



Configuring Control Plane Policing

- [Restrictions for CoPP, on page 1](#)
- [Information About CoPP, on page 2](#)
- [How to Configure CoPP, on page 10](#)
- [Configuration Examples for CoPP, on page 14](#)
- [Monitoring CoPP, on page 18](#)
- [Feature History and Information for CoPP, on page 18](#)

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.
- One or more CPU queues are part of each class-map. Changing the policer rate of a class-map affects all CPU queues that belong to that class-map. Similarly, disabling the policer in a class-map disables all queues that belong to that class-map. See [#unique_707 unique_707_Connect_42_table_9655B5349A524FCAA9AC1EDCDB9FC5F4](#) for information about which CPU queues belong to each class-map.

- 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_CONTROL_LOW_PRI(3)	WK_CPU_Q_GENERAL_PUNT(25)
system-cpp-police-punt-webauth	WK_CPP_POLICE_PUNT_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)

Class Maps Names	Policer Index (Policer No.)	CPU queues (Queue No.)
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)
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(5)	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(6)	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

CPU queues (Queue No.)	Feature(s)
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)
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

CPU queues (Queue No.)	Feature(s)
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
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)]

CPU queues (Queue No.)	Feature(s)
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_CRYPTOCONTROL(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
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

CPU queues (Queue No.)	Feature(s)
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.

Create User-Defined Class Maps

If a given traffic class does not have a designated class map, and you want to protect this traffic, you can create specific class maps (with filters) for such traffic packets and add these user-defined class maps to `system-cpp-policy`.

While `system-cpp-policy` is applied in the ingress direction, the forwarding engine driver (FED) changes policers on user-defined class maps to the egress. The filters and the policers in all user-defined classes must therefore be applied as egress classifications and actions, respectively. The policy map itself is unaffected by this change in the direction.

When you add a user-defined class map to `system-cpp-policy`, the system automatically installs it on all 32 CPU queues (in addition to the control plane), resulting in 33 instances of the policy. You can see this by entering the **show platform software fed switch** `{switch_number | active | standby} qos policy target status` command in privileged EXEC mode.

The police rate on these class maps is controlled by the Active Queue Management (AQM) policer. AQM provides buffering control of traffic flows prior to queuing a packet into the transmit queue of a port, ensuring that certain flows do not hog the switch packet memory. If the AQM policer feature is enabled, any user-defined police rates exceeding the AQM policer limits are disregarded.

User defined class maps have normal QoS or ACL classification filters.

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-specific 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.

	Command or Action	Purpose
Step 2	configure terminal Example: <pre>Device# configure terminal</pre>	Enters global configuration mode.
Step 3	policy-map <i>policy-map-name</i> Example: <pre>Device(config)# policy-map system-cpp-policy Device(config-pmap)#</pre>	Enters the policy map configuration mode.
Step 4	class <i>class-name</i> Example: <pre>Device(config-pmap)# class system-cpp-police-protocol-snooping Device(config-pmap-c)#</pre>	Enters the class action configuration mode. Enter the name of the class that corresponds to the CPU queue you want to enable. See table <i>System-Defined Values for CoPP</i> .
Step 5	police rate <i>rate</i> pps Example: <pre>Device(config-pmap-c)# police rate 100 pps Device(config-pmap-c-police)#</pre>	<p>Specifies an upper limit on the number of incoming packets processed per second, for the specified traffic class.</p> <p>Note The rate you specify is applied to all CPU queues that belong to the class-map you have specified.</p>
Step 6	exit Example: <pre>Device(config-pmap-c-police)# exit Device(config-pmap-c)# exit Device(config-pmap)# exit Device(config)#</pre>	Returns to the global configuration mode.
Step 7	control-plane Example: <pre>Device(config)# control-plane Device(config-cp)#</pre>	Enters the control plane (config-cp) configuration mode
Step 8	service-policy input <i>policy-name</i> Example: <pre>Device(config)# control-plane Device(config-cp)# service-policy input system-cpp-policy Device(config-cp)#</pre>	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.

	Command or Action	Purpose
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 <code>system-cpp policy</code> , 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. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	policy-map <i>policy-map-name</i> Example: Device(config) # policy-map system-cpp-policy Device(config-pmap) #	Enters the policy map configuration mode.
Step 4	class <i>class-name</i> 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, <i>System-Defined Values for CoPP</i> .
Step 5	no police rate <i>rate</i> pps Example:	Disables incoming packet processing for the specified traffic class.

	Command or Action	Purpose
	Device(config-pmap-c)# no police rate 100 pps	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 <code>system-cpp policy</code> 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. <ul style="list-style-type: none">• 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.

	Command or Action	Purpose
Step 5	show platform hardware fed switch {switch-number} qos que stats internal cpu policer Example: Device# show platform hardware fed switch 1 qos que stat internal cpu policer	Displays the rates configured for the various traffic types.

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 switch 1 qos queue stats internal cpu policer
```

```
CPU Queue Statistics
```

=====							
QId	PlcIdx	Queue Name	Enabled	(default) Rate	(set) Rate	Queue Drop (Bytes)	Queue 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 Index	Policer Accept Bytes	Policer Accept Frames	Policer Drop Bytes	Policer Drop 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
show platform hardware fed switch {switch-number} qos que stats internal cpu policer	Displays the rates configured for the various traffic types
show platform software fed {switch-number} qos policy target status	Displays information about policy status and the target port type.

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 16.5.1a	<p>This feature was introduced.</p> <p>The CoPP feature improves security on your device by protecting the CPU from unnecessary traffic, or DoS traffic, and by prioritizing control plane and management traffic.</p> <p>The feature provides CLI configuration options to enable and disable CPU queues, to change the policer rate, set policer rates to default, and to create user-defined class-maps.</p>

Feature	Release	Feature Information
Changes in system-defined values for CoPP	Cisco IOS XE Everest 16.6.1	<p>These new system-defined classes were introduced:</p> <ul style="list-style-type: none">• system-cpp-police-stackwise-virt-control• system-cpp-police-l2lvx-control <p>These new CPU queues were added to the existing system-cpp-default class:</p> <ul style="list-style-type: none">• WK_CPU_Q_UNUSED (7)• WK_CPU_Q_EWLC_CONTROL(9)• WK_CPU_Q_EWLC_DATA(10) <p>This new CPU queues was added to the existing system-cpp-police-sw-forward:</p> <p>WK_CPU_Q_L2_LVX_DATA_PACK (11)</p> <p>This CPU queue is no longer available:</p> <p>WK_CPU_Q_SGT_CACHE_FULL(27)</p>
Change in the system behavior for policer rates that are set.	Cisco IOS XE Everest 16.6.4	<p>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.</p>

Feature	Release	Feature Information
Changes in system-defined values for CoPP	Cisco IOS XE Fuji 16.8.1a	<ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> WK_CPU_Q_SHOW_FORWARD WK_CPU_Q_UNUSED The default policer rate (pps) for some CPU queues has changed: <ul style="list-style-type: none"> 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.