QoS Classification

QoS classification identifies and marks traffic flows that require congestion management or congestion avoidance on a data path. The Modular Quality of Service (QoS) command-line interface (MQC) is used to define the traffic flows that should be classified, where each traffic flow is called a class of service, or class. Subsequently, a traffic policy is created and applied to a class. All traffic not identified by defined classes falls into the category of a default class.

This chapter provides the conceptual and configuration information for QoS packet classification.

Packet Classification Overview

Packet classification involves categorizing a packet within a specific group (or class) and assigning it a traffic descriptor to make it accessible for QoS handling on the network. The traffic descriptor contains information about the forwarding treatment (quality of service) that the packet should receive. Using packet classification, you can partition network traffic into multiple priority levels or classes of service. The source agrees to adhere to the contracted terms and the network promises a quality of service. Traffic policers and traffic shapers use the traffic descriptor of a packet to ensure adherence to the contract.

Traffic policers and traffic shapers rely on packet classification features, such as IP precedence, to select packets (or traffic flows) traversing a router or interface for different types of QoS service. For example, by using the three precedence bits in the type of service (ToS) field of the IP packet header, you can categorize packets into a limited set of up to eight traffic classes. After you classify packets, you can use other QoS features to assign the appropriate traffic handling policies including congestion management, bandwidth allocation, and delay bounds for each traffic class.

The maximum transmission unit (MTU) for interfaces (both bundle and physical) is 9644, that is, interfaces can accept packets with a maximum of 9644 bytes.

Ingress classification techniques:

- match cos
- match dscp
- match ip precedence

- match exponential
- match dei

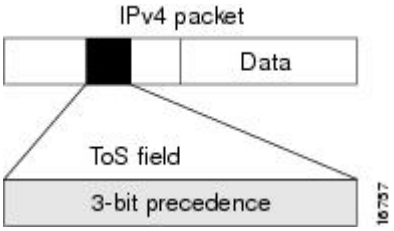
Egress classification techniques:

- match traffic-class
- match discard-class
- match qos-group

Specification of the CoS for a Packet with IP Precedence

Use of IP precedence allows you to specify the CoS for a packet. You can create differentiated service by setting precedence levels on incoming traffic and using them in combination with the QoS queuing features. So that, each subsequent network element can provide service based on the determined policy. IP precedence is usually deployed as close to the edge of the network or administrative domain as possible. This allows the rest of the core or backbone to implement QoS based on precedence.

Figure 1: IPv4 Packet Type of Service Field



You can use the three precedence bits in the type-of-service (ToS) field of the IPv4 header for this purpose. Using the ToS bits, you can define up to eight classes of service. Other features configured throughout the network can then use these bits to determine how to treat the packet in regard to the ToS to grant it. These other QoS features can assign appropriate traffic-handling policies, including congestion management strategy and bandwidth allocation. For example, queuing features such as LLQ can use the IP precedence setting of the packet to prioritize traffic.

IP Precedence Bits Used to Classify Packets

Use the three IP precedence bits in the ToS field of the IP header to specify the CoS assignment for each packet. As mentioned earlier, you can partition traffic into a maximum of eight classes and then use policy maps to define network policies in terms of congestion handling and bandwidth allocation for each class.

For historical reasons, each precedence corresponds to a name. These names are defined in RFC 791.

This table lists the numbers and their corresponding names, from least to most important.

Table 1: IP Precedence Values

Number	Name
0	routine

Number	Name
1	priority
2	immediate
3	flash
4	flash-override
5	critical
6	internet
7	network

The IP precedence feature allows you considerable flexibility for precedence assignment. That is, you can define your own classification mechanism. For example, you might want to assign precedence based on application or access router.

Note IP precedence bit settings 6 and 7 are reserved for network control information, such as routing updates.

Classification based on DEI

You can classify traffic based on the Drop Eligible Indicator (DEI) bit that is present in 802.1ad frames and in 802.1ah frames. Default DEI marking is supported. The set DEI action in policy maps is supported on 802.1ad packets for:

- Ingress
- Layer 2 sub-interfaces

Default DEI marking

Incoming packet	Default DEI on imposed 802.1ad headers
802.1 q packet	0
802.1 ad packet	DEI of the topmost tag of the incoming packet
802.1q packet translated to 802.1ad packet or 802.1ad packet	0 or 1 ; based on the DEI value of the set action

How to Configure Modular QoS Packet Classification

Creating a Classification Criterion

To create a classification criterion, containing match criteria, use the **class-map** command to specify the traffic class name, and then use the following **match** commands in class-map configuration mode, as needed.

Note Users can provide multiple values for a match type in a single line of configuration; that is, if the first value does not meet the match criteria, then the next value indicated in the match statement is considered for classification.

Restrictions

- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.

Procedure

Step 1 **configure**

Step 2 **class-map** [**type qos**] [**match-any**] *class-map-name*

Example:

```
RP/0/RP0:hostname(config)# class-map class201
```

Creates a class map to be used for matching packets to the class whose name you specify and enters the class map configuration mode.

If you specify **match-any**, one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class. This is the default. If you specify **match-all**, the traffic must match all the match criteria.

Step 3 **match cos** [*cos-value*] [*class-name*]

Example:

```
RP/0/RP0:hostname(config-cmap)# match cos 5
```

(Optional) Specifies a *cos-value* in a class map to match packets. The *cos-value* arguments are specified as an integer from 0 to 7.

Step 4 **match traffic-class** *class-name*

Example:

```
RP/0/RP0:hostname(config-cmap)# match traffic-class c1
```

(Optional) Specifies a *traffic class name*in a class map to match packets on an egress queuing policy. The arguments are specified as an integer from 0 to 7.

Step 5 **match dscp** [**ipv4** *dscp-value* [*dscp-value ... dscp-value*]

Example:

```
RP/0/RP0:hostname(config-cmap)# match dscp ipv4 15
```

(Optional) Identifies a specific DSCP value as a match criterion.

- Value range is from 0 to 63.
- Reserved keywords can be specified instead of numeric values.
- Up to eight values or ranges con be used per match statement.

Step 6 **match mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*]

Example:

```
RP/0/RP0:hostname(config-cmap)# match mpls experimental topmost 3
```

(Optional) Configures a class map so that the three-bit experimental field in the topmost Multiprotocol Label Switching (MPLS) labels are examined for experimental (EXP) field values. The value range is from 0 to 7.

Step 7 **match mpls experimental imposition** *value*

Example:

```
RP/0/RP0:hostname(config-cmap)# match mpls experimental imposition 5
```

(Optional) Configures a class map for the three-bit experimental field in the topmost Multiprotocol Label Switching (MPLS) imposition labels. The value range is from 0 to 7.

Step 8 **match precedence** [**ipv4**] *precedence-value* [*precedence-value1 ... precedence-value6*]

Example:

```
RP/0/RP0:hostname(config-cmap)# match precedence ipv4 5
```

(Optional) Identifies IP precedence values as match criteria.

- Value range is from 0 to 7.
- Reserved keywords can be specified instead of numeric values.

Step 9 **match qos-group** [*qos-group-value1 ... qos-group-value8*]

Example:

```
RP/0/RP0:hostname(config-cmap)# match qos-group 0 1 2 3 4 5 6 7
```

(Optional) Specifies service (QoS) group values in a class map to match packets.

- *qos-group-value* identifier argument is specified as the exact value or range of values from 0 to 7.
- Up to eight values (separated by spaces) can be entered in one match statement.
- **match qos-group** command is supported only for an egress marking policy.

Step 10 **commit**

Running configuration example for traffic class

```
class-map match-any CLASS_1_POLICERIPV4PREC
match precedence 1
end-class-map

class-map match-any class2
match qos-group 7
end-class-map
!
```

QoS over Bundle Interfaces

The Cisco NCS 4000 supports up to 30 unique policies distributed across 512 physical sub interfaces by default. While configuring QoS policies on bundle interfaces, the policer id is replicated on both the cores.

Scale Mode

The user can choose bundle or non-bundle scale requirements based on the three profiles. The following table displays the number of supported sub-interfaces, with QoS ingress policy, for physical and bundle interfaces.

Table 2: Supported Bundles based on Scale Mode

	Physical		**LAG**	
	Core	Card per Network Processor Unit	Core	Card per Network Processor Unit
Default	512	1024	0	0
High QoS LAG	64	128	448	448
Medium QoS LAG	256	512	256	256
Low QoS LAG	448	896	64	64

Restrictions

Restrictions for QoS on bundle interfaces:

- All ingress QoS configurations on a line card must be removed before the scale profile can be added or modified or deleted. If this is not done, the system can go in to an unpredictable and inconsistent state, and any such configuration attempt is not supported. To recover, if the currently configured ingress QoS scale is within the scale allowed by the newly configured scale mode, a LC hw-module reset can be done to recover the system.
- For bundle sub-interfaces whose parent interfaces are across multiple line cards, the maximum possible scale is limited by the least applicable scale-mode. If a bundle and its sub-interfaces have one or more

members going over a line card which does not have a valid scale-mode configured, then, ingress QoS configuration applied to it will be denied.

Configuring QoS on a Bundle Interface

The following configuration procedure enables the user to configure the qos policy over a bundle with a specific scale mode.

Procedure

Step 1 **configure**

Step 2 **hw-module profile qos bundle** [*scale-mode*] **location** *location-id*

Example:

```
RP/0/RP0:router(config)#hw-module profile qos bundle high-scale location 0/13
```

Enables a bundle on the router with the specified scale mode. The available options for scale mode are high-scale, medium-scale and low-scale. The number of supported sub-interfaces is based on the configured scale mode.

Note The **hw-module profile qos bundle** command is service-affecting. The following warning is displayed - *QoS profile CLI on any Line Card should be used only when no ingress QoS is configured on the same Line Card. Failure to do so can result in functionality breakage and system wide inconsistencies. In case of any misconfiguration, the scale mode config has to be reverted followed by LC hw-module reset to get the system back into original state.*

Step 3 **hw-module location** *location-id* **reload**

Example:

```
RP/0/RP0:router(config)#hw-module location 0/1 reload
```

Reloads the line card.

Step 4 **commit**

Configuring a Service Policy on a Bundle Interface

The following configuration procedure enables the user to attach a service policy to a bundle interface.

Procedure

Step 1 **config**

Step 2 **interface** *type interface-id* **l2transport**

Example:

```
RP/0/RP0:router(config)#interface bundle-ether 2000.1 l2 transport
```

Enters the interface configuration mode.

Step 3 **service-policy** [**input** | **output**] *policy-name*

Example:

```
RP/0/RP0:router(config-subif)#service-policy input policy-1
```

Attaches a service policy to the bundle interface.

Step 4 **commit**