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.
- One or more CPU queues are part of each class-map. Where multiple CPU queues belong to one 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 Table 1: System-Defined Values for CoPP, on page 3 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.

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.)	Default Policer Rate (pps)
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)	600 600 600
system-cpp-police-l2- control	WK_CPP_POLICE_L2_ CONTROL(1)	WK_CPU_Q_L2_CONTROL(1)	2000
system-cpp-police-routing-control	WK_CPP_POLICE_ROUTING_CONTROL(2)	WK_CPU_Q_ROUTING_CONTROL(4) WK_CPU_Q_LOW_LATENCY (27)	5400 5400
system-cpp-police-control-low-priority	WK_CPP_POLICE_CO NTROL_LOW_PRI(3)	WK_CPU_Q_GENERAL_PUNT(25)	200
system-cpp-police-punt-webauth	WK_CPP_POLICE_PU NT_WEBAUTH(7)	WK_CPU_Q_PUNT_WEBAUTH(22)	1000
system-cpp-police- topology-control	WK_CPP_POLICE_TOPOLOGY_CONTROL(8)	WK_CPU_Q_TOPOLOGY_CONTROL(15)	13000
system-cpp-police- multicast	WK_CPP_POLICE_MULTICAST(9)	WK_CPU_Q_TRANSIT_TRAFFIC(18) WK_CPU_Q_MCAST_DATA(30)	500 500
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)	100 100 200 100 100 100 100
system-cpp-police-dot1x-auth	WK_CPP_POLICE_DOT1X(11)	WK_CPU_Q_DOT1X_AUTH(0)	1000
system-cpp-police- protocol-snooping	WK_CPP_POLICE_PR	WK_CPU_Q_PROTO_SNOOPING(16)	2000
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)	1000 1000 1000

Class Maps Names	Policer Index (Policer No.)	CPU queues (Queue No.)	Default Policer Rate (pps)
system-cpp-police-forus	WK_CPP_POLICE_FORUS(14)	WK_CPU_Q_FORUS_ADDR_RESOLUTION(5) WK_CPU_Q_FORUS_TRAFFIC(2)	1000 1000
system-cpp-police-multicast-end-station	WK_CPP_POLICE_MULTICAST_SNOOPING(6)	WK_CPU_Q_MCAST_END_STATION_SERVICE(20)	2000
system-cpp-default	WK_CPP_POLICE_DEFAULT_POLICER	WK_CPU_Q_DHCP_SNOOPING(17) WK_CPU_Q_UNUSED (7) WK_CPU_Q_EWLC_CONTROL(9) WK_CPU_Q_EWLC_DATA(10)	1000 1000 1000 1000
system-cpp-police-stackwise-virt-control	WK_CPP_STACKWISE_VIRTUAL_CONTROL	WK_CPU_Q_STACKWISE_VIRTUAL_CONTROL (29)	8000
system-cpp-police-l2lvx-control	WK_CPP_ L2_LVX_CONT_PACK	WK_CPU_Q_L2_LVX_CONT_PACK(8)	1000

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)
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)

CPU queues (Queue No.)	Feature(s)
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
WK_CPU_Q_UNUSED (7)	Unused
WK_CPU_Q_L2_LVX_CONT_PACK(8)	Exchange ID (XID) packet

CPU queues (Queue No.)	Feature(s)
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)	Data - access control list (ACL) Full Data - IPv4 options Data - IPv6 hop-by-hop Data - out-of-resources / catch all Data - Reverse Path Forwarding (RPF) incomplete Glean packet
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 Controls

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

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-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.

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`	Returns to the global configuration mode.

	Command or Action	Purpose

	`Device(config-pmap-c)# exit`
	`Device(config-pmap)# exit`
	`Device(config)#`	

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.

	Command or Action	Purpose

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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	

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 switch**{*switch-number* | *active* | *standby*}**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.

Creating A User-Defined Class Map

Follow these steps to create user-defined class maps in system-cpp-policy and set the policer rates in bps

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	**class-map** *class-map-name* **Example:**	Specify the class map you want to create. Enters the class map configuration mode.

	Command or Action	Purpose
	`Device(config)# class-map example_class` `Device(config-cmap)#`	

Step 4	**exit**
	Example:
	`Device(config-cmap)# exit`
	`Device(config)#`
		Exits the class map configuration mode.

Step 5	**policy-map** *policy-map-name*
	Example:
	`Device(config)# policy-map system-cpp-policy`
	`Device(config-pmap)#`
		Enter the policy map name. Enters the policy map configuration mode.

Step 6	**class-map** *class-map-name*
	Example:
	`Device(config-pmap)# class example_class`
	`Device(config-pmap-c)#`
		Enters the class action configuration mode. Enter the name of the class.

Step 7	[**no**] **police rate** *target_bit_rate* **Example:** `Device(config-pmap-c)# police 90000`	Specifies the bit rate per second, enter a value between 8000 and 10000000000. **Note** The police rate for user-defined class-maps must not exceed 10000 pps worth of traffic.

Step 8	**end**
	Example:
	`Device(config-pmap-c-police)# end`
	`Device#`
		Returns to the privileged EXEC mode.

Step 9	**show policy-map control-plane** **Example:** `Device# show policy-map control-plane`	Displays all the classes configured under `system-cpp policy`, including the user-defined class maps, and the rates configured.

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.

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.

```
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
============================================================================================

                                              (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     Openflow                     Yes     100       100      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       500      0             0

18   9      Transit Traffic              Yes     500       500      0             0

19   10     RPF Failed                   Yes     100       100      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   18     High Rate App                Yes     13000     13000    0             0

24   10     Exception                    Yes     100       100      0             0

25   3      System Critical              Yes     1000      1000     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       100      0            0

29   5      Stackwise Virtual OOB       Yes     8000      8000     0            0

30   9      MCAST Data                  Yes     500       500      0            0

31   10     Gold Pkt                    Yes     100       100      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
18         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               :  Openflow/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-system-critical        :  System Critical/
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 OOB/
16     system-cpp-default                     :  Inter FED Traffic/EWLC Control/EWLC Data/
18     system-cpp-police-high-rate-app          :  High Rate App/
```

Example: Creating a User-Defined Class Map

Device

This example shows how to create a user-defined class map, apply it to `system-cpp-policy` and display information about where the policy is applied.

A user-defined class map is applied to `system-cpp-policy`, which means that any control traffic matching the user-defined class map `class-cpp-user` is subject to the aggregate policer, under the user-defined class map. Statistics for the user defined traffic class are reported in Bytes.

```
Device> enable
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# class-map match-any class-cpp-user
Device(config-cmap)# match dscp cs1
Device(config-cmap)# exit
Device(config)# policy-map system-cpp-policy
Device(config-pmap)# class class-cpp-user
Device(config-pmap-c)# police rate 2m bps
Device(config-pmap-c-police)# end
```

```
Device# show policy-map control-plane
<output truncated>
Class-map: class-cpp-user (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match:  dscp cs1 (8)
      police:
          rate 2000000 bps, burst 62500 bytes
        conformed 0 bytes; actions:
          transmit
        exceeded 0 bytes; actions:
          drop
        conformed 0000 bps, exceeded 0000 bps
<output truncated>
```

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).

Note how the direction is display as egress (OUT), even though system-cpp-policy is applied in the ingress

```
Device# show platform software fed switch active qos policy target status

TCG status summary:

Loc Interface             IIF-ID           Dir State:(cfg,opr) Policy
--- --------------------- ---------------- --- --------------- --------------------
?:255 Control Plane       0x00000001000001 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-0          0x0000000100000d OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-1          0x0000000100000e OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-2          0x0000000100000f OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-3          0x00000001000010 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-4          0x00000001000011 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-5          0x00000001000012 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-6          0x00000001000013 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-7          0x00000001000014 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-8          0x00000001000015 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-9          0x00000001000016 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-10         0x00000001000017 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-11         0x00000001000018 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-12         0x00000001000019 OUT VALID,SET_INHW  system-cpp-policy
```

```
?:0 CoPP-Queue-13         0x000000010001a OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-14         0x000000010001b OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-15         0x000000010001c OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-16         0x000000010001d OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-17         0x000000010001e OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-18         0x000000010001f OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-19         0x0000000100020 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-20         0x0000000100021 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-21         0x0000000100022 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-22         0x0000000100023 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-23         0x0000000100024 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-24         0x0000000100025 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-25         0x0000000100026 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-26         0x0000000100027 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-27         0x0000000100028 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-28         0x0000000100029 OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-29         0x000000010002a OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-30         0x000000010002b OUT VALID,SET_INHW  system-cpp-policy
?:0 CoPP-Queue-31         0x000000010002c OUT VALID,SET_INHW  system-cpp-policy
```

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* | *active* | *standby* } **qos que stats internal cpu policer**	Displays the rates configured for the various traffic types
show platform software fed { *switch-number* | *active* | *standby* } **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 3.2SE	This feature was introduced.

Feature	Release	Feature Information
CLI configuration for CoPP	Cisco IOS XE Denali 16.1.2	This feature was made user-configurable. CLI configuration options to enable and disable CPU queues, to change the policer rate, and to set policer rates to default.
User-defined class maps	Cisco IOS XE Everest 16.5.1a	Starting with this release, you can create class maps (with filters) and add these user-defined class maps to system-cpp-policy.
Changes in system-defined values for CoPP	Cisco IOS XE Everest 16.6.1	These new system-defined classes were introduced: • system-cpp-police-stackwise-virt-control • system-cpp-police-l2lvx-control These new CPU queues were added to the existing system-cpp-default class: • WK_CPU_Q_UNUSED (7) • WK_CPU_Q_EWLC_CONTROL(9) • WK_CPU_Q_EWLC_DATA(10) This new CPU queues was added to the existing system-cpp-police-sw-forward: WK_CPU_Q_L2_LVX_DATA_PACK (11) This CPU queue is no longer available: WK_CPU_Q_SGT_CACHE_FULL(27)