one-to-one mapping of a class-map to a policer; and one-to-many mapping of a class-map to CPU queues.

Table 1: System-Defined Values for CoPP

Class Maps Names	Policer Index (Policer No.)	CPU queues (Queue No.)
system-cpp- police-data	WK_CPP_POLICE_DATA(0)	WK_CPU_Q_ICMP_GEN(3) WK_CPU_Q_BROADCAST(12) WK_CPU_Q_ICMP_REDIRECT(6)
system-cpp-police-l2- control	WK_CPP_POLICE_L2_ CONTROL(1)	WK_CPU_Q_L2_CONTROL(1)
system-cpp-police-routing-control	WK_CPP_POLICE_ROUTING_CONTROL(2)	WK_CPU_Q_ROUTING_CONTROL(4) WK_CPU_Q_LOW_LATENCY (27)
system-cpp-police-control-low-priority	WK_CPP_POLICE_CO NTROL_LOW_PRI(3)	WK_CPU_Q_GENERAL_PUNT(25)
system-cpp-police-punt-webauth	WK_CPP_POLICE_PU NT_WEBAUTH(7)	WK_CPU_Q_PUNT_WEBAUTH(22)
system-cpp-police- topology-control	WK_CPP_POLICE_TOPOLOGY_CONTROL(8)	WK_CPU_Q_TOPOLOGY_CONTROL(15)
system-cpp-police- multicast	WK_CPP_POLICE_MULTICAST(9)	WK_CPU_Q_TRANSIT_TRAFFIC(18) WK_CPU_Q_MCAST_DATA(30)
system-cpp-police-sys- data	WK_CPP_POLICE_SYS _DATA(10)	WK_CPU_Q_LEARNING_CACHE_OVFL(13) WK_CPU_Q_CRYPTO_CONTROL(23) WK_CPU_Q_EXCEPTION(24) WK_CPU_Q_EGR_EXCEPTION(28) WK_CPU_Q_NFL_SAMPLED_DATA(26) WK_CPU_Q_GOLD_PKT(31) WK_CPU_Q_RPF_FAILED(19)
system-cpp-police-dot1x-auth	WK_CPP_POLICE_DOT1X(11)	WK_CPU_Q_DOT1X_AUTH(0)
system-cpp-police- protocol-snooping	WK_CPP_POLICE_PR(12)	WK_CPU_Q_PROTO_SNOOPING(16)
system-cpp-police-dhcp-snooping	WK_CPP_DHCP_SNOOPING(6)	WK_CPU_Q_DHCP_SNOOPING(17)

Class Maps Names	Policer Index (Policer No.)	CPU queues (Queue No.)
system-cpp-police-sw-forward	WK_CPP_POLICE_SW_FWD (13)	WK_CPU_Q_SW_FORWARDING_Q(14) WK_CPU_Q_LOGGING(21) WK_CPU_Q_L2_LVX_DATA_PACK(11)
system-cpp-police-forus	WK_CPP_POLICE_FORUS(14)	WK_CPU_Q_FORUS_ADDR_RESOLUTION(5) WK_CPU_Q_FORUS_TRAFFIC(2)
system-cpp-police- multicast-end-station	WK_CPP_POLICE_MULTICAST_SNOOPING(15)	WK_CPU_Q_MCAST_END_STA TION_SERVICE(20)
system-cpp-default	WK_CPP_POLICE_DEFAULT_POLICER(16)	WK_CPU_Q_INTER_FED_TRAFFIC(7) WK_CPU_Q_EWLC_CONTROL(9) WK_CPU_Q_EWLC_DATA(10)
system-cpp-police-stackwise-virt-control	WK_CPP_STACKWISE_VIRTUAL_CONTROL(5)	WK_CPU_Q_STACKWISE_VIRTUAL_CONTROL (29)
system-cpp-police-l2lvx-control	WK_CPP_ L2_LVX_CONT_PACK(4)	WK_CPU_Q_L2_LVX_CONT_PACK(8)

The following table lists the CPU queues and the feature(s) associated with each CPU queue.

Table 2: CPU Queues and Associated Feature(s)

CPU queues (Queue No.)	Feature(s)
WK_CPU_Q_DOT1X_AUTH(0)	IEEE 802.1x Port-Based Authentication
WK_CPU_Q_L2_CONTROL(1)	Dynamic Trunking Protocol (DTP) VLAN Trunking Protocol (VTP) Port Aggregation Protocol (PAgP) Client Information Signaling Protocol (CISP) Message session relay protocol Multiple VLAN Registration Protocol (MVRP) Metropolitan Mobile Network (MMN) Link Level Discovery Protocol (LLDP) UniDirectional Link Detection (UDLD) Link Aggregation Control Protocol (LACP) Cisco Discovery Protocol (CDP) Spanning Tree Protocol (STP)

CPU queues (Queue No.)	Feature(s)
WK_CPU_Q_FORUS_TRAFFIC(2)	Host such as Telnet,Pingv4 and Pingv6, and SNMP Keepalive / loopback detection Initiate-Internet Key Exchange (IKE) protocol (IPSec)
WK_CPU_Q_ICMP_GEN(3)	ICMP - destination unreachable ICMP-TTL expired
WK_CPU_Q_ROUTING_CONTROL(4)	Routing Information Protocol version 1 (RIPv1) RIPv2 Interior Gateway Routing Protocol (IGRP) Border Gateway Protocol (BGP) PIM-UDP Virtual Router Redundancy Protocol (VRRP) Hot Standby Router Protocol version 1 (HSRPv1) HSRPv2 Gateway Load Balancing Protocol (GLBP) Label Distribution Protocol (LDP) Web Cache Communication Protocol (WCCP) Routing Information Protocol next generation (RIPng) Open Shortest Path First (OSPF) Open Shortest Path First version 3(OSPFv3) Enhanced Interior Gateway Routing Protocol (EIGRP) Enhanced Interior Gateway Routing Protocol version 6 (EIGRPv6) DHCPv6 Protocol Independent Multicast (PIM) Protocol Independent Multicast version 6 (PIMv6) Hot Standby Router Protocol next generation (HSRPng) IPv6 control Generic Routing Encapsulation (GRE) keepalive Network Address Translation (NAT) punt Intermediate System-to-Intermediate System (IS-IS)
WK_CPU_Q_FORUS_ADDR_RESOLUTION(5)	Address Resolution Protocol (ARP) IPv6 neighbor advertisement and neighbor solicitation
WK_CPU_Q_ICMP_REDIRECT(6)	Internet Control Message Protocol (ICMP) redirect

CPU queues (Queue No.)	Feature(s)
WK_CPU_Q_INTER_FED_TRAFFIC(7)	Layer 2 bridge domain inject for internal communication.
WK_CPU_Q_L2_LVX_CONT_PACK(8)	Exchange ID (XID) packet
WK_CPU_Q_EWLC_CONTROL(9)	Embedded Wirelss Controller (eWLC) [Control and Provisioning of Wireless Access Points (CAPWAP) (UDP 5246)]
WK_CPU_Q_EWLC_DATA(10)	eWLC data packet (CAPWAP DATA, UDP 5247)
WK_CPU_Q_L2_LVX_DATA_PACK(11)	Unknown unicast packet punted for map request.
WK_CPU_Q_BROADCAST(12)	All types of broadcast
WK_CPU_Q_LEARNING_CACHE_OVFL(13)	Learning cache overflow (Layer 2 + Layer 3)
WK_CPU_Q_SW_FORWARDING_Q(14)	Software forwarding
WK_CPU_Q_TOPOLOGY_CONTROL(15)	Spanning Tree Protocol (STP) Resilient Ethernet Protocol (REP) Shared Spanning Tree Protocol (SSTP)
WK_CPU_Q_PROTO_SNOOPING(16)	Address Resolution Protocol (ARP) snooping for Dynamic ARP Inspection (DAI)
WK_CPU_Q_DHCP_SNOOPING(17)	DHCP snooping
WK_CPU_Q_TRANSIT_TRAFFIC(18)	This is used for packets punted by NAT, which need to be handled in the software path.
WK_CPU_Q_RPF_FAILED(19)	Data – mRPF (multicast RPF) failed
WK_CPU_Q_MCAST_END_STATION _SERVICE(20)	Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD) control
WK_CPU_Q_LOGGING(21)	Access control list (ACL) logging
WK_CPU_Q_PUNT_WEBAUTH(22)	Web Authentication
WK_CPU_Q_CRYPTO_CONTROL(23)	Crypto control
WK_CPU_Q_EXCEPTION(24)	IKE indication IP learning violation IP port security violation IP Static address violation IPv6 scope check Remote Copy Protocol (RCP) exception Unicast RPF fail

CPU queues (Queue No.)	Feature(s)
WK_CPU_Q_GENERAL_PUNT(25)	General punt
WK_CPU_Q_NFL_SAMPLED_DATA(26)	Netflow sampled data and Media Services Proxy (MSP)
WK_CPU_Q_LOW_LATENCY(27)	Bidirectional Forwarding Detection (BFD), Precision Time Protocol (PTP)
WK_CPU_Q_EGR_EXCEPTION(28)	Egress resolution exception
WK_CPU_Q_STACKWISE_VIRTUAL _CONTROL(29)	Front side stacking protocols, namely SVL
WK_CPU_Q_MCAST_DATA(30)	Data - (S,G) creation Data - local joins Data - PIM Registration Data - SPT switchover Data - Multicast
WK_CPU_Q_GOLD_PKT(31)	Gold

User-Configurable Aspects of CoPP

You can perform these tasks to manage control plane traffic:

Note All `system-cpp-policy` configurations must be saved so they are retained after reboot.

Enable or Disable a Policer for CPU Queues

Enable a policer for a CPU queue, by configuring a policer action (in packets per second) under the corresponding class-map, within the `system-cpp-policy` policy-map.

Disable a policer for CPU queue, by removing the policer action under the corresponding class-map, within the `system-cpp-policy` policy-map.

Note If a default policer is already present, carefully consider and control its removal; otherwise the system may see a CPU hog or other anomalies, such as control packet drops.

Change the Policer Rate

You can do this by configuring a policer rate action (in packets per second), under the corresponding class-map, within the`system-cpp-policy` policy-map.

When setting a policer rate, note that the rate you set is automatically converted to the nearest multiple of 200. For instance, if you set the policer rate of a CPU queue 100 pps, the system changes it to 200; or if set

the policer rate to 650, the system changes it to 600. See *Example: Setting the Default Policer Rates for All CPU Queues* in this chapter, for sample output that displays this behavior.

Set Policer Rates to Default

Set the policer for CPU queues to their default values, by entering the **cpp system-default** command in global configuration mode.

Upgrading or Downgrading the Software Version

Software Version Upgrades and CoPP

When you upgrade the software version on your device, the system checks and make the necessary updates as required for CoPP (For instance, it checks for the `system-cpp-policy` policy map and creates it if missing). You may also have to complete certain tasks before or after the upgrade activity. This is to ensure that any configuration updates are reflected correctly and CoPP continues to work as expected. Depending on the method you use to upgrade the software, upgrade-related tasks may be optional or recommended in some scenarios, and mandatory in others.

The system actions and user actions for an upgrade, are described here. Also included, are any release-specfic caveats.

System Actions for an Upgrade

When you upgrade the software version on your device, the system performs these actions. This applies to all upgrade methods:

- If the device did not have a `system-cpp-policy` policy map before upgrade, then on upgrade, the system creates a default policy map.
- If the device had a `system-cpp-policy` policy map before upgrade, then on upgrade, the system does not re-generate the policy.

User Actions for an Upgrade

User actions for an upgrade – depending on upgrade method:

Upgrade Method	Condition	Action Time and Action	Purpose
Regular[1]	None	**After upgrade (required)** Enter the **cpp system-default** command in global configuration mode	To get the latest, default policer rates.

[1] Refers to a software upgrade method that involves a reload of the switch. Can be install or bundle mode.

Software Version Downgrades and CoPP

The system actions and user actions for a downgrade, are described here.

System Actions for a Downgrade

When you downgrade the software version on your device, the system performs these actions. This applies to all downgrade methods:

- The system retains the `system-cpp-policy` policy map on the device, and installs it on the control plane.

User Actions for a Downgrade

User actions for a downgrade:

Upgrade Method	Condition	Action Time and Action	Purpose
Regular[2]	None	No action required	Not applicable

[2] Refers to a software upgrade method that involves a reload of the switch. Can be install or bundle mode.

If you downgrade the software version and then again upgrade, the system action and user actions that apply are the same as those mentioned for upgrades.

How to Configure CoPP

Enabling a CPU Queue or Changing the Policer Rate

The procedure to enable a CPU queue and change the policer rate of a CPU queue is the same. Follow these steps:

Procedure

	Command or Action	Purpose
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	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map system-cpp-policy`
	`Device(config-pmap)#`
		Enters the policy map configuration mode.

	Command or Action	Purpose

Step 4	**class** *class-name*
	Example:
	`Device(config-pmap)# class system-cpp-police-protocol-snooping`
	`Device(config-pmap-c)#`
		Enters the class action configuration mode. Enter the name of the class that corresponds to the CPU queue you want to enable. See table *System-Defined Values for CoPP*.

Step 5	**police rate** *rate* **pps**
	Example:
	`Device(config-pmap-c)# police rate 100 pps`
	`Device(config-pmap-c-police)#`
		Specifies an upper limit on the number of incoming packets processed per second, for the specified traffic class.
		Note The rate you specify is applied to all CPU queues that belong to the class-map you have specified.

Step 6	exit
	Example:
	`Device(config-pmap-c-police)# exit`
	`Device(config-pmap-c)# exit`
	`Device(config-pmap)# exit`
	`Device(config)#`
		Returns to the global configuration mode.

Step 7	**control-plane**
	Example:
	`Device(config)# control-plane`
	`Device(config-cp)#`
		Enters the control plane (config-cp) configuration mode

Step 8	**service-policy input** *policy-name*
	Example:
	`Device(config)# control-plane`
	`Device(config-cp)#service-policy input system-cpp-policy`
	`Device(config-cp)#`
		Installs system-cpp-policy in FED. This command is required for you to see the FED policy. Not configuring this command will lead to an error.

Step 9	**end** **Example:** `Device(config-cp)# end`	Returns to the privileged EXEC mode.
Step 10	**show policy-map control-plane** **Example:** `Device# show policy-map control-plane`	Displays all the classes configured under `system-cpp policy`, the rates configured for the various traffic types, and statistics

Disabling a CPU Queue

Follow these steps to disable a CPU queue:

Procedure

	Command or Action	Purpose
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	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map system-cpp-policy`
	`Device(config-pmap)#`
		Enters the policy map configuration mode.

Step 4	**class** *class-name*
	Example:
	`Device(config-pmap)# class system-cpp-police-protocol-snooping`
	`Device(config-pmap-c)#`
		Enters the class action configuration mode. Enter the name of the class that corresponds to the CPU queue you want to disable. See the table, *System-Defined Values for CoPP*.

Step 5	**no police rate** *rate* **pps** **Example:** `Device(config-pmap-c)# no police rate 100 pps`	Disables incoming packet processing for the specified traffic class. **Note** This disables all CPU queues that belong to the class-map you have specified.
Step 6	**end** **Example:** `Device(config-pmap-c)# end`	Returns to the privileged EXEC mode.
Step 7	**show policy-map control-plane** **Example:** `Device# show policy-map control-plane`	Displays all the classes configured under `system-cpp policy` and the rates configured for the various traffic types and statistics.

Setting the Default Policer Rates for All CPU Queues

Follow these steps to set the policer rates for all CPU queues to their default rates:

Procedure

	Command or Action	Purpose
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	**cpp system-default**
	Example:
	`Device(config)# cpp system-default`
	`Defaulting CPP : Policer rate for all classes will be set to their defaults`
		Sets the policer rates for all the classes to the default rate.

Step 4	**end** **Example:** `Device(config)# end`	Returns to the privileged EXEC mode.
Step 5	**show platform hardware fed** { **active** | **standby** } **qos queue stats internal cpu policer** **Example:** `Device# show platform hardware fed active qos queue stats internal cpu policer`	Displays device-specific internal queue information.

Configuration Examples for CoPP

Example: Enabling a CPU Queue or Changing the Policer Rate of a CPU Queue

This example shows how to enable a CPU queue or to change the policer rate of a CPU queue. Here the **class system-cpp-police-protocol-snooping** CPU queue is enabled with the policer rate of **2000 pps** .

```
Device> enable
Device# configure terminal
Device(config)# policy-map system-cpp-policy
Device(config-pmap)# class system-cpp-police-protocol-snooping
Device(config-pmap-c)# police rate 2000 pps
Device(config-pmap-c-police)# end


Device# show policy-map control-plane
Control Plane

  Service-policy input: system-cpp-policy

    <output truncated>


    Class-map: system-cpp-police-dot1x-auth (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: none
      police:
          rate 1000 pps, burst 244 packets
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop

    Class-map: system-cpp-police-protocol-snooping (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: none
      police:
          rate 2000 pps, burst 488 packets
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop

    <output truncated>

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any
```

Example: Disabling a CPU Queue

This example shows how to disable a CPU queue. Here the **class system-cpp-police-protocol-snooping** CPU queue is disabled.

```
Device> enable
Device# configure terminal
Device(config)# policy-map system-cpp-policy
Device(config-pmap)# class system-cpp-police-protocol-snooping
Device(config-pmap-c)# no police rate 100 pps
Device(config-pmap-c)# end


Device# show running-config | begin system-cpp-policy

policy-map system-cpp-policy
 class system-cpp-police-data
  police rate 200 pps
 class system-cpp-police-sys-data
  police rate 100 pps
 class system-cpp-police-sw-forward
  police rate 1000 pps
 class system-cpp-police-multicast
  police rate 500 pps
 class system-cpp-police-multicast-end-station
  police rate 2000 pps
 class system-cpp-police-punt-webauth
 class system-cpp-police-l2-control
 class system-cpp-police-routing-control
  police rate 500 pps
 class system-cpp-police-control-low-priority
 class system-cpp-police-wireless-priority1
 class system-cpp-police-wireless-priority2
 class system-cpp-police-wireless-priority3-4-5
 class system-cpp-police-topology-control
 class system-cpp-police-dot1x-auth
 class system-cpp-police-protocol-snooping
 class system-cpp-police-forus
 class system-cpp-default

<output truncated>
```

Example: Setting the Default Policer Rates for All CPU Queues

This example shows how to set the policer rates for all CPU queues to their default and then verify the setting.

Note For some CPU queues, the `default rate` and the `set rate` values will not be the same, even if you set the default rate for all classes. This because the set rate is rounded off to the nearest multiple of 200. This behavior is controlled by the clock speed of your device. In the sample output below, the default and set rate values for `DHCP Snooping` and `NFL SAMPLED DATA` display this difference.

```
Device> enable
Device# configure terminal
Device(config)# cpp system-default
```

```
Defaulting CPP : Policer rate for all classes will be set to their defaults
Device(config)# end

Device# show platform hardware fed active qos queue stats internal cpu policer

CPU Queue Statistics
============================================================================================

                                                    (default) (set)      Queue         Queue
QId PlcIdx  Queue Name                 Enabled      Rate      Rate       Drop(Bytes)   Drop(Frames)
--------------------------------------------------------------------------------------------
0   11      DOT1X Auth                   Yes        1000      1000       0             0
1   1       L2 Control                   Yes        2000      2000       0             0
2   14      Forus traffic                Yes        4000      4000       0             0
3   0       ICMP GEN                     Yes        600       600        0             0
4   2       Routing Control              Yes        5400      5400       0             0
5   14      Forus Address resolution     Yes        4000      4000       0             0
6   0       ICMP Redirect                Yes        600       600        0             0
7   16      Inter FED Traffic            Yes        2000      2000       0             0
8   4       L2 LVX Cont Pack             Yes        1000      1000       0             0
9   16      EWLC Control                 Yes        2000      2000       0             0
10  16      EWLC Data                    Yes        2000      2000       0             0
11  13      L2 LVX Data Pack             Yes        1000      1000       0             0
12  0       BROADCAST                    Yes        600       600        0             0
13  10      Learning cache ovfl          Yes        100       200        0             0
14  13      Sw forwarding                Yes        1000      1000       0             0
15  8       Topology Control             Yes        13000     13000      0             0
16  12      Proto Snooping               Yes        2000      2000       0             0
17  6       DHCP Snooping                Yes        500       400        0             0
18  9       Transit Traffic              Yes        500       400        0             0
19  10      RPF Failed                   Yes        100       200        0             0
20  15      MCAST END STATION            Yes        2000      2000       0             0
21  13      LOGGING                      Yes        1000      1000       0             0
22  7       Punt Webauth                 Yes        1000      1000       0             0
23  10      Crypto Control               Yes        100       200        0             0
24  10      Exception                    Yes        100       200        0             0
25  3       General Punt                 Yes        200       200        0             0
26  10      NFL SAMPLED DATA             Yes        100       200        0             0
```

```
27   2      Low Latency                Yes     5400      5400      0           0

28   10     EGR Exception              Yes     100       200       0           0

29   5      Stackwise Virtual Control  Yes     8000      8000      0           0

30   9      MCAST Data                 Yes     500       400       0           0

31   10     Gold Pkt                   Yes     100       200       0           0

* NOTE: CPU queue policer rates are configured to the closest hardware supported value

                      CPU Queue Policer Statistics
=============================================================================
Policer    Policer Accept   Policer Accept  Policer Drop  Policer Drop
  Index         Bytes          Frames         Bytes          Frames
-----------------------------------------------------------------------------
0          0                0               0             0
1          0                0               0             0
2          0                0               0             0
3          0                0               0             0
4          0                0               0             0
5          0                0               0             0
6          0                0               0             0
7          0                0               0             0
8          0                0               0             0
9          0                0               0             0
10         0                0               0             0
11         0                0               0             0
12         0                0               0             0
13         0                0               0             0
14         0                0               0             0
15         0                0               0             0
16         0                0               0             0
17         0                0               0             0

                        CPP Classes to queue map
=============================================================================================
PlcIdx CPP Class                                 :  Queues
---------------------------------------------------------------------------------------------
0      system-cpp-police-data                    :  ICMP GEN/BROADCAST/ICMP Redirect/
10     system-cpp-police-sys-data                :  Learning cache ovfl/Crypto
Control/Exception/EGR Exception/NFL SAMPLED DATA/Gold Pkt/RPF Failed/
13     system-cpp-police-sw-forward              :  Sw forwarding/LOGGING/L2 LVX Data Pack/
9      system-cpp-police-multicast               :  Transit Traffic/MCAST Data/
15     system-cpp-police-multicast-end-station   :  MCAST END STATION /
7      system-cpp-police-punt-webauth            :  Punt Webauth/
1      system-cpp-police-l2-control              :  L2 Control/
2      system-cpp-police-routing-control         :  Routing Control/Low Latency/
3      system-cpp-police-control-low-priority    :  General Punt/
4      system-cpp-police-l2lvx-control           :  L2 LVX Cont Pack/
8      system-cpp-police-topology-control        :  Topology Control/
11     system-cpp-police-dot1x-auth              :  DOT1X Auth/
12     system-cpp-police-protocol-snooping       :  Proto Snooping/
6      system-cpp-police-dhcp-snooping           :  DHCP Snooping/
14     system-cpp-police-forus                   :  Forus Address resolution/Forus traffic/
5      system-cpp-police-stackwise-virt-control  :  Stackwise Virtual Control/
16     system-cpp-default                        :  Inter FED Traffic/EWLC Control/EWLC Data/
```

Monitoring CoPP

Use these commands to display policer settings, such as, traffic types and policer rates (user-configured and default rates) for CPU queues:

Command	Purpose
show policy-map control-plane	Displays the rates configured for the various traffic types
show policy-map system-cpp-policy	Displays all the classes configured under system-cpp policy, and policer rates

Feature History and Information for CoPP

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Feature	Release	Feature Information
Control Plane Policing (CoPP) or CPP	Cisco IOS XE Everest 16.6.1	This feature was introduced.
Change in the system behavior for policer rates that are set.	Cisco IOS XE Everest 16.6.4	For some CPU queues, the default rate and the set rate values will not be the same, even if you set the default rate for all classes. This because the set rate is rounded off to the nearest multiple of 200.

Feature	Release	Feature Information

Changes in system-defined values for CoPP	Cisco IOS XE Fuji 16.8.1a	• Starting from this release, the creation of user-defined class-maps is not supported.
		• This new system-defined class was introduced: system-cpp-police-dhcp-snooping
		• This new CPU queue was added to the existing system-cpp-default class: WK_CPU_Q_INTER_FED_TRAFFIC
		• These CPU queues are no longer available:
		  • WK_CPU_Q_SHOW_FORWARD
		  • WK_CPU_Q_UNUSED
		• The default policer rate (pps) for some CPU queues has changed:
		  • The default rate for WK_CPU_Q_EXCEPTION(24) was changed to 100
		  • The default rate for all the CPU queues under system-cpp-default was increased to 2000.
		  • The default rate for all the CPU queues under system-cpp-police-forus was increased to 4000.