Modular QoS Configuration Guide for Cisco ASR 9000 Series Routers, IOS XR Release 7.8.x

First Published: 2022-11-01

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

From Release 6.1.2 onwards, Cisco introduces support for the 64-bit Linux-based IOS XR operating system. Extensive feature parity is maintained between the 32-bit and 64-bit environments. Unless explicitly marked otherwise, the contents of this document are applicable for both the environments. For more details on Cisco IOS XR 64 bit, refer to the Release Notes for Cisco ASR 9000 Series Routers, Release 6.1.2 document.

This guide describes the Cisco IOS XR QoS configurations. The preface for the *Modular QoS Configuration Guide for Cisco ASR 9000 Series Routers* contains these sections:

Changes to this Document

This table lists the changes made to this document since it was first published.

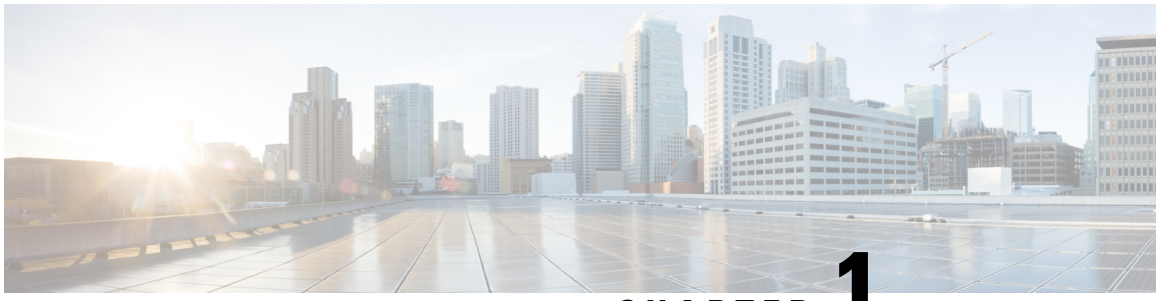
Date	Summary
November 2022	Initial release of this document.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business results you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco DevNet.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

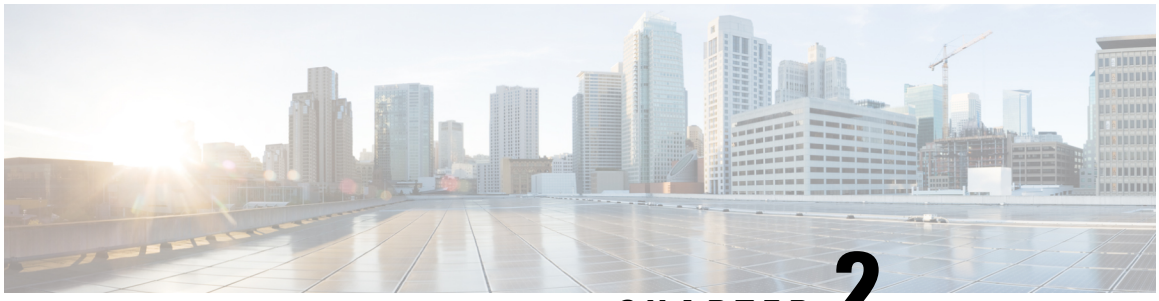
New and Changed QoS Features

New and Changed QoS Features

Table 1: QoS Features Added or Modified in IOS XR Release 7.8.x

Feature	Description	Changed in Release	Where Documented
None	No new features introduced	Not applicable	Not applicable

New and Changed QoS Features



CHAPTER 2

YANG Data Models for QoS Features

This chapter provides information about the YANG data models for QoS features.

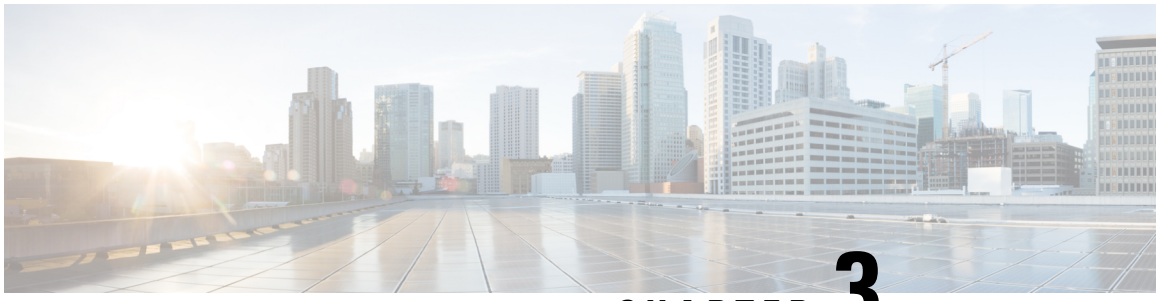
Using YANG Data Models

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the **Available-Content.md** file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.



CHAPTER 3

Modular QoS Overview

Quality of Service (QoS) is the technique of prioritizing traffic flows and providing preferential forwarding for higher-priority packets. The fundamental reason for implementing QoS in your network is to provide better service for certain traffic flows. A traffic flow can be defined as a combination of source and destination addresses, source and destination socket numbers, and the session identifier. A traffic flow can more broadly be described as a packet moving from an incoming interface that is destined for transmission to an outgoing interface. The traffic flow must be identified, classified, and prioritized on all routers and passed along the data forwarding path throughout the network to achieve end-to-end QoS delivery. The terms *traffic flow* and *packet* are used interchangeably throughout this module.

To implement QoS on a network requires the configuration of QoS features that provide better and more predictable network service by supporting bandwidth allocation, improving loss characteristics, avoiding and managing network congestion, metering network traffic, or setting traffic flow priorities across the network.

This module contains overview information about modular QoS features within a service provider network.

Information About Modular Quality of Service Overview

Before configuring modular QoS on your network, you must understand these concepts:

Benefits of Cisco IOS XR QoS Features

The Cisco IOS XR QoS features enable networks to control and predictably service a variety of networked applications and traffic types. Implementing Cisco IOS XR QoS in your network promotes these benefits:

- Control over resources. You have control over which resources (bandwidth, equipment, wide-area facilities, and so on) are being used. For example, you can limit bandwidth consumed over a backbone link by FTP transfers or give priority to an important database access.
- Tailored services. If you are an Internet Service Provider (ISP), the control and visibility provided by QoS enables you to offer carefully tailored grades of service differentiation to your customers.
- Coexistence of mission-critical applications. Cisco IOS XR QoS features ensure:
  - That bandwidth and minimum delays required by time-sensitive multimedia and voice applications are available.

- That your WAN is used efficiently by mission-critical applications that are most important to your business.
- That bandwidth and minimum delays required by time-sensitive multimedia and voice applications are available.
- That other applications using the link get their fair service without interfering with mission-critical traffic.

QoS Techniques

QoS on Cisco IOS XR software relies on these techniques to provide for end-to-end QoS delivery across a heterogeneous network:

- Packet classification and marking
- Congestion management
- Congestion avoidance

Before implementing the QoS features for these techniques, you should identify and evaluate the traffic characteristics of your network because not all techniques are appropriate for your network environment.

Packet Classification and Marking

Packet classification and marking techniques identify the traffic flow, and provide the capability to partition network traffic into multiple priority levels or classes of service. After classification is complete, any other QoS actions can be performed.

Identification of a traffic flow can be performed by using several methods within a single router, such as access control lists (ACLs), protocol match, IP precedence, IP differentiated service code point (DSCP), MPLS EXP bit, or Class of Service (CoS).

Marking of a traffic flow is performed by:

- Setting IP Precedence or DSCP bits in the IP Type of Service (ToS) byte.
- Setting (CoS, DEI) bits in the Layer 2 headers.
- Setting EXP bits within the imposed or the topmost Multiprotocol Label Switching (MPLS) label.
- Setting qos-group and discard-class bits.

Marking can be carried out:

- Unconditionally—As part of the class-action.
- Conditionally—As part of a policer-action.
- Combination of conditionally and unconditionally.

You can set a maximum of two fields per class-map. For example:

```
class c1
set cos 2
set dscp 21
```

You can place these two fields in any of the following combinations:

- Two sets per policer (conform/exceed/violate)
- Two sets without policing

For example:

```
Policy-map p1
class c1
     set dscp af11
     set cos 5
     police rate 10 mbps peak-rate 20 mbps
          conform-action set dscp af12
          conform-action set cos 6
          exceed-action set dscp af13
          exceed-action set cos 4
```

For detailed conceptual and configuration information about packet marking, see the “Configuring Modular Quality of Service Packet Classification on Cisco ASR 9000 Series Routers” module in this guide for unconditional marking, and the “Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers” module in this guide for conditional marking.

Default Marking Behavior

When an ingress or egress interface adds VLAN tags or MPLS labels, it requires a default value for the CoS and EXP values that go into those tags and labels. The default value can be then overridden based on the policy map. The default value for CoS and EXP is based on a trusted field in the packet upon ingress to the system. The router implements an implicit trust of certain fields based on the packet type and ingress interface forwarding type (Layer 2 or Layer 3).

By default, the router does not modify the IP precedence or DSCP without a policy-map being configured. The default behavior is described below.

On an ingress or egress Layer 2 interface, such as xconnect or bridge-domain, the outermost CoS value is used for any field that gets added in the ingress interface. If there is a VLAN tag that gets added due to a Layer 2 rewrite, the incoming outermost CoS value is used for the new VLAN tag. If an MPLS label is added, the CoS value would be used for the EXP bits in the MPLS tag.

On an ingress or egress Layer 3 interface (routed or label weighted for IPv4 or IPv6 packets), the three DSCP and precedence bits are identified in the incoming packet. For MPLS packets, the outermost label’s EXP bit is identified, and this value is used for any new field that gets added at the ingress interface. If an MPLS label is added, then the identified precedence, DSCP, or MPLS EXP value is used for the EXP bits in the newly added MPLS tag.

Provider Backbone Bridge (PBB) Configuration

In a PBB configuration, when a packet goes from a customer network to a service provider network using PBB encapsulation, the class of service (CoS) and discard eligibility indicator (DEI) used in the backbone VLAN tag (B-tag) and service instance tag (I-tag) of the PBB header is by default the CoS and DEI in the topmost tag of the incoming packet.

When a packet goes from a service provider to a customer network, the PBB header is removed and the I-tag CoS and DEI is used by default on any tags that are imposed on the customer interface. The default marking occurs only on imposed tags, and not on existing or translated tags.

Congestion Management

Congestion management techniques control congestion after it has occurred. One way that network elements handle an overflow of arriving traffic is to use a queuing algorithm to sort the traffic, then determine some servicing method of prioritizing it onto an output link.

Cisco IOS XR software implements the low-latency Queuing (LLQ) feature, which brings strict priority queuing (PQ) to the Modified Deficit Round Robin (MDRR) scheduling mechanism. LLQ with strict PQ allows delay-sensitive data such as voice, to be dequeued and sent before packets in other queues are dequeued.

Cisco IOS XR software includes traffic policing capabilities available on a per-class basis as well as class-based shaping.

The traffic policing feature limits the input or output transmission rate of a class of traffic based on user-defined criteria, and can mark packets by setting values such as IP Precedence, QoS group, or DSCP value.

Traffic shaping allows control over the traffic that leaves an interface to match its flow to the speed of the remote target interface and ensure that the traffic conforms to the policies contracted for it. Thus, traffic adhering to a particular profile can be shaped to meet downstream requirements, thereby eliminating bottlenecks in topologies with data-rate mismatches.

Cisco IOS XR software supports a class-based traffic shaping method through a CLI mechanism in which parameters are applied per class.

For detailed conceptual and configuration information about congestion management, see the “Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers” module.

Congestion Avoidance

Congestion avoidance techniques monitor network traffic flows in an effort to anticipate and avoid congestion at common network and internetwork bottlenecks before problems occur. These techniques are designed to provide preferential treatment for traffic (such as a video stream) that has been classified as real-time critical under congestion situations while concurrently maximizing network throughput and capacity utilization and minimizing packet loss and delay. Cisco IOS XR software supports the Random Early Detection (RED), Weighted RED (WRED), and tail drop QoS congestion avoidance features.

For detailed conceptual and configuration information about congestion avoidance techniques, see the “Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers” module in this guide.

Differentiated Service Model for Cisco IOS XR Software

Cisco IOS XR software supports a differentiated service that is a multiple-service model that can satisfy different QoS requirements. However, unlike in the integrated service model, an application using differentiated service does not explicitly signal the router before sending data.

For differentiated service, the network tries to deliver a particular kind of service based on the QoS specified by each packet. This specification can occur in different ways, for example, using the IP Precedence bit settings in IP packets or source and destination addresses. The network uses the QoS specification to classify, mark, shape, and police traffic, and to perform intelligent queuing.

The differentiated service model is used for several mission-critical applications and for providing end-to-end QoS. Typically, this service model is appropriate for aggregate flows because it performs a relatively coarse level of traffic classification.

Access Node Control Protocol

Access Node Control Protocol (ANCP) creates a control plane between a service-oriented aggregation device and an access node (AN) (for example, a DSLAM) in order to perform QoS-related, service-related, and subscriber-related operations. An ANCP Network Access Server (NAS) accepts and maintains ANCP adjacencies (sessions with an ANCP neighbor), and sending and receiving ANCP messages.

ANCP allows static mapping between AN ports and VLAN subinterfaces so that DSL rate updates for a specific subscriber received by the ANCP server are applied to the QoS configuration corresponding to that subscriber. DSL train rates received via ANCP are used to alter shaping rates on subscriber-facing interfaces and subinterfaces on the router.

Additional Cisco IOS XR QoS Supported Features

These sections describe the additional features that play an important role in the implementation of QoS on Cisco IOS XR software.

Modular QoS Command-Line Interface

In Cisco IOS XR software, QoS features are enabled through the Modular QoS command-line interface (MQC) feature. The *MQC* is a command-line interface (CLI) structure that allows you to create policies and attach these policies to interfaces. A traffic policy contains a traffic class and one or more QoS features. A traffic class is used to classify traffic, whereas the QoS features in the traffic policy determine how to treat the classified traffic. One of the main goals of MQC is to provide a platform-independent interface for configuring QoS across Cisco platforms.

For detailed conceptual and configuration information about the MQC feature, see the “Configuring Modular Quality of Service Packet Classification on Cisco ASR 9000 Series Routers” module in this guide.

Fabric QoS

There is no separate configuration for fabric QoS. The fabric priority is derived from the priority action in the ingress service policy.

Where to Go Next

To configure the packet classification features that involve identification and marking of traffic flows, see the Configuring Modular Quality of Service Packet Classification module in this guide.

To configure the queuing, scheduling, policing, and shaping features, see the Configuring Modular Quality of Service Congestion Management module in this guide.

To configure the WRED and RED features, see the Configuring Modular QoS Congestion Avoidance module in this guide.

To configure Access Node Control Protocol (ANCP) features, see the “Configuring Access Node Control Protocol on Cisco ASR 9000 Series Routers” module in this guide.

Additional References

The following sections provide references related to implementing QoS.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Getting Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Modular of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Route of Cisco Cisco ASR 9000 Series Aggregation Services Route Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

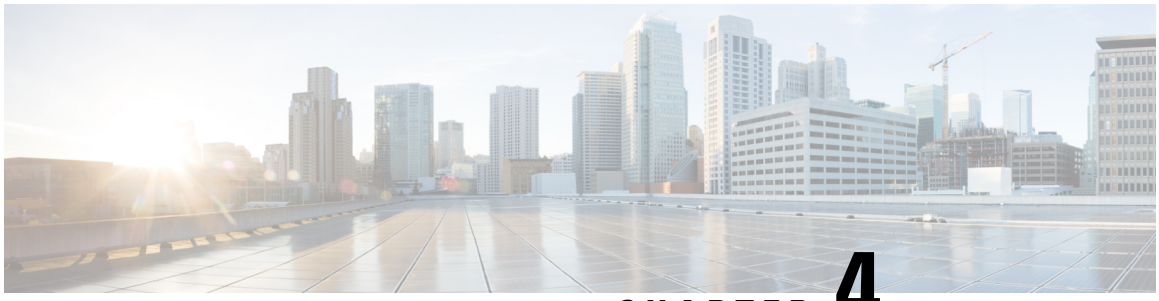
MIBs	MIBs Link
CISCO-CLASS-BASED-QOS-MIB	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 4

Configuring Access Node Control Protocol

Access Node Control Protocol (ANCP) creates a control plane between a service-oriented aggregation device and an access node (AN) (for example, a DSLAM) in order to perform QoS-related, service-related, and subscriber-related operations. An ANCP server accepts and maintains ANCP adjacencies (sessions with an ANCP neighbor), and sending and receiving ANCP messages. ANCP allows static mapping between ANCP ports and VLAN subinterfaces so that DSL rate updates for a specific subscriber received by the ANCP server are applied to the QoS configuration corresponding to that subscriber. DSL train rates received via ANCP are used to alter shaping rates on subscriber-facing interfaces and subinterfaces on the router. ANCP runs as a single process on the route processor (RP).

This module provides the conceptual and configuration information for implementing ANCP.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Access Node Control Protocol	yes	no

Feature History for Configuring Access Node Protocol on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.2	The Access Node Control Protocol feature was introduced.
Release 3.9.0	Mapping of ANCP ports to VLAN interfaces over Ethernet bundles was added.
Release 4.0.0	ANCP over Multi Chassis Link Aggregation was introduced.

Prerequisites for Configuring ANCP

Restrictions for Configuring ANCP

The following restrictions apply when configuring ANCP on your network:

- Only Rate Adaptive Mode is supported in Cisco IOS XR Release 3.7.2.
- VPN routing and forwarding (VRF) awareness is not supported in Cisco IOS XR Release 3.7.2. All IP interfaces receiving ANCP traffic should be in default VRF.
- ANCP over IPv6 is not supported for Cisco IOS XR Release 3.7.2.
- Only VLAN subinterfaces over Ethernet and Ethernet bundle ports can be mapped to AN ports using ANCP.

Information About Configuring ANCP

To implement ANCP, you must understand the following concepts:

ANCP Adjacencies

The ANCP server accepts TCP connections from access nodes. An ANCP neighbor is any access node that establishes an adjacency with an ANCP server. ANCP is configured globally, and as long as it is IP-enabled, there is no restriction on whether ANCP messages are received on the physical or logical interface.

TCP creates a separate connection socket for each access node. Because access nodes are not identified explicitly in ANCP messages, the TCP socket serves as the ANCP neighbor identifier for the ANCP server.

Once the TCP connection between ANCP neighbors has been made, the ANCP adjacency protocol establishes an ANCP session over that connection and negotiates ANCP capabilities. There is a single ANCP session per ANCP neighbor. ANCP session information becomes a subset of the information of a corresponding neighbor.

ANCP protocol supports dynamic neighbor detection so no configuration of access nodes is required. ANCP neighbors can also be statically preconfigured on the ANCP server. In such a case, access nodes are explicitly identified by their IDs, which then must match the **sender-name** field in the ANCP adjacency protocol messages.

Neighbor Adjacency Timing

The adjacency timer defines the maximum delay between different stages of ANCP session establishment and the period of ANCP keepalive.

ANCP adjacency lifetime is governed by the adjacency protocol. If synchronization with the peer access node is lost (for example, if the adjacency dead timer expires), the ANCP server removes the adjacency, and the underlying TCP connection is closed.

ANCP Messages

Two ANCP message types are processed by the ANCP server: Port Up and Port Down. Port Up messages contain DSL rate information; Port Down messages indicate that the corresponding access line is no longer available. DSL rate updates from Port Up messages are made available to the QoS subsystem. Port Down messages are used to internally track the ANCP port state.

These messages can only be received by the server after the ANCP adjacency is established. However, once a Port Up message is received, the DSL rate information it contains is considered valid indefinitely, provided AN-port-to-interface mapping is configured for that port. It is stored in the AN port database until it is overwritten by another Port Up message for this port or is cleared manually. The removal of an adjacency or the reception of a Port Down message is reflected in the database for display and troubleshooting purposes, but DSL rate information is not invalidated.

Port Mapping

AN ports are statically mapped to VLAN subinterfaces, referred to as AN-port-to-interface mapping. This implies that there is at least one VLAN subinterface configured per subscriber line. There is no limit to the number of interfaces that can be mapped to an AN port.

VLAN subinterfaces mapped to an AN port can be created or removed. When mapping is configured, VLAN subinterfaces are referenced in the ANCP module by name. This name is used for notifications of interface creation and deletion and provides the information that is used in updating the DSL rate.

An AN port database is maintained for all ports learned from Port Up messages. This database also contains the AN-port-to-interface mapping database. If a Port Up message for an AN port arrives but no interface is mapped to that port, the rate information is stored in the AN port database but not published. When a mapping for that port is configured, the AN port database is scanned to identify any ANCP messages that were received on this port prior to the mapping configuration. If there were, the known rate is published.

Rate Adjustment

ANCP can apply a correction factor to the DSL line rate reported in Port Up messages before publishing the rate update to the system. This correction factor or rate adjustment is configurable in the global configuration mode per DSL type and access encapsulation type (ATM or Ethernet). DSL type and encapsulation type are provided in mandatory type, length, and value (TLV) data in the Port Up message.

Note To use the rate adjustment feature for non-default loop types (Ethernet), DSLAMs must support the optional Access Loop Encapsulation sub-TLV.

ANCP rate-adaptive mode information is processed by the ANCP module to determine the maximum bandwidth (shape rate) available for a given subscriber line. A fixed correction factor is then applied to the ANCP bandwidth based on the DSL type to account for the overhead of different DSL technologies. For example, a given subscriber's ANCP bandwidth may be 15 Mbps, but due to the DSL technology overhead, the effective bandwidth for that subscriber should be limited to 80 percent of 15 Mbps, which is 12 Mbps. This corrected effective bandwidth is conveyed to QoS modules to limit the maximum rate for the subscriber's traffic.

Note The ANCP rate is used as a QoS shaping rate only if the ANCP rate is greater than the currently configured QoS shaping rate. (The ANCP rate used by QoS is rounded down to the nearest 128 kbps.)

Prioritization of ANCP Traffic

In case of congestion, the Cisco ASR 9000 Series Router marks ANCP messages as high priority so that the aggregation network between the Network Access Server (NAS) and the access node (AN) can prioritize the ANCP messages ahead of other traffic.

Process Restart

During a process restart, TCP connections with ANCP neighbors normally drop. When the ANCP server comes back, TCP connections and ANCP sessions are reestablished by the neighbors. Upon reconnecting to the server, DSLAMs send Port Up messages for every active port. Any published rate information received prior to restart is restored in the ANCP configuration. If the restart occurred due to a crash, conflicts between published data and configuration data are detected and published data is corrected.

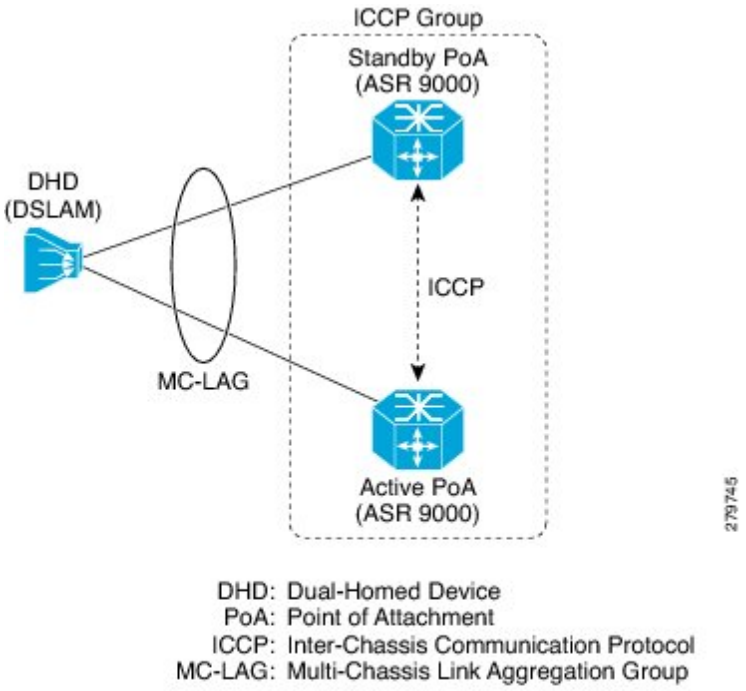
ANCP and QoS Interaction

When the ANCP value is applied correctly, it overrides the configured QoS shaper value. For an example of an ANCP value applied incorrectly and an example of the interaction with QoS when the ANCP value is applied correctly, see "ANCP and QoS Interaction: Example" section on page 105.

Multi Chassis Link Aggregation

Multi Chassis Link Aggregation (MC-LAG) provides a simple redundancy mechanism for a Digital Subscriber Line Access Multiplier (DSLAM) to Cisco ASR 9000 Series Router connection. The redundancy is achieved by allowing a dual-homed connection to two routers. There is no added software complexity on the DSLAM, because the DSLAM views the dual-homed connection as a single LAG. The DSLAM is known as a dual-homed device (DHD), and each router is known as a point of attachment (PoA) in MC-LAG terminology. For more detailed information about MC-LAG, see the *Cisco ASR 9000 Series Aggregation Services Router L2VPN and Ethernet Services Configuration Guide*.

Figure 1: MC-LAG connects DSLAM to ASR 9000 Series Routers



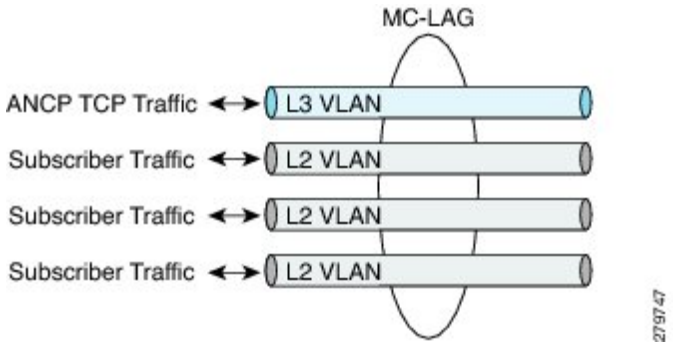
ANCP over MC-LAG

Access Node Control Protocol (ANCP) is required to support a network topology that includes MC-LAG connections to DSLAMs. CPE circuits connect to DSLAMs and adjust line speeds based on signal quality with Rate Adaptive DS. Uplinks connect DSLAMs to routers. If the line speed of a circuit adjusts to a lower data rate than the uplink, subscriber data can be lost on the DSLAM. To prevent data loss, a DSLAM notifies the router of the new DSL rate with ANCP, and downstream shaping is dynamically applied on the router such that the data rate of the uplink does not exceed the CPE circuit data rate.

ANCP applies DSLAM subscriber circuit DSL rate data it learns, to MC-LAG VLAN subinterfaces that are mapped to the subscriber circuit. The rates are applied to QoS shapers. The DSL rates that ANCP has applied to the MC-LAG VLAN subinterfaces are distributed by the ANCP application running on the active PoA for the MC-LAG to the ANCP application that is running on the standby PoA for the MC-LAG, using ICCP (Inter-Chassis Communication Protocol). ANCP on the standby PoA for the MC-LAG applies the DSL rate data to the corresponding MC-LAG VLAN subinterfaces. When an event occurs that causes one of the standby PoAs to assume the active role for the MC-LAG, the ANCP application on the newly active PoA has already applied the DSL rates to shapers on the MC-LAG VLAN subinterfaces, so the correct DSL rates are applied when this LAG goes active and congestion and subsequent data loss does not occur at the DSLAM.

A DSLAM establishes an ANCP adjacency with a router over a TCP connection. The DSL rates for the DSLAM subscriber circuits are communicated over this TCP connection. The DSL rates are applied to Layer 2 VLAN subinterfaces that are mapped to the subscriber circuits. The ANCP TCP connection that is used to send DSL rates for Layer 2 VLAN subinterfaces on an MC-LAG must be on a Layer 3 VLAN subinterface that is in the same MC-LAG as the L2VLAN subinterfaces. Note that this constraint implies that there is one ANCP TCP connection between the DSLAM and router per MC-LAG.

Figure 2: ANCP over MC-LAG VLAN Subinterfaces



When an active PoA for a MC-LAG becomes the standby, the DSLAM ANCP TCP connection is terminated. The DSLAM re-establishes the ANCP TCP connection with the PoA that assumes the active role for the MC-LAG.

How to Configure ANCP on Cisco

This section contains instructions for the following tasks:

- Enabling ANCP, page 88
- Configuring ANCP Server Sender Name, page 90
- Configuring ANCP Neighbors, page 91
- Mapping AN Ports to VLAN Subinterfaces, page 94
- Configuring ANCP Rate Adjustment, page 96

Enabling ANCP

To enable ANCP, use the **ancp** command in global configuration mode.

Prerequisites

To use this command, you must be in a user group associated with a task group that includes the proper task IDs for ANCP.

SUMMARY STEPS

1. **configure** RP/0/RSP0/CPU0:router# configure RP/0/RSP0/CPU0:router(config)#
2. **ancp** RP/0/RSP0/CPU0:router(config)# ancp
3. **end**
4. or **commit**
5. **show ancp summary [statistics][detail]** RP/0/RSP0/CPU0:router# show ancp summary

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** RP/0/RSP0/CPU0:router# configure RP/0/RSP0/CPU0:router(config)#	Enters global configuration mode.
Step 2	**ancp** RP/0/RSP0/CPU0:router(config)# ancp	Enables ANCP.
Step 3	**end**	

Step 4	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ancp)# end`
	or
	`RP/0/RSP0/CPU0:router(config-ancp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 5	**show ancp summary [statistics][detail]** RP/0/RSP0/CPU0:router# show ancp summary	(Optional) Displays ANCP summary and general configuration information.

Configuring ANCP Server Sender Name

The ANCP server sender name is used by the ANCP server in adjacency protocol messages to DSLAMs.

SUMMARY STEPS

1. **configure** RP/0/RSP0/CPU0:router# **configure** RP/0/RSP0/CPU0:router(config)#
2. **ancp server sender-name** {H.H.H | A.B.C.D} RP/0/RSP0/CPU0:router(config)# **ancp server sender-name 0013.1aff.c2bd**
3. **end**
4. or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** RP/0/RSP0/CPU0:router# **configure** RP/0/RSP0/CPU0:router(config)#	Enters global configuration mode.
Step 2	**ancp server sender-name** {H.H.H | A.B.C.D} RP/0/RSP0/CPU0:router(config)# **ancp server sender-name 0013.1aff.c2bd**	Configures a local sender name.
Step 3	**end**	

Step 4	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ancp)# end`
	or
	`RP/0/RSP0/CPU0:router(config-ancp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring ANCP Neighbors

The TCP connection from any neighbor is accepted on any interface. To match a neighbor configuration to a respective TCP connection, ANCP neighbors are identified by a sender name that must match the corresponding field in adjacency protocol messages. Optionally, a description string can be supplied to identify the ANCP neighbor on the system and an adjacency timer interval configured.

SUMMARY STEPS

1. **configure**
2. **ancp** neighbor sender-name {H.H.H | A.B.C.D}[**description** string]
3. **ancp neighbor sender-name** {*H.H.H* | *A.B.C.D*} [**adjacency-timer** *interval*]
4. **end** or **commit**

5. show ancp neighbor {**description** *description-string*| **sender-name** {*H.H.H* | *A.B.C.D*}} [**statistics**][**detail**] RP/0/RSP0/CPU0:router# show ancp neighbor sender-name 0006.2aaa.281b
6. show ancp neighbor summary [**statistics**][**detail**] RP/0/RSP0/CPU0:router# show ancp neighbor summary
7. clear ancp neighbor {**all** | **description** description-string | **sender-name** {H.H.H | A.B.C.D}}[**state** | **statistics**] RP/0/RSP0/CPU0:router# clear ancp neighbor all
8. clear ancp summary [statistics | detail] RP/0/RSP0/CPU0:router# clear ancp summary statistics
9. show ancp neighbor [all] [statistics] RP/0/RSP0/CPU0:router# show ancp neighbor statistics
10. show ancp neighbor state [none | synsent | synrcvd | estab} [statistics] RP/0/RSP0/CPU0:router# show ancp neighbor none

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	**configure**
	Example:
	`RP/0/RSP0/CPU0:router# configure`
	`RP/0/RSP0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	**ancp** neighbor sender-name {H.H.H | A.B.C.D}[**description** string] **Example:** `RP/0/RSP0/CPU0:router(config)# ancp neighbor sender-name oo13.1aff.c2bd description vendorA1`	Sets neighbor description parameter to easily identify DSLAMs.
Step 3	**ancp neighbor sender-name** {*H.H.H* | *A.B.C.D*} [**adjacency-timer** *interval*] **Example:** `RP/0/RSP0/CPU0:router(config)# ancp neighbor sender-name 0013.1aff.c2bd adjacency-timer 20`	Sets neighbor adjacency timer parameter. If a neighbor session is already established, it will be reset so this timer can take affect. **Note** • Configured ports are placed in a down state while unconfigured ports are released.

Step 4	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ancp)# end`
	or
	`RP/0/RSP0/CPU0:router(config-ancp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.
Step 5	show ancp neighbor {**description** *description-string*| **sender-name** {*H.H.H* | *A.B.C.D*}} [**statistics**][**detail**] RP/0/RSP0/CPU0:router# show ancp neighbor sender-name 0006.2aaa.281b	(Optional) Displays data or message statistics associated with individual ANCP adjacencies or sets of adjacencies.
Step 6	show ancp neighbor summary [**statistics**][**detail**] RP/0/RSP0/CPU0:router# show ancp neighbor summary	(Optional) Displays adjacency counts by state.
Step 7	clear ancp neighbor {**all** | **description** description-string | **sender-name** {H.H.H | A.B.C.D}}[**state** | **statistics**] RP/0/RSP0/CPU0:router# clear ancp neighbor all	(Optional) Clears ANCP neighbors, either all or individually. Configured ports are placed in a down state while releasing unconfigured ports. If state is selected, the adjacency is reset without clearing the TCP socket.
Step 8	clear ancp summary [statistics | detail] RP/0/RSP0/CPU0:router# clear ancp summary statistics	(Optional) Clears aggregate message statistics only, without modifying individual neighbor or port statistics.
Step 9	show ancp neighbor [all] [statistics] RP/0/RSP0/CPU0:router# show ancp neighbor statistics	(Optional) Displays ANCP neighbor information.
Step 10	show ancp neighbor state [none | synsent | synrcvd | estab} [statistics] RP/0/RSP0/CPU0:router# show ancp neighbor none	(Optional) Displays adjacency protocol state information.

Mapping AN Ports to VLAN Subinterfaces

Port mapping associates DSLAM access ports or customer premises equipment (CPE) clients of a DSLAM with VLAN subinterfaces. The VLANs can be IEEE 802.1Q or QinQ hierarchical VLANs. To map AN ports to VLAN subinterfaces, use the **ancp an-port** command in global configuration mode.

SUMMARY STEPS

1. **configure**
2. ancp an-port circuit-id *Access-Loop-Circuit*-ID [**interface** type interface-path-id | **interface Bundle-Ether** *bundle-id*] RP/0/RSP0/CPU0:router(config)# ancp an-port circuit-id circuit1 interface gigabitethernet 2/0/1/1.1
3. **end** or **commit**
4. show ancp an-port {circuit-id *Access-Loop-Circuit-ID* | **interface** type interface-path-id | **interface Bundle-Ether** *bundle-id* | **mapping} [statistics** | detail]
5. show ancp an-port [configured | dynamic-only][statistics]

6. show ancp an-port summary [**statistics**][**detail**]
7. **clear ancp an-port** {**all** | **circuit-id** Access-Loop-Circuit-Id | **interface** type interface-path-id | **interface Bundle-Ether** *bundle-id* | **neighbor** {**description** string | **sender-name** {H.H.H | A.B.C.D}}[**statistics**]
8. **show ancp an-port** {**description** description-string | **sender-name** {*H.H.H* | *A.B.C.D*}}
9. **show ancp an-port state** [**up** | **down** | **none**] [**statistics**]

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	**configure**
	Example:
	`RP/0/RSP0/CPU0:router# configure`
	`RP/0/RSP0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	ancp an-port circuit-id *Access-Loop-Circuit*-ID [**interface** type interface-path-id | **interface Bundle-Ether** *bundle-id*] RP/0/RSP0/CPU0:router(config)# ancp an-port circuit-id circuit1 interface gigabitethernet 2/0/1/1.1	Defines a unique access node ID. This ID information is included in the ANCP Port Up and Port Down messages. The Circuit ID must be supplied before the access node port configuration can be committed. When using a shared policy instance in subinterfaces with ANCP, the same AN port circuit ID must be mapped to all subinterfaces that have the same shared policy instance.

Step 3	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ancp)# end`
	or
	`RP/0/RSP0/CPU0:router(config-ancp)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

	Command or Action	Purpose
Step 4	show ancp an-port {circuit-id *Access-Loop-Circuit-ID* | **interface** type interface-path-id | **interface Bundle-Ether** *bundle-id* | **mapping} [statistics** | detail] **Example:** `RP/0/RSP0/CPU0:router# show ancp an-port gigabitethernet 2/0/1/1.1`	(Optional) Displays information about the association of DSLAM access ports (or CPE clients of a DSLAM) with VLAN subinterfaces.
Step 5	show ancp an-port [configured | dynamic-only][statistics] **Example:** `RP/0/RSP0/CPU0:router# show ancp an-port configured`	(Optional) Displays summary data or statistics for AN ports that are or are not mapped to interfaces.
Step 6	show ancp an-port summary [**statistics**][**detail**] **Example:** `RP/0/RSP0/CPU0:router# show ancp an-port summary`	(Optional) Displays port counts by state.
Step 7	**clear ancp an-port** {**all** | **circuit-id** Access-Loop-Circuit-Id | **interface** type interface-path-id | **interface Bundle-Ether** *bundle-id* | **neighbor** {**description** string | **sender-name** {H.H.H | A.B.C.D}}[**statistics**] **Example:** `RP/0/RSP0/CPU0:router# clear ancp an-port all`	(Optional) Clears AN ports of dynamic data or statistics either individually or in groups. Published information is cleared and information learned from the DSLAM is cleared.
Step 8	**show ancp an-port** {**description** description-string | **sender-name** {*H.H.H* | *A.B.C.D*}} **Example:** `RP/0/RSP0/CPU0:router# show ancp an-port description vendor3b`	(Optional) Displays AN port information.
Step 9	**show ancp an-port state** [**up** | **down** | **none**] [**statistics**] **Example:** `RP/0/RSP0/CPU0:router# show ancp an-port state up`	(Optional) Displays AN port state information.

Configuring ANCP Rate Adjustment

Use the **ancp rate-adjustment** command to apply a mathematical correction to the ANCP rate update prior to applying it as a shaper rate.

SUMMARY STEPS

1. **configure** RP/0/RSP0/CPU0:router# **configure** RP/0/RSP0/CPU0:router(config)#

2. **ancp rate-adjustment** dsl-type **access-loop-type percent-factor** factor
3. **end** or **commit**
4. show ancp summary detail RP/0/RSP0/CPU0:router# show ancp summary detail

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** RP/0/RSP0/CPU0:router# **configure** RP/0/RSP0/CPU0:router(config)#	Enters global configuration mode.
Step 2	**ancp rate-adjustment** dsl-type **access-loop-type percent-factor** factor **Example:** `RP/0/RSP0/CPU0:router(config)# ancp rate-adjustment adsl2 ethernet percent-factor 90`	Sets the parameters for the ANCP shaper percent factor. *dsl-type* and *access-loop-type* are compared to appropriate values in optional type-length values (TLVs) in the ANCP Port Up message and the ANCP rate is adjusted by a configured factor in case of a match. • dsl-type—(Required) Sets DSL type code: **adsl1 adsl2 adsl2+ vdsl1 vdsl2 sdsl** • *access-loop-type*—(Required) Sets *access-loop-type* to **ATM** or **Ethernet**. • **percent-factor** factor—(Required) A percent value to be applied to the ANCP reported rate update prior to configuring it as a shaping rate.

Step 3	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config)# end`
	or
	`RP/0/RSP0/CPU0:router(config)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

	Command or Action	Purpose
Step 4	show ancp summary detail RP/0/RSP0/CPU0:router# show ancp summary detail	(Optional) Shows generic ANCP configuration information along with rate adjustment configuration information.

Configuration Examples for Configuring ANCP contains the following examples:

Configuring ANCP Server Sender Name: Example

Configuring ANCP Neighbors: Example

The following example shows how to set ANCP neighbor parameters:

```
configure
ancp neighbor sender-name 0001.2222.3333 description VendorA-1
ancp neighbor sender-name 0001.2222.3333 adjacency-timer 20

commit
```

The following example shows the output from a specific neighbor using the **sender-name** MAC address:

```
show ancp neighbor sender-name 0006.2aaa.281b

            ANCP Neighbor Data
  -----------------------------------------
  Sender Name              0006.2aaa.281b
  Description              first
  State                    ESTAB
  Capability               Topology Discovery
  Ports:
    State Up               25
    State Down             5
    Total                  30
```

The following example shows the same command with the addition of the **detail** keyword, showing a summary of AN ports that were reported by that neighbor:

```
show ancp neighbor sender-name 0006.2aaa.281b detail
```

```
            ANCP Neighbor Data
  ------------------------------------------
  Sender Name               0006.2aaa.281b
  Description               first
  State                     ESTAB
  Capability                Topology Discovery
  Ports:
    State Up                4
    State Down              0
    Total                   4
Remote IP Addr/TCP Port   209.165.200.225/11126
Local  IP Addr/TCP Port   209.165.200.250/6068
Server Sender Name        0013.1aff.c2bd
Remote Timeout            25500 msec
Local  Timeout            10000 msec
Adjacency Uptime          01:25:20
Time Since Last Port Msg  00:00:04
Remote Port               0
Remote Instance           1
Local Instance            1
Remote Partition ID       0

List of AN port data for neighbor sender name 0006.2aaa.281b
------------------------------ ----- ---------- -------- ----  ------------
                                                 Line     Num   Adjusted DS
Circuit-id                     State Uptime     State    Intf  Rate (kbps)
------------------------------ ----- ---------- -------- ----  ------------
circuit1                       UP    00:27:49   SHOWTIME   3    2250
circuit2                       UP    00:00:49   SHOWTIME   2    2250
circuit3                       UP    00:00:49   SHOWTIME   2    2250
circuit4                       UP    00:00:49   SHOWTIME   0    2250
```

The following example shows the same command, this time with the addition of the **statistics** keyword, showing a summary of message statistics for the selected neighbor:

```
show ancp neighbor sender-name 0006.2aaa.281b statistics

ANCP Neighbor Message Statistics
for Sender-name -, Description 0006.2aaa.281b
-----------------------------------------------
              Sent         Received
SYN           1            2
SNYACK        1            0
ACK           589          238
RSTACK        0            0
Port Up       -            10
Port Down     -            0
Drops         0            0
Total         600          250
```

The following example shows how to display generic information about ANCP configuration, along with neighbor and port counts by state:

```
show ancp summary

 ANCP Summary Information
 ------------------------------------------
 Capability:                Topology Discovery
 Server sender-name:        0013:1aff.c2bd

 Neighbor count by state:
   -                          0
   SYNSENT                    0
```

```
   SUNRCVD                      0
   ESTAB                        1
   ---------------------------------
   Total                        1

Port count by state:
  State Up                    1
  State Down                  0
  State Unknown               0
  ---------------------------------
  Total                       1

  No. configured ports        1
  No. mapped sub-interfaces   4
```

The following example shows how to display rate adjustment configuration information in addition to the generic information shown in the previous example:

```
show ancp summary detail

    ANCP Summary Information
   ---------------------------------------------
   Capability:                Topology Discovery
   Server sender-name:        0013:1aff.c2bd

Neighbor count by state:
  -                           0
  SYNSENT                     0
  SUNRCVD                     0
  ESTAB                       1
  ---------------------------------
  Total                       1
Port count by state:
  State Up                    1
  State Down                  0
  State Unknown               0
  ---------------------------------
  Total                       1

  No. configured ports        1
  No. mapped sub-interfaces   4

Rate adjustment configuration:
-----------------------------------------
DSL Type   Loop Type        Percent-Factor
-----------------------------------------
ADSL1      ETHERNET               90
ADSL2      ETHERNET               100
ADSL2PLUS  ETHERNET               100
VDSL1      ETHERNET               100
VDSL2      ETHERNET               100
SDSL       ETHERNET               100
ADSL1      ATM                    100
ADSL2      ATM                    100
ADSL2PLUS  ATM                    100
VDSL1      ATM                    100
VDSL2      ATM                    100
SDSL       ATM                    100
```

The following example shows how to display a summary of ANCP message statistics:

```
show ancp summary statistics
```

```
 ANCP Summary Message Statistics
-------------------------------------
              Sent          Received
SYN           3             6
SYNACK        4             0
ACK           7105          2819
RSTACK        2             0
Port Up       -             6
Port Down     -             0
Drops         0             0
Total         7114          2831
```

The following example shows how to clear all neighbor data and statistics:

```
clear ancp neighbor all
```

The following example shows how to clear a specific neighbor:

```
clear ancp neighbor description vendor1a
```

The following example shows how to clear aggregate message statistics:

```
clear ancp summary statistics
```

Mapping AN ports to VLAN Subinterfaces: Example

The following example shows a unique access node ID being defined:

```
configure
ancp an-port circuit-id circuit1 interface gigabitethernet 2/0/1/1.1
```

The following example shows how to display information for a port identified by its subinterface:

```
show ancp an-port interface gigabitethernet 0/0/0/37.1
```

```
  AN port circuit-id ccc1:

  State                                                 UP
  UPtime                                                02:23:45
  Time Since Last Message                               00:00:00
  Encap Type                                            ETHERNET
  DSL type                                              ADSL1
  DSL Line State                                        SHOWTIME
  Number of Mapped Interfaces                           3
  Neighbor sender-name                                  0006.2aaa.281b
  Neighbor description                                   7200-client
  Configured Rate Adjustment                            90%
  Actual Downstream Data Rate (kbps)                    2500
  Effective Downstream Data Rate (kbps)                 2250
```

The following example shows how use the **detail** keyword to display port information as well as a list of the interfaces mapped to that port.

```
show ancp an-port circuit-id ccc1 detail
```

```
  AN port circuit-id ccc1:

  State                                                 UP
  UPtime                                                02:31:36
```

```
  Time Since Last Message                                  00:00:00
  Encap Type                                               ETHERNET
  DSL type                                                 ADSL1
  DSL Line State                                           SHOWTIME
  Number of Mapped Interfaces                              3
  Neighbor sender-name                                     0006.2aaa.281b
  Neighbor description                                      7200-client
  Configured Rate Adjustment                               90%
  Actual Downstream Data Rate (kbps)                       2500
  Effective Downstream Data Rate (kbps)                    2250
  Actual Data Rate Upstream/Downstream (kbps)              2500/2500
  Minimum Data Rate Upstream/Downstream (kbps)             0/0
  Attainable Data Rate Upstream/Downstream (kbps)          0/0
  Maximum Data Rate Upstream/Downstream (kbps)             0/0
  Minimum Low Power Data Rate Upstream/Downstream (kbps)   0/0
  Maximum Interleaving delay Upstream/Downstream (ms)      0/0
  Actual Interleaving Delay Upstream/Downstream (ms)       0/0

  Sub-interface Summary: total 3
  ------------------------------------------------
  Sub-interface Name                   ifhandle
  ---------------------------------    ----------
  GigabitEthernet0/0/0/37.1            0x0
  GigabitEthernet0/0/0/37.11           0x0
  GigabitEthernet0/0/0/38.10           0xb80
```

The following example uses the **statistics** keyword to display port message statistics for a specific AN port:

```
show ancp an-port circuit-id ccc1 statistics

  Port message statistics for circuit-id ccc1:

    Port Up       5
    Port Down     0
```

The following example shows how to display port counts by state:

```
show ancp an-port summary

     AN Port Count Summary
  ------------------------------
  State UP                 4
  State DOWN               0
  Config only ports        0
  Total                    4
  # Configured ports       1
  # Mapped sub-interfaces  4
```

The following example shows how to clear message statistics for all AN ports:

```
clear ancp an-port all statistics
```

The following example shows how to clear dynamic data for all AN ports:

```
clear ancp an-port all
```

The following example show how to clear dynamic data for a specific interface:

```
clear ancp an-port interface gigabitethernet 0/1/0/10.5
```

Configuring ANCP Rate Adjustment: Example

ANCP and QoS Interaction: Example

The following example shows a hierarchical QoS policy configuration with and without an ANCP value applied:

```
policy-map child-3play
 class 3play-voip
  priority level 1
  police rate 65 kbps
  !
 !
 class 3play-video
  priority level 2
  police rate 128 kbps
  !
  random-detect cos 3 10 ms 100 ms
  random-detect cos 4 20 ms 200 ms
 !
 class 3play-premium
  bandwidth percent 100
 !
 class class-default
 !
 end-policy-map
!
policy-map parent-3play-subscriber-line
 class class-default
  service-policy child-3play
  shape average 1 mbps
 !
 end policy-map
!
```

A policy is applied on an interface without ANCP:

```
interface GigabitEthernet 0/1/0/0.1 l2transport
encapsulation dot1q 2
 service-policy output parent-3play-subscriber-line
!
```

The **show qos** command verifies that ANCP has not been applied (ANCP is shown as 0 kbps).

```
RP/0/RSP0/CPU0:router# show qos interface GigabitEthernet 0/1/0/0.1 out

Interface: GigabitEthernet0_1_0_0.1 output Bandwidth: 1000000 kbps
ANCP: 0 kbps
Policy: parent-3-play-subscriber-line Total number of classes: 5
------------------------------------------------------------------------------
Level: 0 Policy: parent-3-play-subscriber-line Class: class-default
QueueID: N/A
Shape Profile: 1 CIR: 960 kbps CBS: 1024 bytes PIR: 960 kbps PBS: 13312 bytes
WFQ Profile: 1 Committed Weight: 1 Excess Weight: 1
Bandwidth: 0 kbps, BW sum for Level 0: 1000000 kbps, Excess Ratio: 1
------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-voip
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 8 (Priority 1)
Queue Limit: 16 kbytes Profile: 3 Scale Profile: 0
```

```
Policer Profile: 0 (Single)
Conform: 65 kbps (65 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-video
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 9 (Priority 2)
Queue Limit: 8 kbytes (11 Unknown) Profile: 4 Scale Profile: 0
Policer Profile: 24 (Single)
Conform: 128 kbps (128 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
WRED Type: COS based Table: 0 Profile: 4 Scale Profile: 0 Curves: 3
Default RED Curve Thresholds Min : 8 kbytes Max: 8 kbytes
WRED Curve: 1 Thresholds Min : 8 kbytes Max: 8kbytes
 Match: 3
WRED Curve: 2 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 4
------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3-play-premium
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 10 (Priority Normal)
Queue Limit: 16 kbytes Profile: 1 Scale Profile: 1
WFQ Profile: 4 Committed Weight: 100 Excess Weight: 100
Bandwidth: 1000 kbps, BW sum for Level 1: 1000 kbps, Excess Ratio: 1
------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: class-default
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 11 (Priority Normal)
Queue Limit: 8 kbytes Profile: 1 Scale Profile: 0
WFQ Profile: 5 Committed Weight: 1 Excess Weight: 1
Bandwidth: 0 kbps, BW sum for Level 1: 1000 kbps, Excess Ratio: 1
------------------------------------------------------------------------------
RP/0/RSP0/CPU0:router#
```

ANCP AN-Port to Interface Mapping is applied:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# ancp an-port circuit-id dslam1_port1 interface GigabitEthernet
 0/1/0/0.1
```

The **show ancp an-port interface** command shows the ANCP rate for the interface:

```
RP/0/RSP0/CPU0:router# show ancp an-port interface GigabitEthernet 0/1/0/0.1 detail

   AN port circuit-id dlsam1_port1:

   State                                                UP
   Uptime                                               00:00:32
   Time Since Last Message                              00:00:32
   Encap Type                                           ATM
   DSL Type                                             ADSL1
   DSL Line State                                       SHOWTIME
   Number of Mapped Sub-interfaces                      1
   Neighbor sender-name                                 0000.0000.1bec
   Neighbor description                                 -
   Configured Rate Adjustment                           100%
   Actual Downstream Data Rate (kbps)                   2000
   Effective Downstream Data Rate (kbps)                2000
   Actual Data Rate Upstream/Downstream (kbps)          2000/2000
```

```
   Minimum Data Rate Upstream/Downstream (kbps)           0/0
   Attainable Data Rate Upstream/Downstream (kbps)        0/0
   Maximum Data Rate Upstream/Downstream (kbps)           0/0
   Minimum Low Power Data Rate Upstream/Downstream (kbps) 0/0
   Maximum Interleaving Delay Upstream/Downstream (ms)    0/0
   Actual Interleaving Delay Upstream/Downstream (ms)     0/0

Sub-interface Summary: total 1
-----------------------------------------------------
Sub-interface name                    ifhandle
----------------------------------    ----------
GigabitEthernet0/1/0.1                0x215e042
```

The **show qos** command verifies that ANCP has been applied (ANCP is now shown as 1920 kbps).

```
RP/0/RSP0/CPU0/router# show qos interface GigabitEthernet 0/1/0.1 out

Interface GigabitEthernet0_1_0_0.1 output Bandwidth: 1000000 kbps
ANCP: 1920 kbps
Policy: parent-3play-subscriber-line Total number of classes: 5
------------------------------------------------------------------------
Level: 0 Policy: parent-3-play-subscriber-line Class: class-default
QueueID: N/A
Shape Profile: 1 CIR: 1920 kbps CBS: 1024 bytes PIR: 1920 kbps PBS: 13312 bytes
WFQ Profile: 1 Committed Weight: 1 Excess Weight: 1
Bandwidth: 0 kbps, BW sum for Level 0: 1000000 kbps, Excess Ratio: 1
------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-voip
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 8 (Priority 1)
Queue Limit: 16 kbytes Profile: 3 Scale Profile: 0
Policer Profile: 0 (Single)
Conform: 65 kbps (65 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-video
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 9 (Priority 2)
Queue Limit: 8 kbytes (11 Unknown) Profile: 4 Scale Profile: 0
Policer Profile: 24 (Single)
Conform: 128 kbps (128 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
WRED Type: COS based Table: 0 Profile: 4 Scale Profile: 0 Curves: 3
Default RED Curve Thresholds Min : 8 kbytes Max: 8 kbytes
WRED Curve: 1 Thresholds Min : 8 kbytes Max: 8kbytes
 Match: 3
WRED Curve: 2 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 4
------------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3-play-premium
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 10 (Priority Normal)
Queue Limit: 24 kbytes Profile: 1 Scale Profile: 8
WFQ Profile: 4 Committed Weight: 100 Excess Weight: 100
Bandwidth: 1920 kbps, BW sum for Level 1: 1920 kbps, Excess Ratio: 1
------------------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: class-default
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 11 (Priority Normal)
Queue Limit: 8 kbytes Profile: 1 Scale Profile: 0
```

```
WFQ Profile: 5 Committed Weight: 1 Excess Weight: 1
Bandwidth: 0 kbps, BW sum for Level 1: 1920 kbps, Excess Ratio: 1
------------------------------------------------------------------------------
```

QoS Policy Inconsistency on an Interface: Example

A valid QoS policy with absolute or percentage values must satisfy the following requirement:

interface speed > ANCP rate > QoS parent shaper rate

A Qos policy successfully applied to an interface can become invalid due to two possible external factors. These two factors are an ANCP rate change or a port speed change:

- ANCP Rate Change—If the ANCP rate falls, or the ANCP rate adjustment factor makes the ANCP rate fall below the shaper rate of the top-most QoS policy map, the QoS policy on the interface becomes invalid.
- Port Speed Change—The port of a GigabitEthernet interface can be configured to 10 Mbps or 100 Mbps mode from the default of 1000 Mbps. When this happens, the interface speed drops to less than the ANCP rate and QoS parent shaper rate. The QoS policy on the interface becomes invalid.

When either of these changes occur, the QoS policy on the interface is placed in the inconsistency state. To recover from the inconsistency state, perform one of the following tasks:

- Remove the QoS policy from the interface, adjust the QoS policy values, then reapply the QoS policy to the interface.
- If the ANCP adjustment rate or the ANCP rate has been modified, update the ANCP rate to satisfy the QoS policy rate requirement.
- If port speed has been modified, update the speed to satisfy the QoS policy rate requirement.

Following are examples of the effects of an ANCP rate change and a port speed change have on the following QoS policy configuration on a Gigabit Ethernet interface:

```
policy-map child-3play
 class 3play-voip
  priority level 1
  police rate 65 kbps
  !
 !
 class 3play-video
  priority level 2
  police rate 128 kbps
  !
   random-detect cos 3 10 ms 100 ms
   random-detect cos 4 20 ms 200 ms
 !
 class 3play-premium
  bandwidth percent 100
 !
 Class class-default
 !
 end-policy-map
!
policy-map parent-3play-subscriber-line
 class class-default
  service-policy child-3play
  bandwidth 200 mbps
```

```
  bandwidth remaining percent 100
  shape average 800 mbps
 !
 end-policy-map
!
```

If the ANCP rate value 999936 kbps, and the ANCP rate factor is 100 percent, the ANCP rate value of 999936 is applied to the interface. This satisfies the requirement:

Interface speed (1000000 kbps) > ANCP rate (999936 kbps) > QoS parent shaper rate (800000 kbps)

This is a successful application of the policy as shown by the following **show qos interface** command output:

```
show qos interface gig0/0/0/11.1 output

Wed Mar 18 18:25:20.140 UTC
Interface: GigabitEthernet0_0_0_11.1 output Bandwidth: 1000000 kbps ANCP: 999936 kbps
Policy: parent-3play-subscriber-line Total number of classes: 5
------------------------------------------------------------------------
Level: 0 Policy: parent-3play-subscriber-line Class: class-default
QueueID: N/A
Shape Profile: 1 CIR: 200000 kbps (200 mbps)
CBS: 100352 bytes PIR: 999936 kbps PBS: 12517376 bytes
WFQ Profile: 1 Committed Weight: 51 Excess Weight: 100
Bandwidth: 200000 kbps, BW sum for Level 0: 1000000 kbps, Excess Ratio: 100
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-voip
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 136 (Priority 1)
Queue Limit: 16 kbytes Profile: 3 Scale Profile: 0
Policer Profile: 0 (Single)
Conform: 65 kbps (65 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-video
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 137 (Priority 2)
Queue Limit: 8 kbytes (11 Unknown) Profile: 4 Scale Profile: 0
Policer Profile: 24 (Single)
Conform: 128 kbps (128 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
WRED Type: COS based Table: 0 Profile: 4 Scale Profile: 0 Curves: 3
Default RED Curve Thresholds Min : 8 kbytes Max: 8 kbytes
WRED Curve: 1 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 3
WRED Curve: 2 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 4
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-premium
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 138 (Priority Normal)
Queue Limit: 2097 kbytes Profile: 2 Scale Profile: 0
WFQ Profile: 6 Committed Weight: 1020 Excess Weight: 1020
Bandwidth: 200000 kbps, BW sum for Level 1: 200000 kbps, Excess Ratio: 1
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: class-default
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 139 (Priority Normal)
Queue Limit: 65 kbytes Profile: 1 Scale Profile: 3
WFQ Profile: 0 Committed Weight: 1 Excess Weight: 1020
```

```
Bandwidth: 0 kbps, BW sum for Level 1: 200000 kbps, Excess Ratio: 1
------------------------------------------------------------------------
```

ANCP Rate Change

If the ANCP rate falls below the QoS parent shaper rate for example, to 300000 kbps, and the ANCP rate adjustment factor remains at 100 percent, the ANCP rate is no longer greater than the QoS parent shaper rate of 800000 kbps. This causes the QoS policy on the interface to be placed in the inconsistency state as shown by the following **show qos interface** command output:

```
show qos interface gig0/0/0/11.1 output

Wed Mar 18 18:21:11.180 UTC
Interface: GigabitEthernet0_0_0_11.1 output Bandwidth: 1000000 kbps ANCP: 299904 kbps
 *Inconsistency* :  ANCP - Downstream Rate less than Shaper Rate
Policy: parent-3play-subscriber-line Total number of classes: 5
------------------------------------------------------------------------
Level: 0 Policy: parent-3play-subscriber-line Class: class-default
QueueID: N/A
Shape Profile: 2 CIR: 200000 kbps (200 mbps)
CBS: 100352 bytes PIR: 800000 kbps PBS: 10027008 bytes
WFQ Profile: 1 Committed Weight: 51 Excess Weight: 100
Bandwidth: 200000 kbps, BW sum for Level 0: 1000000 kbps, Excess Ratio: 100
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-voip
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 136 (Priority 1)
Queue Limit: 16 kbytes Profile: 3 Scale Profile: 0
Policer Profile: 0 (Single)
Conform: 65 kbps (65 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-video
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 137 (Priority 2)
Queue Limit: 8 kbytes (11 Unknown) Profile: 4 Scale Profile: 0
Policer Profile: 24 (Single)
Conform: 128 kbps (128 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
WRED Type: COS based Table: 0 Profile: 4 Scale Profile: 0 Curves: 3
Default RED Curve Thresholds Min : 8 kbytes Max: 8 kbytes
WRED Curve: 1 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 3
WRED Curve: 2 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 4
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-premium
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 138 (Priority Normal)
Queue Limit: 2097 kbytes Profile: 2 Scale Profile: 0
WFQ Profile: 6 Committed Weight: 1020 Excess Weight: 1020
Bandwidth: 200000 kbps, BW sum for Level 1: 200000 kbps, Excess Ratio: 1
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: class-default
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 139 (Priority Normal)
Queue Limit: 65 kbytes Profile: 1 Scale Profile: 3
WFQ Profile: 0 Committed Weight: 1 Excess Weight: 1020
```

```
Bandwidth: 0 kbps, BW sum for Level 1: 200000 kbps, Excess Ratio: 1
------------------------------------------------------------------------
```

Once the ANCP rate returns to the configured value, the inconsistency is automatically cleared, which can be confirmed by issuing the **show qos interface** command.

Note If the ANCP rate has been configured to a value less than the shape rate, the inconsistency is not automatically cleared, and the policy must be modified and reapplied. To prevent this from occurring, be sure to configure the policy-map shape rate to the minimum value of all ANCP rates for a given service level.

Port Speed Change

If the port speed is configured to less than the QoS parent shaper rate for example to 100 Mbps (100000 kbps), the requirement is no longer met since the port speed is no longer greater than the QoS parent shaper rate of 800000 kbps.

```
RP/0/RSP0/CPU0:ro-node1#conf
RP/0/RSP0/CPU0:ro-node1(config)#int gigabitEthernet 0/0/0/1
RP/0/RSP0/CPU0:ro-node1(config-if)#speed 100
RP/0/RSP0/CPU0:ro-node1(config-if)#commit
LC/0/0/CPU0:Nov  4 05:36:55.041 : qos_ma_ea[197]: %QOS-QOS_EA_MODIFY_FAIL-3-ERROR :
inconsistency detected due to ANCP or Bandwidth modification. Execute show qos inconsistency,
 to obtain information. Policy resolution failure
RP/0/RSP0/CPU0:ro-node1(config-if)#end
```

This causes the QoS policy on the interface to be placed in the inconsistency state as shown by the following **show qos interface** command output:

```
RP/0/RSP0/CPU0:ro-node1#sh qos int gigabitEthernet 0/0/0/1.1 output
Interface: GigabitEthernet0_0_0_1.1 output Bandwidth: 1000000 kbps ANCP: 0 kbps
 *Inconsistency* :  Port speed modify fails on Policy
Policy: parent-3play-subscriber-line Total number of classes: 5
------------------------------------------------------------------------
Level: 0 Policy: parent-3play-subscriber-line Class: class-default
QueueID: N/A
Shape Profile: 1 CIR: 200000 kbps (200 mbps)
CBS: 100352 bytes PIR: 800000 kbps PBS: 10027008 bytes
WFQ Profile: 1 Committed Weight: 51 Excess Weight: 100
Bandwidth: 200000 kbps, BW sum for Level 0: 1000000 kbps, Excess Ratio: 100
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-voip
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 640 (Priority 1)
Queue Limit: 16 kbytes Profile: 3 Scale Profile: 0
Policer Profile: 0 (Single)
Conform: 65 kbps (65 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-video
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 641 (Priority 2)
Queue Limit: 8 kbytes Profile: 4 Scale Profile: 0
Policer Profile: 24 (Single)
Conform: 128 kbps (128 kbps) Burst: 1598 bytes (0 Default)
Child Policer Conform: TX
```

```
Child Policer Exceed: DROP
Child Policer Violate: DROP
WRED Type: COS based Table: 2 Profile: 4 Scale Profile: 0 Curves: 3
Default RED Curve Thresholds Min : 8 kbytes Max: 8 kbytes
WRED Curve: 1 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 3
WRED Curve: 2 Thresholds Min : 8 kbytes Max: 8 kbytes
 Match: 4
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: 3play-premium
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 642 (Priority Normal)
Queue Limit: 4194 kbytes Profile: 2 Scale Profile: 1
WFQ Profile: 3 Committed Weight: 1020 Excess Weight: 1020
Bandwidth: 200000 kbps, BW sum for Level 1: 200000 kbps, Excess Ratio: 1
------------------------------------------------------------------------
Level: 1 Policy: child-3play Class: class-default
Parent Policy: parent-3play-subscriber-line Class: class-default
QueueID: 643 (Priority Normal)
Queue Limit: 4194 kbytes Profile: 2 Scale Profile: 1
WFQ Profile: 4 Committed Weight: 1 Excess Weight: 1
Bandwidth: 0 kbps, BW sum for Level 1: 200000 kbps, Excess Ratio: 1
------------------------------------------------------------------------
```

To resolve this issue, the port speed must be set back to 1000 Mbps (1000000 kbps) using the **no speed** command.

```
RP/0/RSP0/CPU0:ro-node1#conf
RP/0/RSP0/CPU0:ro-node1(config)#int gigabitEthernet 0/0/0/1
RP/0/RSP0/CPU0:ro-node1(config-if)#no speed
RP/0/RSP0/CPU0:ro-node1(config-if)#commit
LC/0/0/CPU0:Nov  4 05:37:39.171 : ifmgr[144]: %PKT_INFRA-LINEPROTO-5-UPDOWN : Line protocol
 on Interface GigabitEthernet0/0/0/1, changed state to Up
```

The clearing of the inconsistency can be verified by again issuing the **show qos interface** command.

The show qos inconsistency Command: Example

A command related to **show qos interface** command provides additional detail about QoS policy inconsistency:

```
RP/0/RSP0/CPU0:RO2#show qos inconsistency detail 0 location 0/7/CPU0

Interface Lists with QoS Inconsistency Warning:
=======================================================

                     Node 0/7/CPU0
-------------------------------------------------------

Interfaces with QoS Inconsistency:  ANCP - No Shaper at top policymap
========================================================================
Interface                  Direction  Policy Name             SPI Name
------------------------------------------------------------------------
GigabitEthernet0/7/0/1.5     output    parent-none

Interfaces with QoS Inconsistency:  ANCP - Downstream Rate less than Shaper Rate
========================================================================
Interface                  Direction  Policy Name             SPI Name
------------------------------------------------------------------------
GigabitEthernet0/7/0/1       output    parent                  SPI1
GigabitEthernet0/7/0/1.2     output    parent
GigabitEthernet0/7/0/1       output    normal-policy-name      normal-spi-name
```

```
RP/0/RSP0/CPU0:RO2#
RP/0/RSP0/CPU0:RO2#show qos inconsistency summary location 0/7/CPU0

Summary Counts of QoS Inconsistency Warnings:
============================================================


                     Node 0/7/CPU0

   Inconsistency Warning Type                 Count
------------------------------------------------------------
ANCP - No Shaper at top policymap:               1
ANCP - Downstream Rate less than Shaper Rate:    4
RP/0/RSP0/CPU0:RO2#
```

Additional References

The following sections provide references related to implementing ANCP.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Gett Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Mod of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Ro of Cisco Cisco ASR 9000 Series Aggregation Services Re Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html

Configuring Access Node Control Protocol

Access Node Control Protocol (ANCP) creates a control plane between a service-oriented aggregation device and an access node (AN) (for example, a DSLAM) in order to perform QoS-related, service-related, and subscriber-related operations. An ANCP server accepts and maintains ANCP adjacencies (sessions with an ANCP neighbor), and sending and receiving ANCP messages. ANCP allows static mapping between ANCP ports and VLAN subinterfaces so that DSL rate updates for a specific subscriber received by the ANCP server are applied to the QoS configuration corresponding to that subscriber. DSL train rates received via ANCP are used to alter shaping rates on subscriber-facing interfaces and subinterfaces on the router. ANCP runs as a single process on the route processor (RP).

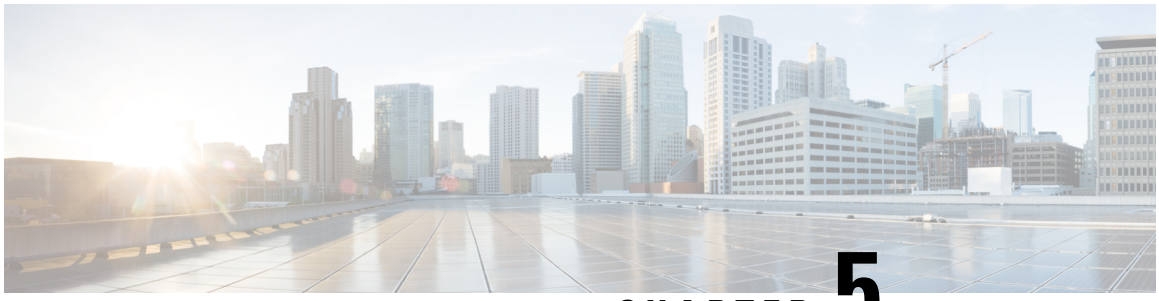
This module provides the conceptual and configuration information for implementing ANCP.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Access Node Control Protocol	yes	no

Feature History for Configuring Access Node Protocol on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.2	The Access Node Control Protocol feature was introduced.
Release 3.9.0	Mapping of ANCP ports to VLAN interfaces over Ethernet bundles was added.
Release 4.0.0	ANCP over Multi Chassis Link Aggregation was introduced.



CHAPTER 5

Configuring Modular QoS Congestion Avoidance

Congestion avoidance techniques monitor traffic flow in an effort to anticipate and avoid congestion at common network bottlenecks. Avoidance techniques are implemented before congestion occurs as compared with congestion management techniques that control congestion after it has occurred.

Congestion avoidance is achieved through packet dropping. Cisco IOS XR software supports these quality of service (QoS) congestion avoidance techniques that drop packets:

- Random early detection (RED)
- Weighted random early detection (WRED)
- Tail drop

The module describes the concepts and tasks related to these congestion avoidance techniques.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Random Early Detection	yes	yes
Weighted Random Early Detection	yes	yes
Tail Drop	yes	yes

Feature History for Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.2	The Congestion Avoidance feature was introduced on ASR 9000 Ethernet Line Cards. The Random Early Detection, Weighted Random Early Detection, and Tail Drop features were introduced on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The Random Early Detection, Weighted Random Early Detection, and Tail Drop features were supported on the SIP 700 for the ASR 9000.

• Additional References, on page 52

Prerequisites for Configuring Modular QoS Congestion Avoidance

This prerequisite is required for configuring QoS congestion avoidance on your network:

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Configuring Modular QoS Congestion Avoidance

Random Early Detection and TCP

The Random Early Detection (RED) congestion avoidance technique takes advantage of the congestion control mechanism of TCP. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to decrease its transmission rate. Assuming the packet source is using TCP, it decreases its transmission rate until all packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow transmission of packets. TCP not only pauses, but it also restarts quickly and adapts its transmission rate to the rate that the network can support.

RED distributes losses in time and maintains normally low queue depth while absorbing traffic bursts. When enabled on an interface, RED begins dropping packets when congestion occurs at a rate you select during configuration.

Queue-limit for WRED

Queue-limit is used to fine-tune the number of buffers available for each queue. It can only be used on a queuing class. Default queue limit is 100 ms of the service rate for the given queue. The service rate is the sum of minimum guaranteed bandwidth and bandwidth remaining assigned to a given class either implicitly or explicitly.

The queue-limit is rounded up to the nearest power of 2, and depending on the line cards on your system, the queue-limit values vary. To check the current queue-limit for class-default, use the **show qos interface** command.

Because WRED needs a queue to operate on, the class that WRED is applied on must have either a bandwidth statement or a parent policy with a shaper if WRED is applied only on a class default queue.

Examples

The following policy configuration does not use the queue limit because the policy is flat and doesn't have a designated queue on which it operates.

```
policy-map incorrect-flat
class class-default
  random-detect dscp 16 250 packets 500 packets
```

```
  queue-limit 158000 kbytes
```

The following policy configuration can use the queue limit because it uses a parent policy map with the shape average command.

```
policy-map parent
class class-default
shape average 100 mbps
service-policy child

policy-map child
class class-default
  random-detect dscp 16 250 packets 500 packets
  queue-limit 158000 kbytes
```

The following policy configuration can use the queue limit because it provides a flat policy with a shaped queue through the bandwidth command for the class-default.

```
policy-map correct-flat
class class-default
  bandwidth 100 mbps
  random-detect dscp 16 250 packets 500 packets
  queue-limit 158000 kbytes
```

Tail Drop and the FIFO Queue

Tail drop is a congestion avoidance technique that drops packets when an output queue is full until congestion is eliminated. Tail drop treats all traffic flow equally and does not differentiate between classes of service. It manages the packets that are unclassified, placed into a first-in, first-out (FIFO) queue, and forwarded at a rate determined by the available underlying link bandwidth.

See the “Default Traffic Class” section of the “Configuring Modular Quality of Service Packet Classification and Marking on Cisco ASR 9000 Series Routers”.

Configuring Random Early Detection

This configuration task is similar to that used for WRED except that the **random-detect precedence** command is not configured and the **random-detect** command with the **default** keyword must be used to enable RED.

Restrictions

If you configure the **random-detect default** command on any class including class-default, you must configure one of the following commands:

- **shape average**
- **bandwidth**
- **bandwidth remaining**

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-map-name*
3. **class** *class-name*

4. **random-detect** {**cos** *value* | **default** | **discard-class** *value* | **dscp** *value* | **exp** *value* | **precedence** *value* | *min-threshold* [*units*] *max-threshold* [*units*] }
5. **bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value*
6. **shape average** {**percent** *percentage* | *value* [*units*]}
7. **exit**
8. **exit**
9. **interface** *type interface-path-id*
10. **service-policy** {**input** | **output**} *policy-map*
11. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**random-detect** {**cos** *value* | **default** | **discard-class** *value* | **dscp** *value* | **exp** *value* | **precedence** *value* | *min-threshold* [*units*] *max-threshold* [*units*] } **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# random-detect default`	Enables RED with default minimum and maximum thresholds.
Step 5	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30` or	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. or (Optional) Specifies how to allocate leftover bandwidth to various classes.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	
Step 6	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`	(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 9	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/2/0/0`	Enters the configuration mode and configures an interface.
Step 10	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Configuring Weighted Random Early Detection

WRED drops packets selectively based on any specified criteria, such as CoS, DSCP, EXP, discard-class, or precedence . WRED uses these matching criteria to determine how to treat different types of traffic.

Configure WRED using the **random-detect** command and different CoS, DSCP, EXP, and discard-class values. The value can be range or a list of values that are valid for that field. You can also use minimum and maximum queue thresholds to determine the dropping point.

When a packet arrives, the following actions occur:

- If the queue size is less than the minimum queue threshold, the arriving packet is queued.
- If the queue size is between the minimum queue threshold for that type of traffic and the maximum threshold for the interface, the packet is either dropped or queued, depending on the packet drop probability for that type of traffic.
- If the queue size is greater than the maximum threshold, the packet is dropped.

Restrictions

- On systems with Cisco ASR 9000 High-Density 100GE Ethernet line cards and fifth-generation line cards, ensure that you configure the minimum and maximum threshold values that are greater than the default minimum and maximum threshold values. If you apply a policy that has lesser than default values to a bundle that has both these line cards, the **show policy-map interface** command displays a mismatch in statistics bag size.
- When configuring the **random-detect dscp** command, you must configure one of the following commands: **shape average**, **bandwidth**, and **bandwidth remaining**.

Note The Cisco ASR 9000 Series ATM SPA supports only time-based WRED thresholds. Therefore, if you try to configure the WRED threshold using the **random-detect default** command with bytes or packet as the threshold units, the "Unsupported WRED unit on ATM interface" error occurs.

- Only two minimum and maximum thresholds (each with different match criteria) can be configured per class.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **random-detect dscp** *dscp-value min-threshold* [*units*] *max-threshold* [*units*]
5. **bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value]*
6. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
7. **bandwidth remaining percent** *value*
8. **shape average** {**percent** *percentage* | *value* [*units*]}
9. **queue-limit** *value* [*units*]
10. **exit**
11. **interface** *type interface-path-id*
12. **service-policy** {**input** | **output**} *policy-map*
13. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**random-detect dscp** *dscp-value min-threshold* [*units*] *max-threshold* [*units*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# random-detect dscp af11 1000000 bytes 2000000 bytes`	Modifies the minimum and maximum packet thresholds for the DSCP value. • Enables WRED. • *dscp-value*—Number from 0 to 63 that sets the DSCP value. Reserved keywords can be specified instead of numeric values. • *min-threshold*—Minimum threshold in the specified units. When the average queue length reaches the minimum threshold, WRED randomly drops some packets with the specified DSCP value. • *max-threshold*—Maximum threshold in the specified units. When the average queue length exceeds the maximum threshold, WRED drops all packets with the specified DSCP value. • *units*—Units of the threshold value. This can be **bytes**, **gbytes**, **kbytes**, **mbytes**, **ms** (milliseconds), **packets**, or **us** (microseconds). The default is **packets**. • This example shows that for packets with DSCP AF11, the WRED minimum threshold is 1,000,000 bytes and maximum threshold is 2,000,000 bytes.
Step 5	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} or **bandwidth remaining** [**percent** *value* | **ratio** *ratio-value]* **Example:**	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. or

	Command or Action	Purpose

	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`
		(Optional) Specifies how to allocate leftover bandwidth to various classes.

Step 6	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class1.
Step 7	**bandwidth remaining percent** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	(Optional) Specifies how to allocate leftover bandwidth to various classes. • The remaining bandwidth of 70 percent is shared by all configured classes. • In this example, class class1 receives 20 percent of the 70 percent.
Step 8	**shape average** {**percent** *percentage* | *value* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`	(Optional) Shapes traffic to the specified bit rate or a percentage of the available bandwidth.
Step 9	**queue-limit** *value* [*units*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# queue-limit 50 ms`	(Optional) Changes queue-limit to fine-tune the amount of buffers available for each queue. The default queue-limit is 100 ms of the service rate for a non-priority class and 10ms of the service rate for a priority class. **Note** Even though this command is optional, it is recommended that you use it to fine-tune the queue limit, instead of relying on your system default settings. If the queue limit is too large, the buffer consumption goes up, resulting in delays. On the other hand, too small a queue limit may result in extra drops while allowing for faster rate adaption.
Step 10	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 11	**interface** *type interface-path-id* **Example:**	Enters the configuration mode and configures an interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/2/0/0`	
Step 12	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • In this example, the traffic policy evaluates all traffic leaving that interface. • Ingress policies are not valid; the **bandwidth** and **bandwidth remaining** commands cannot be applied to ingress policies.
Step 13	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Tail Drop

Packets satisfying the match criteria for a class accumulate in the queue reserved for the class until they are serviced. The **queue-limit** command is used to define the maximum threshold for a class. When the maximum threshold is reached, enqueued packets to the class queue result in tail drop (packet drop).

The **queue-limit** value uses the guaranteed service rate (GSR) of the queue as the reference value for the **queue_bandwidth**. If the class has bandwidth percent associated with it, the **queue-limit** is set to a proportion of the bandwidth reserved for that class.

If the GSR for a queue is zero, use the following to compute the default **queue-limit**:

- 1 percent of the interface bandwidth for queues in a nonhierarchical policy.
- 1 percent of parent maximum reference rate for hierarchical policy.

  The parent maximum reference rate is the minimum of parent shape, policer maximum rate, and the interface bandwidth.

Note The default **queue-limit** is set to bytes of 100 ms of queue bandwidth. The following formula is used to calculate the default queue limit (in bytes):??bytes = (100 ms / 1000 ms) * queue_bandwidth kbps)) / 8

Restrictions

- When configuring the **queue-limit** command in a class, you must configure one of the following commands: **priority**, **shape average**, **bandwidth**, or **bandwidth remaining**, except for the default class.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **queue-limit** *value* [*units*]
5. **priority** [**level** *priority-level*]
6. **police rate percent** *percentage*
7. **class** *class-name*
8. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
9. **bandwidth remaining percent** *value*
10. **exit**
11. **exit**
12. **interface** *type interface-path-id*
13. **service-policy** {**input** | **output**} *policy-map*
14. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and also enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**queue-limit** *value* [*units*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# queue-limit 1000000 bytes`	Specifies or modifies the maximum the queue can hold for a class policy configured in a policy map. The default value of the *units* argument is **packets**. In this example, when the queue limit reaches 1,000,000 bytes, enqueued packets to the class queue are dropped.

	Command or Action	Purpose
Step 5	**priority** [**level** *priority-level*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# priority level 1`	Specifies priority to a class of traffic belonging to a policy map.
Step 6	**police rate percent** *percentage* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate percent 30`	Configures traffic policing.
Step 7	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change. In this example, class2 is configured.
Step 8	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`	(Optional) Specifies the bandwidth allocated for a class belonging to a policy map. This example guarantees 30 percent of the interface bandwidth to class class2.
Step 9	**bandwidth remaining percent** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	(Optional) Specifies how to allocate leftover bandwidth to various classes. This example allocates 20 percent of the leftover interface bandwidth to class class2.
Step 10	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface POS 0/2/0/0`	Enters the configuration mode and configures an interface.
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:**	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Additional References

These sections provide references related to implementing QoS congestion avoidance.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Getting Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Modular of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Route of Cisco Cisco ASR 9000 Series Aggregation Services Route Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

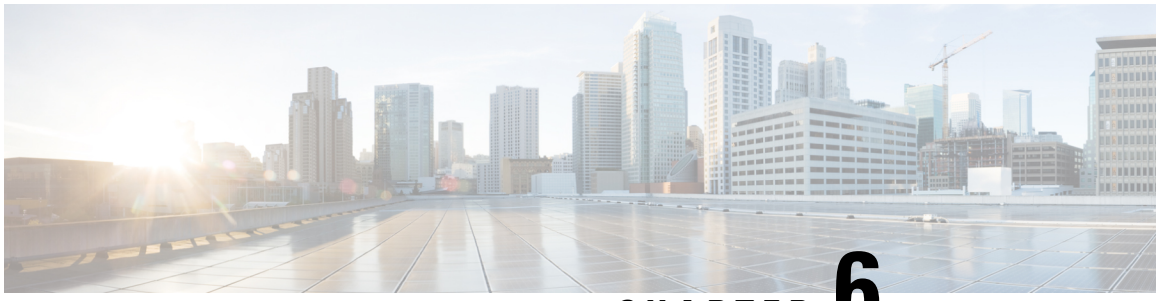
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 6

Configuring Modular QoS Congestion Management

Congestion management controls congestion after it has occurred on a network. Congestion is managed on Cisco IOS XR software by using packet queuing methods and by shaping the packet flow through use of traffic regulation mechanisms.

The types of traffic regulation mechanisms supported are:

- Traffic shaping:
  - Modified Deficit Round Robin (MDRR)
  - Low-latency queuing (LLQ) with strict priority queuing (PQ)
- Traffic policing:
  - Color blind
  - Color-aware (ingress direction)

Line Card, SIP, and SPA Support

This table lists the features that are supported on the ASR 9000 Ethernet Line Cards and SIP 700 for the ASR 9000.

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Congestion Management Using DEI	yes	yes
Guaranteed and Remaining Bandwidth	yes	yes
Low-Latency Queueing with Strict Priority Queueing	yes	yes
Traffic Policing	yes	yes
Traffic Shaping	yes	yes

Note Ingress queueing is not supported on the A9K-24X10GE-1G-SE line card.

Feature History for Configuring Modular QoS Congestion Management on Cisco ASR 9000 Series Router

Release	Modification
Release 3.7.2	The Congestion Avoidance feature was introduced on ASR 9000 Ethernet Line Cards.. The Guaranteed and Remaining Bandwidth, Low-Latency Queueing with Strict Priority Queueing, Traffic Policing, and Traffic Shaping features were introduced on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The Guaranteed and Remaining Bandwidth, Low-Latency Queueing with Strict Priority Queueing, Traffic Policing, and Traffic Shaping features were supported on the SIP 700 for the ASR 9000.
Release 4.0.0	The Congestion Management Using DEI feature was introduced on ASR 9000 Ethernet Line Cards.
Release 4.0.1	The **police rate** command was updated to include packet-based specifications of policing rates and burst sizes.
Release 4.1.0	The 2-rate 3-color policer feature was added, including the **conform-color** and **exceed-color** commands. This feature is applicable to the SIP 700 line cards, ingress side.
Release 4.2.1	The Configured Accounting and QoS for IPv6ACLs features were added.
Release 5.3.2	The existing egress priority levels are enhanced from P1, P2 and P3 to P1, P2, P3, P4, P5, P6 and P7. For all the releases prior to 5.3.2, system supports Priority levels P1, P2 and P3 only.
Release 6.0.1	Traffic Policing on Layer 2 ATM Interfaces

Prerequisites for Configuring QoS Congestion Management

These prerequisites are required for configuring QoS congestion management on your network:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- You must be familiar with Cisco IOS XR QoS configuration tasks and concepts.

Information About Configuring Congestion Management

Congestion Management Overview

Congestion management features allow you to control congestion by determining the order in which a traffic flow (or packets) is sent out an interface based on priorities assigned to packets. Congestion management entails the creation of queues, assignment of packets to those queues based on the classification of the packet, and scheduling of the packets in a queue for transmission. The congestion management features in Cisco IOS XR software allow you to specify creation of a different number of queues, affording greater or lesser degree of differentiation of traffic, and to specify the order in which that traffic is sent.

During periods with light traffic flow, that is, when no congestion exists, packets are sent out the interface as soon as they arrive. During periods of transmit congestion at the outgoing interface, packets arrive faster than the interface can send them. If you use congestion management features, packets accumulating at an interface are queued until the interface is free to send them; they are then scheduled for transmission according to their assigned priority and the queuing method configured for the interface. The router determines the order of packet transmission by controlling which packets are placed in which queue and how queues are serviced with respect to each other.

In addition to queuing methods, QoS congestion management mechanisms, such as policers and shapers, are needed to ensure that a packet adheres to a contract and service. Both policing and shaping mechanisms use the traffic descriptor for a packet.

Policers and shapers usually identify traffic descriptor violations in an identical manner through the token bucket mechanism, but they differ in the way they respond to violations. A policer typically drops traffic flow; whereas, a shaper delays excess traffic flow using a buffer, or queuing mechanism, to hold the traffic for transmission at a later time.

Traffic shaping and policing can work in tandem. For example, a good traffic shaping scheme should make it easy for nodes inside the network to detect abnormal flows.

Modified Deficit Round Robin

MDRR is a class-based composite scheduling mechanism that allows for queueing of up to eight traffic classes. It operates in the same manner as class-based weighted fair queueing (CBWFQ) and allows definition of traffic classes based on customer match criteria (such as access lists); however, MDRR does not use the weighted fair queuing algorithm.

When MDRR is configured in the queuing strategy, nonempty queues are served one after the other. Each time a queue is served, a fixed amount of data is dequeued. The algorithm then services the next queue. When a queue is served, MDDR keeps track of the number of bytes of data that were dequeued in excess of the configured value. In the next pass, when the queue is served again, less data is dequeued to compensate for the excess data that was served previously. As a result, the average amount of data dequeued per queue is close to the configured value. In addition, MDRR allows for a strict priority queue for delay-sensitive traffic.

Each queue within MDRR is defined by two variables:

- Quantum value—Average number of bytes served in each round.

- Deficit counter—Number of bytes a queue has sent in each round. The counter is initialized to the quantum value.

Packets in a queue are served as long as the deficit counter is greater than zero. Each packet served decreases the deficit counter by a value equal to its length in bytes. A queue can no longer be served after the deficit counter becomes zero or negative. In each new round, the deficit counter for each nonempty queue is incremented by its quantum value.

Low-Latency Queueing with Strict Priority Queueing

The LLQ feature brings strict priority queuing (PQ) to the MDRR scheduling mechanism. PQ in strict priority mode ensures that one type of traffic is sent, possibly at the expense of all others. For PQ, a low-priority queue can be detrimentally affected, and, in the worst case, never allowed to send its packets if a limited amount of bandwidth is available or the transmission rate of critical traffic is high.

Strict PQ allows delay-sensitive data, such as voice, to be dequeued and sent before packets in other queues are dequeued.

LLQ enables the use of a single, strict priority queue within MDRR at the class level, allowing you to direct traffic belonging to a class. To rank class traffic to the strict priority queue, you specify the named class within a policy map and then configure the **priority** command for the class. (Classes to which the **priority** command is applied are considered priority classes.) Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is enqueued to the same, single, strict priority queue.

Through use of the **priority** command, you can assign a strict PQ to any of the valid match criteria used to specify traffic. These methods of specifying traffic for a class include matching on access lists, protocols, IP precedence, and IP differentiated service code point (DSCP) values. Moreover, within an access list you can specify that traffic matches are allowed based on the DSCP value that is set using the first six bits of the IP type of service (ToS) byte in the IP header.

Overhead Accounting

Traffic shapers and policers use packet traffic descriptors to ensure adherence to the service level agreement in QoS. However, when traffic flows from one hop to another in a network, headers added or removed at interim hops affect the packet bytes being accounted for by QoS at each hop. When your end-user network measures the packet bytes to ensure they receive the payload as agreed, these additional header bytes cause a discrepancy.

QoS overhead accounting provides the flexibility to operators to decide which header bytes can be excluded by the traffic shaper and policer and which can be included, depending on the end user's requirements and device capabilities, to meet the committed payload in units of bytes.

For example, if the QoS commitment includes the additional header bytes, the overhead accounting feature allows your router to account for this overhead and reduces the traffic policing and shaping rates accordingly. This is also called a **positive accounting overhead**.

If however, the committed rate doesn't include the additional bytes, overhead accounting allows your router to adjust the core stream traffic such that the traffic policing and shaping rates are increased. This is also called a **negative accounting overhead**.

To summarize, QoS overhead accounting enables the router to account for packet overhead when shaping and policing traffic to a specific rate. This accounting ensures that the router runs QoS features on the actual bandwidth that the subscriber traffic consumes.

Any interface that supports QoS policies supports overhead accounting.

Note You can enable user overhead accounting using the optional configuration of **accounting user-defined <*overhead size in bytes*>** while attaching the service policy on the egress interface.

Prerequisites and Restrictions

- Overhead accounting for ingress shaping is not supported.
- Overhead accounting is not reflected in any QoS counters (classification, policing, or queuing). In other words, when you run show policy-map statistics, the results do not include the overhead bytes but display the actual packet sizes.
- Dynamic changing of accounting overhead after application of the policy on the interface is not supported.
- You must remove the service policy from the interface and apply it back with the required overhead value. You can, however, remove and reapply the service policy in a single configuration commit.

Configuring for Overhead Accounting

To configure overhead accounting, you must:

1. Create a policy map and configure QoS actions for that map.
2. Configure overhead accounting and attach the map to an interface.

```
/* create QoS policy */
Router#configure terminal
Router(config)#policy-map policer
Router(config-pmap)#class class-default
Router(config-pmap-c)#police rate percent 10
Router(config-pmap-c-police)#commit

/* configure account overhead value while attaching the QoS policy to interface */
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#service-policy input policer account user-defined 12
Router(config-if)#commit
Router(config-if)#root
Router(config)#end
```

Running Configuration

```
Router#sh run int hundredGigE 0/0/0/2
interface HundredGigE0/0/0/2
service-policy input policer account user-defined 12
!
```

Verification

```
Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
Total number of classes:   1
Total number of UBRL classes:  0
Total number of CAC classes:   0
------------------------------------------------------
Policy name: policer
Hierarchical depth 1
Interface type HundredGigE
Interface rate 100000000 kbps
```

```
Port Shaper rate 0 kbps
Interface handle 0x00000680
ul_ifh 0x00000000, ul_id  0x00000080
uidb index 0x0003
qos_ifh 0x10200020800003
Local port 16, NP 0
Policy map id 0x6018, format 8, uidb index 0x0003
------------------------------------------------------
Index 0 Level 0 Class name class-default service_id 0x0 Policy name policer
Node flags: LEAF DEFAULT DEFAULT-ALL
 Stats flags: Policer type 1 Max category 0
 Node Config:
 Police Color aware 0 Type 1 CIR/CBS/PIR/PBS: 10000000kbps/125000000B/0kbps/0B
 Node Result: Class-based stats:Stat ID 0x00000310
 Queue: N/A Stat ID(Commit): 0x00000000
 Stat ID(Drop Curve 0): 0x00000000
 Stat ID(Drop Curve 1): 0x00000000
 Stat ID(Drop Curve 2): 0x00000000
 Stat ID(Drop Curve 3): 0x00000000
 Police ID
(Token/Conform/Exceed/Violate/Parent-Drop):0x00000008/0x00000310/0x00000311/0x00000312/0x00000313


Hash key (host format):
0x55b44c6b93dd: 02 00 00 00 00 14 00 14
UIDB Index: 0x0000 (0)
Class ID  : 0x0000 (0)
SW SOURCE-PORT: 0x14 (20)

HW NP-PORT: 0x14 (20)

hash result (host format):
0x55b44c6b939b: 02 00 33 00 00 00 00 00
0x55b44c6b93a3: 00 00 00 00 08 00 00 00
0x55b44c6b93ab: 00 00 00 00 00 00 00 00
0x55b44c6b93b3: 10 03 00 00 00 00 00 00
0x55b44c6b93bb: 00 00 00 00 00 00 00 00
0x55b44c6b93c3: 00 00 00 00 00 00 00 00
0x55b44c6b93cb: 00 00 00 00 00 00 00 00
0x55b44c6b93d3: 00 00 00 00 00 00 00 00
0x55b44c6b93db: 0000
FID0                : 0x00000000 (0)
FID1                : 0x00000000 (0)
Token Bucket        : 0x000008 (8)
Parent Token Bucket : 0x000000 (0)
Stats ID            : 0x000310 (784)
Parent Stats ID     : 0x000000 (0)
COS                 : 0x000003 (3)
L1 Overhead Size    : 0x00000C (12)
L1 Overhead Negative: 0x000000 (0)


hash result (ucode format):
0300000000001400400000000000000800310a88000000c0000000000000000000000000000000000000000000000000000000
hash_tbl_id                       =           0 (0x0)
uidb_index                        =           3 (0x3)
class_id                          =           0 (0x0)
np_port                           =           0 (0x0)
sat_icl_bundle_port               =           0 (0x0)
ctrl_egress                       =           0 (0x0)
ctrl_ingress                      =           0 (0x0)
RESULT:
  q_action_valid                  =           1 (0x1)
  turbo                           =           0 (0x0)
```

```
  l1_oh_neg                       =           0 (0x0)
  match                           =           1 (0x1)
  queue_stats_offset              =           4 (0x4)
  c_stat                          =     8389392 (0x800310)
  p_stat                          =     8388608 (0x800000)
  val                             =          12 (0xc)
  c_tb                            =           0 (0x0)
  p_tb                            =           0 (0x0)
  gp_tb                           =           0 (0x0)
  queue_en                        =           0 (0x0)
  profile_table_ptr               =           0 (0x0)
queue_id 0
  offset_table_ptr                =           0 (0x0)
  offset_array_0                  =           0 (0x0)
                                  =  0 0 0 0 0 0 0 0
  offset_array_1                  =           0 (0x0)
                                  =  0 0 0 0 0 0 0 0

tcam result (host format):
0x55b44c6b9374: 80 41 00 03 00 00 00 00
0x55b44c6b937c: 4A FF 00 00 FF 00 00 FF
0x55b44c6b9384: 00 00 40 FF 00 00 FF 00
0x55b44c6b938c: 00 FF 00 00 00 00 00 00
0x55b44c6b9394: 00000000000000
```

The following example shows how to **configure a negative overhead accounting value**:

```
Router#conf
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#service-policy input policer account user-defined -12
Router(config-if)#commit
```

The following example shows **how to verify the negative overhead accounting value** you configured in the preceding example:

```
Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
RP/0/RSP0/CPU0:Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
Total number of classes:   1
Total number of UBRL classes:  0
Total number of CAC classes:   0
-----------------------------------------------------------
Policy name: policer
Hierarchical depth 1
Interface type HundredGigE
Interface rate 100000000 kbps
Port Shaper rate 0 kbps
Interface handle 0x00000680
ul_ifh 0x00000000, ul_id  0x00000080
uidb index 0x0003
qos_ifh 0x10200020800003
Local port 16, NP 0
Policy map id 0x601C, format 8, uidb index 0x0003
-----------------------------------------------------------
Index 0 Level 0 Class name class-default service_id 0x0 Policy name policer
Node flags: LEAF DEFAULT DEFAULT-ALL
 Stats flags: Policer type 1 Max category 0
 Node Config:
 Police Color aware 0 Type 1 CIR/CBS/PIR/PBS: 10000000kbps/125000000B/0kbps/0B
 Node Result: Class-based stats:Stat ID 0x00000310
 Queue: N/A Stat ID(Commit): 0x00000000
 Stat ID(Drop Curve 0): 0x00000000
 Stat ID(Drop Curve 1): 0x00000000
 Stat ID(Drop Curve 2): 0x00000000
```

```
 Stat ID(Drop Curve 3): 0x00000000
 Police ID
(Token/Conform/Exceed/Violate/Parent-Drop):0x00000010/0x00000310/0x00000311/0x00000312/0x00000313


Hash key (host format):
0x55c9fa43d3dd: 02 00 00 00 00 14 00 14
UIDB Index: 0x0000 (0)
Class ID  : 0x0000 (0)
SW SOURCE-PORT: 0x14 (20)

HW NP-PORT: 0x14 (20)

hash result (host format):
0x55c9fa43d39b: 82 00 33 00 00 00 00 00
0x55c9fa43d3a3: 00 00 00 00 10 00 00 00
0x55c9fa43d3ab: 00 00 00 00 00 00 00 00
0x55c9fa43d3b3: 10 03 00 00 00 00 00 00
0x55c9fa43d3bb: 00 00 00 00 00 00 00 00
0x55c9fa43d3c3: 00 00 00 00 00 00 00 00
0x55c9fa43d3cb: 00 00 00 00 00 00 00 00
0x55c9fa43d3d3: 00 00 00 00 00 00 00 00
0x55c9fa43d3db: 0000
FID0                : 0x00000000 (0)
FID1                : 0x00000000 (0)
Token Bucket        : 0x000010 (16)
Parent Token Bucket : 0x000000 (0)
Stats ID            : 0x000310 (784)
Parent Stats ID     : 0x000000 (0)
COS                 : 0x000003 (3)
L1 Overhead Size    : 0x00000C (12)
L1 Overhead Negative: 0x000001 (1)


hash result (ucode format):
0300000000001400400000000000000800310b88000000c0000000000000000000000000000000000000000000000000000000
hash_tbl_id                         =           0 (0x0)
uidb_index                          =           3 (0x3)
class_id                            =           0 (0x0)
np_port                             =           0 (0x0)
sat_icl_bundle_port                 =           0 (0x0)
ctrl_egress                         =           0 (0x0)
ctrl_ingress                        =           0 (0x0)
RESULT:
RP/0/RSP0/CPU0:Router#
```

To **modify an overhead accounting value,** you must:

1. Remove the existing QoS policy and re-add it.
2. Configure the new overhead accounting value.

```
Router#conf
Router(config)#int hundredGigE 0/0/0/2
Router(config-if)#no service-policy input policer
Router(config-if)#service-policy input policer account user-defined -20
Router(config-if)#commit
Router#sh run int hundredGigE 0/0/0/2
interface HundredGigE0/0/0/2
service-policy input policer account user-defined -20
!
```

To **verify the modified overhead accounting value**:

```
Router#sh qos-ea int hundredGigE 0/0/0/2 input detail
Interface: HundredGigE0_0_0_2 input policy: policer
Total number of classes:   1
Total number of UBRL classes:  0
Total number of CAC classes:   0
------------------------------------------------------
Policy name: policer
Hierarchical depth 1
Interface type HundredGigE
Interface rate 100000000 kbps
Port Shaper rate 0 kbps
Interface handle 0x00000680
ul_ifh 0x00000000, ul_id  0x00000080
uidb index 0x0003
qos_ifh 0x10200020800003
Local port 16, NP 0
Policy map id 0x6020, format 8, uidb index 0x0003
------------------------------------------------------
Index 0 Level 0 Class name class-default service_id 0x0 Policy name policer
Node flags: LEAF DEFAULT DEFAULT-ALL
 Stats flags: Policer type 1 Max category 0
 Node Config:
 Police Color aware 0 Type 1 CIR/CBS/PIR/PBS: 10000000kbps/125000000B/0kbps/0B
 Node Result: Class-based stats:Stat ID 0x00000314
 Queue: N/A Stat ID(Commit): 0x00000000
 Stat ID(Drop Curve 0): 0x00000000
 Stat ID(Drop Curve 1): 0x00000000
 Stat ID(Drop Curve 2): 0x00000000
 Stat ID(Drop Curve 3): 0x00000000
 Police ID
(Token/Conform/Exceed/Violate/Parent-Drop):0x00000018/0x00000314/0x00000315/0x00000316/0x00000317


Hash key (host format):
0x557e9a5ad3dd: 02 00 00 00 00 14 00 14
UIDB Index: 0x0000 (0)
Class ID  : 0x0000 (0)
SW SOURCE-PORT: 0x14 (20)

HW NP-PORT: 0x14 (20)

hash result (host format):
0x557e9a5ad39b: 82 00 53 00 00 00 00 00
0x557e9a5ad3a3: 00 00 00 00 18 00 00 00
0x557e9a5ad3ab: 00 00 00 00 00 00 00 00
0x557e9a5ad3b3: 14 03 00 00 00 00 00 00
0x557e9a5ad3bb: 00 00 00 00 00 00 00 00
0x557e9a5ad3c3: 00 00 00 00 00 00 00 00
0x557e9a5ad3cb: 00 00 00 00 00 00 00 00
0x557e9a5ad3d3: 00 00 00 00 00 00 00 00
0x557e9a5ad3db: 0000
FID0                : 0x00000000 (0)
FID1                : 0x00000000 (0)
Token Bucket        : 0x000018 (24)
Parent Token Bucket : 0x000000 (0)
Stats ID            : 0x000314 (788)
Parent Stats ID     : 0x000000 (0)
COS                 : 0x000003 (3)
L1 Overhead Size    : 0x000014 (20)
L1 Overhead Negative: 0x000001 (1)


hash result (ucode format):
0300000000001400400000000000000000800314b880000014000000000000000000000000000000000000000000000000000000000000
```

```
hash_tbl_id                     =           0 (0x0)
uidb_index                      =           3 (0x3)
class_id                        =           0 (0x0)
np_port                         =           0 (0x0)
sat_icl_bundle_port             =           0 (0x0)
ctrl_egress                     =           0 (0x0)
ctrl_ingress                    =           0 (0x0)
```

Positive Accounting Use Case

If QoS commitment includes Preamble, Frame Delimiter & Interframe Gap and has the following configuration:

```
service-policy input <foo> account user-defined +20
```

For QoS purposes, your router treats this packet as a packet of size = Actual Packet size + 20. Hence, the effective policing and shaping is *reduced* to match the downstream interface.

Negative Accounting Use Case

If QoS commitment to ASR9000 does not include VLAN header information, and has the following configuration:

```
service-policy input <foo> account user-defined -4
```

For QoS purposes, your router treats this packet as a packet of size = Actual Packet size – 4. Hence, the effective policing and shaping is *increased* to match the downstream interface.

Associated Commands

service-policy (overhead accounting)

Traffic Shaping

Traffic shaping allows you to control the traffic flow exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it. Traffic adhering to a particular profile can be shaped to meet downstream requirements, thereby eliminating bottlenecks in topologies with data-rate mismatches.

To match the rate of transmission of data from the source to the target interface, you can limit the transfer of data to one of the following:

- A specific configured rate
- A derived rate based on the level of congestion

The rate of transfer depends on these three components that constitute the token bucket: burst size, mean rate, and time (measurement) interval. The mean rate is equal to the burst size divided by the interval.

When traffic shaping is enabled, the bit rate of the interface does not exceed the mean rate over any integral multiple of the interval. In other words, during every interval, a maximum of burst size can be sent. Within the interval, however, the bit rate may be faster than the mean rate at any given time.

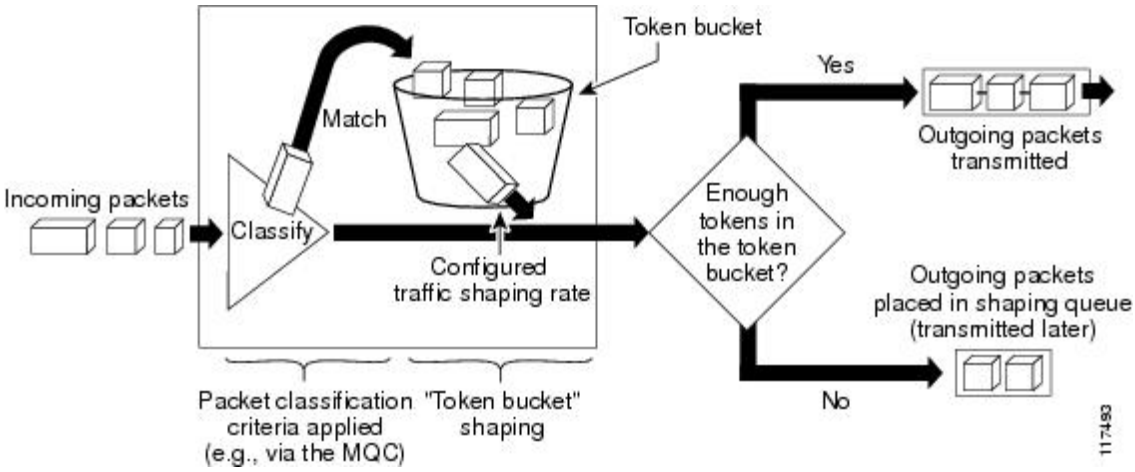
When the peak burst size equals 0, the interface sends no more than the burst size every interval, achieving an average rate no higher than the mean rate. However, when the peak burst size is greater than 0, the interface can send as many as the burst size plus peak burst bits in a burst, if in a previous time period the maximum amount was not sent. Whenever less than the burst size is sent during an interval, the remaining number of bits, up to the peak burst size, can be used to send more than the burst size in a later interval.

Regulation of Traffic with the Shaping Mechanism

When incoming packets arrive at an interface, the packets are classified using a classification technique, such as an access control list (ACL) or the setting of the IP Precedence bits through the Modular QoS CLI (MQC). If the packet matches the specified classification, the traffic-shaping mechanism continues. Otherwise, no further action is taken.

This figure illustrates how a traffic shaping mechanism regulates traffic flow.

Figure 3: How a Traffic Shaping Mechanism Regulates Traffic



Packets matching the specified criteria are placed in the token bucket. The maximum size of the token bucket is the confirm burst (Bc) size plus the Be size. The token bucket is filled at a constant rate of Bc worth of tokens at every Tc. This is the configured traffic shaping rate.

If the traffic shaping mechanism is active (that is, packets exceeding the configured traffic shaping rate already exist in a transmission queue) at every Tc, the traffic shaper checks to see if the transmission queue contains enough packets to send (that is, up to either Bc [or Bc plus Be] worth of traffic).

If the traffic shaper is not active (that is, there are no packets exceeding the configured traffic shaping rate in the transmission queue), the traffic shaper checks the number of tokens in the token bucket. One of the following occurs:

- If there are enough tokens in the token bucket, the packet is sent (transmitted).
- If there are not enough tokens in the token bucket, the packet is placed in a shaping queue for transmission at a later time.

Traffic Policing

In general, traffic policing allows you to control the maximum rate of traffic sent or received on an interface and to partition a network into multiple priority levels or class of service (CoS).

Traffic policing manages the maximum rate of traffic through a token bucket algorithm. The token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on an interface at a given moment in time. The token bucket algorithm is affected by all traffic entering or leaving the interface (depending on where the traffic policy with traffic policing is configured) and is useful in managing network bandwidth in cases where several large packets are sent in the same traffic stream.

Traffic policing is often configured on interfaces at the edge of a network to limit the rate of traffic entering or leaving the network. In the most common traffic policing configurations, traffic that conforms to the CIR is sent and traffic that exceeds is sent with a decreased priority or is dropped. Users can change these configuration options to suit their network needs. Traffic policing also provides a certain amount of bandwidth management by allowing you to set the burst size (Bc) for the committed information rate (CIR). When the peak information rate (PIR) is supported, a second token bucket is enforced and then the traffic policer is called a two-rate policer.

Regulation of Traffic with the Policing Mechanism

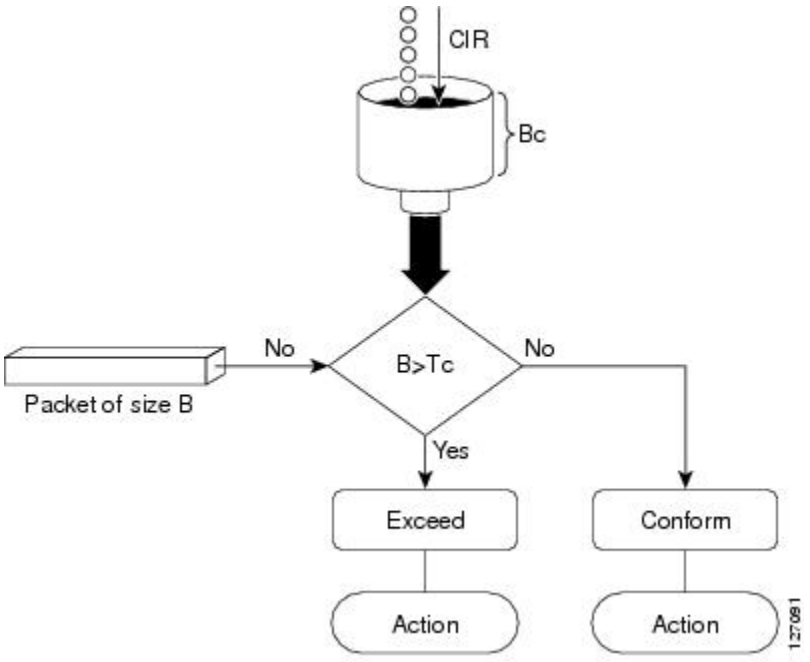
This section describes the single-rate and two-rate policing mechanisms.

Single-Rate Policer

A single-rate, two-action policer provides one token bucket with two actions for each packet: a conform action and an exceed action.

This figure illustrates how a single-rate token bucket policer marks packets as either conforming or exceeding a CIR, and assigns an action.

Figure 4: Marking Packets and Assigning Actions—Single-Rate Policer



The time interval between token updates (Tc) to the token bucket is updated at the CIR value each time a packet arrives at the traffic policer. The Tc token bucket can contain up to the Bc value, which can be a certain number of bytes or a period of time. If a packet of size B is greater than the Tc token bucket, then the packet exceeds the CIR value and a configured action is performed. If a packet of size B is less than the Tc token bucket, then the packet conforms and a different configured action is performed.

Two-Rate Policer

The two-rate policer manages the maximum rate of traffic by using two token buckets: the committed token bucket and the peak token bucket. The dual-token bucket algorithm uses user-configured values to determine the maximum rate of traffic allowed on a queue at a given moment. In this way, the two-rate policer can meter traffic at two independent rates: the committed information rate (CIR) and the peak information rate (PIR).

The committed token bucket can hold bytes up to the size of the committed burst (bc) before overflowing. This token bucket holds the tokens that determine whether a packet conforms to or exceeds the CIR as the following describes:

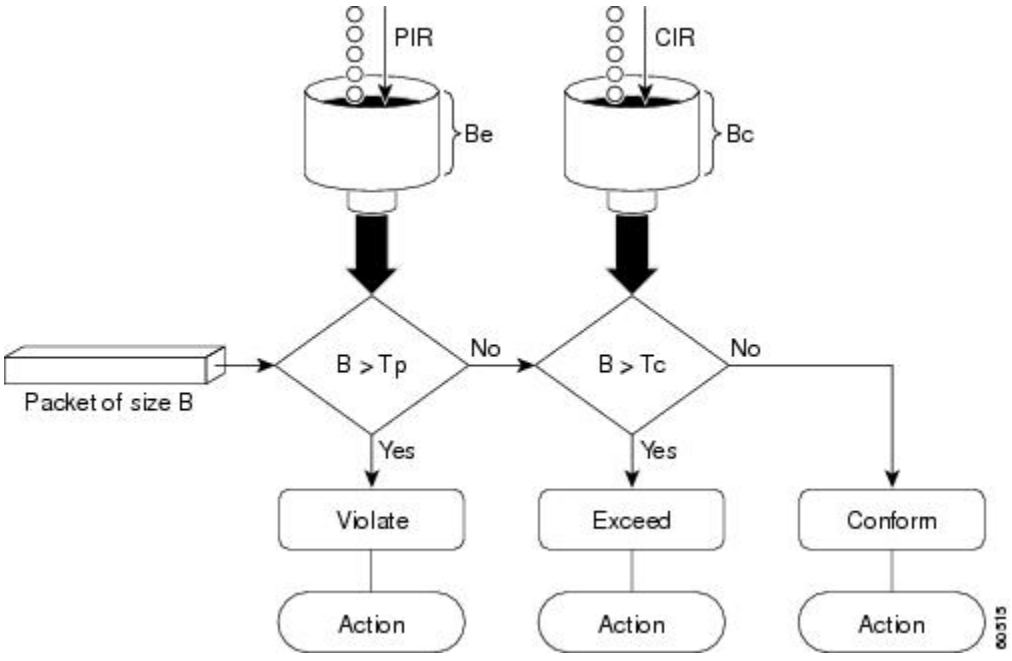
- A traffic stream is conforming when the average number of bytes over time does not cause the committed token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream green.
- A traffic stream is exceeding when it causes the committed token bucket to overflow into the peak token bucket. When this occurs, the token bucket algorithm marks the traffic stream yellow. The peak token bucket is filled as long as the traffic exceeds the police rate.

The peak token bucket can hold bytes up to the size of the peak burst (be) before overflowing. This token bucket holds the tokens that determine whether a packet violates the PIR. A traffic stream is violating when it causes the peak token bucket to overflow. When this occurs, the token bucket algorithm marks the traffic stream red.

The dual-token bucket algorithm provides users with three actions for each packet—a conform action, an exceed action, and an optional violate action. Traffic entering a queue with the two-rate policer configured is placed into one of these categories. Within these three categories, users can decide packet treatments. For instance, packets that conform can be configured to be sent; packets that exceed can be configured to be sent with a decreased priority; and packets that violate can be configured to be dropped.

This figure shows how the two-rate policer marks a packet and assigns a corresponding action to the packet.

Figure 5: Marking Packets and Assigning Actions—2-Rate Policer



For example, if a data stream with a rate of 250 kbps arrives at the two-rate policer, and the CIR is 100 kbps and the PIR is 200 kbps, the policer marks the packet in the following way:

- 100 kbps conforms to the rate
- 100 kbps exceeds the rate
- 50 kbps violates the rate

The router updates the tokens for both the committed and peak token buckets in the following way:

- The router updates the committed token bucket at the CIR value each time a packet arrives at the interface. The committed token bucket can contain up to the committed burst (bc) value.
- The router updates the peak token bucket at the PIR value each time a packet arrives at the interface. The peak token bucket can contain up to the peak burst (be) value.
- When an arriving packet conforms to the CIR, the router takes the conform action on the packet and decrements both the committed and peak token buckets by the number of bytes of the packet.
- When an arriving packet exceeds the CIR, the router takes the exceed action on the packet, decrements the committed token bucket by the number of bytes of the packet, and decrements the peak token bucket by the number of overflow bytes of the packet.
- When an arriving packet exceeds the PIR, the router takes the violate action on the packet, but does not decrement the peak token bucket.

Committed Bursts and Excess Bursts

Unlike a traffic shaper, a traffic policer does not buffer excess packets and transmit them later. Instead, the policer executes a “send or do not send” policy without buffering. During periods of congestion, proper configuration of the excess burst parameter enables the policer to drop packets less aggressively. Therefore, it is important to understand how policing uses the committed (normal) and excess burst values to ensure the router reaches the configured committed information rate (CIR).

Burst parameters are based on a generic buffering rule for routers, which recommends that you configure buffering to be equal to the round-trip time bit-rate to accommodate the outstanding TCP windows of all connections in times of congestion.

The following sections describe committed bursts and excess bursts, and the recommended formula for calculating each of them:

- Committed Bursts, page 24
- Excess Bursts, page 25
- Deciding if Packets Conform or Exceed the Committed Rate, page 26

Committed Bursts

The committed burst (bc) parameter of the police command implements the first, conforming (green) token bucket that the router uses to meter traffic. The bc parameter sets the size of this token bucket. Initially, the token bucket is full and the token count is equal to the committed burst size (CBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the conforming token bucket to send packets:

- If sufficient tokens are in the conforming token bucket when a packet arrives, the meter marks the packet green and decrements the conforming token count by the number of bytes of the packet.
- If there are insufficient tokens available in the conforming token bucket, the meter allows the traffic flow to borrow the tokens needed to send the packet. The meter checks the exceeding token bucket for the number of bytes of the packet. If the exceeding token bucket has a sufficient number of tokens available, the meter marks the packet:

  Green and decrements the conforming token count down to the minimum value of 0.

  Yellow, borrows the remaining tokens needed from the exceeding token bucket, and decrements the exceeding token count by the number of tokens borrowed down to the minimum value of 0.
- If an insufficient number of tokens is available, the meter marks the packet red and does not decrement either of the conforming or exceeding token counts.

Note When the meter marks a packet with a specific color, there must be a sufficient number of tokens of that color to accommodate the entire packet. Therefore, the volume of green packets is never smaller than the committed information rate (CIR) and committed burst size (CBS). Tokens of a given color are always used on packets of that color.

The default committed burst size is the greater of 2 milliseconds of bytes at the police rate or the network maximum transmission unit (MTU).

Committed Burst Calculation

To calculate committed burst, use the following formula:

bc = CIR bps * (1 byte) / (8 bits) * 1.5 seconds

Note 1.5 seconds is the typical round-trip time.

For example, if the committed information rate is 512000 bps, then using the committed burst formula, the committed burst is 96000 bytes.

bc = 512000 * 1/8 * 1.5

bc = 64000 * 1.5 = 96000

Note When the be value equals 0, we recommend that you set the egress bc value to be greater than or equal to the ingress bc value plus 1. Otherwise, packet loss can occur. For example: be = 0 egress bc >= ingress bc + 1

Excess Bursts

The excess burst (be) parameter of the police command implements the second, exceeding (yellow) token bucket that the router uses to meter traffic. The exceeding token bucket is initially full and the token count is equal to the excess burst size (EBS). Thereafter, the meter updates the token counts the number of times per second indicated by the committed information rate (CIR).

The following describes how the meter uses the exceeding token bucket to send packets:

- When the first token bucket (the conforming bucket) meets the committed burst size (CBS), the meter allows the traffic flow to borrow the tokens needed from the exceeding token bucket. The meter marks the packet yellow and then decrements the exceeding token bucket by the number of bytes of the packet.
- If the exceeding token bucket does not have the required tokens to borrow, the meter marks the packet red and does not decrement the conforming or the exceeding token bucket. Instead, the meter performs the exceed-action configured in the police command (for example, the policer drops the packets).

Excess Burst Calculation

To calculate excess burst, use the following formula:

be = 2 * committed burst

For example, if you configure a committed burst of 4000 bytes, then using the excess burst formula, the excess burst is 8000 bytes.

be = 2 * 4000 = 8000

The default excess burst size is 0.

Deciding if Packets Conform or Exceed the Committed Rate

Policing uses normal or committed burst (bc) and excess burst (be) values to ensure that the configured committed information rate (CIR) is reached. Policing decides if a packet conforms or exceeds the CIR based on the burst values you configure. Several factors can influence the policer's decision, such as the following:

- Low burst values—If you configure burst values too low, the achieved rate might be much lower than the configured rate.
- Temporary bursts—These bursts can have a strong adverse impact on throughput of Transmission Control Protocol (TCP) traffic.

It is important that you set the burst values high enough to ensure good throughput. If your router drops packets and reports an exceeded rate even though the conformed rate is less than the configured CIR, use the show interface command to monitor the current burst, determine whether the displayed value is consistently close to the committed burst (bc) and excess burst (be) values, and if the actual rates (the committed rate and exceeded rate) are close to the configured committed rate. If not, the burst values might be too low. Try reconfiguring the burst rates using the suggested calculations in the "Committed Burst Calculation" section on page 25 and the "Excess Burst Calculation" section on page 25.

Two-Rate Three-Color (2R3C) Policer

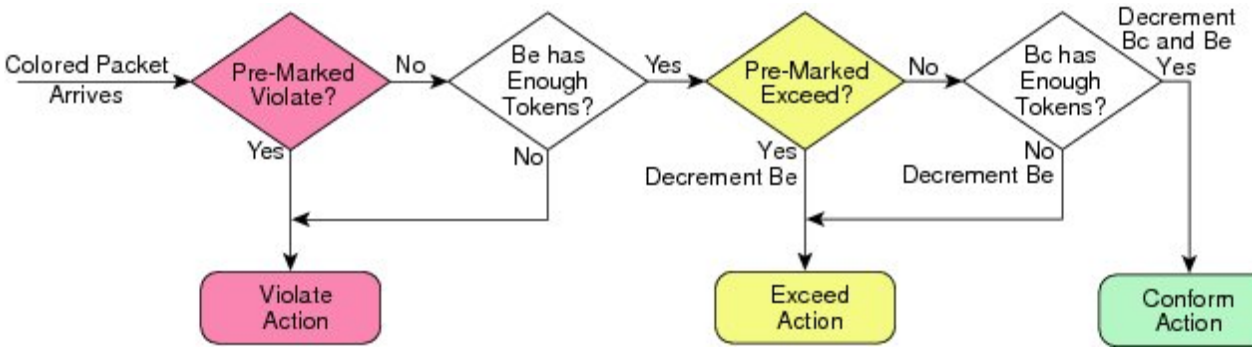
For the SIP 700 card, a two-rate, three-color (2R3C) policer is supported on policy maps for ingress Layer 2 interfaces. The policer reads a preexisting marking—the frame-relay discard-eligibility (FRDE) bit in the packet header—that was set by a policer on a previous network node. By default the FRDE bit is set to 0. At the receiving node, the system uses this bit to determine the appropriate color-aware policing action for the packet:

- To classify the FRDE bit value 0 as conform color, create a conform-color class-map for frde=0 packets. This causes packets to be classified as color green, and the system applies the conform action.
- To classify the FRDE bit value 1 as exceed color, create an exceed-color class-map for frde=1 packets. This causes packets to be classified as color yellow, and the system applies the exceed action.

Note Color-aware policing is not supported for hierarchical QoS.

The 2R3C policing process is shown in this figure.

Figure 6: 2R3C Policing Process Flowchart



Note When ingress QoS policy is applied for 9000v the counters are organized in logical pairs. If 3-color policer is applied, only two counters will be considered:

- Green and Non-Green (Yellow + Red)
- Red and Non-Red (Green + Yellow)

For Cisco ASR 9000 Series 5th Generation High-Density Multi-Rate line cards, the details to calculate excess burst (be) and committed burst (bc) are as follows:

Minimum committed burst value when police rate is configured in packets per second (pps): bc = CIR/1000 * 20

Minimum committed burst value when police rate is configured in units other than pps: bc = CIR * 10

Minimum excess burst value when police rate is configured in pps: be = PIR/1000 * 20

Minimum excess burst value when police rate is configured in units other than pps: be = PIR * 10

Maximum committed burst value when police rate is configured in pps: bc = CIR

Maximum committed burst value when police rate is configured in units other than pps: bc = CIR * 125

Maximum excess burst value when police rate is configured in pps: be = PIR

Maximum excess burst value when police rate is configured in units other than pps: be = PIR * 125

When police rate is configured in pps, the calculated burst value * 256 must be less than 1000000000, else 1000000000/256 is used as the default burst value.

When police rate is configured in units other than pps, the calculated burst values must be less than 1000000000, else 1000000000 is used as the default burst value.

Hierarchical Policing

The Hierarchical Policing feature is an MQC-based solution that supports hierarchical policing on both the ingress and egress interfaces on Cisco ASR 9000 Series Router.

This feature allows enforcement of service level agreements (SLA) while applying the classification submodel for different QoS classes on the inbound interface.

Hiearchical policing provides support at two levels:

- Parent level
- Child level

Multiple Action Set

Packet Marking Through the IP Precedence Value, IP DSCP Value, and the MPLS Experimental Value Setting

In addition to rate-limiting, traffic policing allows you to independently mark (or classify) the packet according to whether the packet conforms or violates a specified rate. Packet marking also allows you to partition your network into multiple priority levels or CoS. Packet marking as a policer action is conditional marking.

Use the traffic policer to set the IP precedence value, IP DSCP value, or Multiprotocol Label Switching (MPLS) experimental value for packets that enter the network. Then networking devices within your network can use this setting to determine how the traffic should be treated. For example, the Weighted Random Early Detection (WRED) feature uses the IP precedence value to determine the probability that a packet is dropped.

If you want to mark traffic but do not want to use traffic policing, see the “Class-based, Unconditional Packet Marking Examples” section to learn how to perform packet classification.

Note Marking IP fields on an MPLS-enabled interface results in non-operation on that particular interface.

Traffic Policing on Layer 2 ATM Interfaces

Traffic policing is supported on the Layer 2 ATM interfaces in the ingress imposition path. The OAM cells are policed along with the user cells unless the QoS policy is explicitly configured to exclude the OAM cells from being policed.

Note Policing is supported for the virtual circuit (VC), and the virtual path (VP) modes. However, policing is not supported for the port mode on the Layer 2 ATM interfaces.

Different match criteria can be used in the policy map with class-default matching all the traffic including the OAM cells.

Policing is performed on the ATM Adaptation Layer type 0 (AAL0) cells but translates to ATM Adaptation Layer type 5 (AAL5) packets as described below:

- AAL5 packet conforms, if all the cells in the packet conform to peak cell rate (PCR) and sustainable cell rate (SCR) buckets.
- AAL5 packet exceeds, if at least one cell does not conform to the SCR bucket.
- AAL5 packet violates, if at least one cell does not conform to the PCR bucket.

The following policer options are supported:

- Rate in cells per second, and percent
- Peak rate in cells per second, and percent
- Delay tolerance in microseconds
- Maximum burst size in cells

The following policer actions are supported on the Layer 2 ATM interfaces in the ingress direction:

- **transmit**
- **drop**
- **set mpls exp imposition** <*exp*> (AToM only)
- **set qos-group** <*qos-group*> (AToM and local switching)
- **set discard-class** <*discard-class*> (AToM and local switching)
- **set atm-clp** (Exceed action only, AToM and local switching)
- **drop** (Violate action)

Multiple policing action is supported on the Layer 2 ATM interfaces using the **set mpls exp imposition** and **set atm-clp** combination.

Restrictions

The following list shows non-supported configuration for traffic policing on a layer 2 ATM interface.

- Applying hierarchical policy maps.
- Configuring service policy on a physical interface.
- Policing of egress traffic.
- Configuring multiple police classes on a policy.
- Configuring conform or violate actions.
- Only **match atm clp** command is supported.

Traffic Policing on a Layer 2 ATM interface: Example

The following example illustrates a sample configuration of traffic policing on a layer 2 ATM interface.

```
policy-map atm
 class class-default
  police rate percent 10
```

```
  !
 !
 end-policy-map
!
interface ATM0/1/0/0.1 l2transport
 pvc 10/100
encapsulation aal0
service-policy input atm
```

Explicit Congestion Notification

In mobile networks, a Base Station Controller (BSC) does not have the knowledge if a particular cell site is being overwhelmed by traffic on a particular link, as it sits behind the ASR9000 series router and it will continue to send traffic even if there is acute congestion on the link. So, once the cell site marks the traffic with the (Explicit Congestion Notification) ECN bits and sends it to the BSC, the BSC will mark the affected session from the congested site with the ECN bit flagged towards the ASR9000 series router.

ECN is an extension to WRED (Weighted Random Early Detection). ECN will mark packets instead of dropping them when the average queue length exceeds a specific threshold value. When configured, ECN helps routers and end hosts to understand that the network is congested and slow down sending packets. However If the number of packets in the queue is above the maximum threshold, packets are dropped based on the drop probability. This is the identical treatment a packet receives when WRED is enabled without ECN configured on the router.

Limitations

- ECN is supported only on ASR 9000 SIP-700 linecards.

For more information on the ECN feature, please refer the *Modular QoS Configuration Guide for Cisco ASR 9000 Series Routers*

Implementing ECN

Implementing ECN requires an ECN-specific field that has two bits—the ECN-capable Transport (ECT) bit and the CE (Congestion Experienced) bit—in the IP header. The ECT bit and the CE bit can be used to make four ECN field combinations of 00 to 11. The first number is the ECT bit and the second number is the CE bit.

ECN Bit Setting

ECT Bit	CE Bit	Combination Indicates
0	0	Not-ECN-capable.
0	1	Endpoints of the transport protocol are ECN-capable.
1	0	Endpoints of the transport protocol are ECN-capable.
1	1	Congestion experienced.

The ECN field combination 00 indicates that a packet is not using ECN. The ECN field combinations 01 and 10—called ECT(1) and ECT(0), respectively—are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable. Routers treat these two field combinations identically. Data senders can use either one or both of these two combinations. The ECN field combination 11 indicates congestion to the endpoints. Packets arriving a full queue of a router will be dropped.

Packet Handling when ECN is enabled

When the number of packets in the queue is below the minimum threshold, packets are transmitted. This happens whether or not ECN is enabled, and this treatment is identical to the treatment a packet receives when WRED only is being used on the network. If the number of packets in the queue is above the maximum threshold, packets are dropped based on the drop probability. This is the identical treatment a packet receives when WRED is enabled without ECN configured on the router. Three different scenarios arise if the number of packets in the queue is between the minimum threshold and the maximum threshold:

- If the ECN field on the packet indicates that the endpoints are ECN-capable (that is, the ECT bit is set to 1 and the CE bit is set to 0, or the ECT bit is set to 0 and the CE bit is set to 1)—and the WRED algorithm determines that the packet should have been dropped based on the drop probability—the ECT and CE bits for the packet are changed to 1, and the packet is transmitted. This happens because ECN is enabled and the packet gets marked instead of dropped.
- If the ECN field on the packet indicates that neither endpoint is ECN-capable (that is, the ECT bit is set to 0 and the CE bit is set to 0), the packet may be dropped based on the WRED drop probability. This is the identical treatment that a packet receives when WRED is enabled without ECN configured on the router.
- If the ECN field on the packet indicates that the network is experiencing congestion (that is, both the ECT bit and the CE bit are set to 1), the packet is transmitted. No further marking is required.

QoS for Bridge-Group Virtual Interfaces

Integrated Routing and Bridging (IRB) provides the ability to route between a bridge group and a routed domain with the help of Bridge-Group Virtual Interface (BVI).

The BVI is a virtual interface within the router that acts like a normal routed interface that does not support bridging, but represents the comparable bridge group to routed interfaces within the router. The interface number of the BVI is the number of the bridge group that the virtual interface represents. The number is the link between the BVI and the bridge group.

For more information on IRB/ BVI, please refer the *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*

QoS on BVI

QoS support on BVI will allow the application of the policy map directly on the virtual interface. This will enable aggregate policing and marking on the virtual interface. The policy can be applied on either the ingress or egress side of the BVI to mark and police traffic going to and from the bridge domain.

These are the QoS features supported on BVI QoS policy:

- Classification
- Policing (hierarchical, conform-aware, conditional marking)
- Marking

QoS Policer Behavior on BVI

Ingress NPU Role

The ingress NPU always processes QoS service policies applied on a BVI for input and output directions on a per-ingress NPU basis.

Policer Enforcement Example

Consider a case where traffic enters the system through multiple ingress NPUs on the same or different line cards. In that case, each of these NPUs will individually police the traffic to the configured rate, resulting in the overall allowed traffic being a multiple of the number of the ingress NPUs and the policer rate.

For example, if you apply a policy map with a policer rate of 1 Mbps for traffic destined to or out of the bridge domain through a single ingress NPU, the total traffic is policed to an aggregate rate of 1 Mbps. If, however, you apply a policy map with a policer rate of 1 Mbps for traffic destined to or out of the bridge domain through two ingress NPUs, each NPU polices the traffic it receives to 1 Mbps. In this specific example, the total traffic that the policer allows may add up to 2 Mbps.

Policer Enforcement for Traffic Transitioning from Layer 3 to Layer 2 or Layer 3 to Layer 2

The table captures the recommendations for policer enforcement at the Layer 3-Layer 2 handoff via BVI.

Policy Direction	Traffic Direction	Policy enforcement
BVI egress	Layer 3 to Layer 2 traffic into the bridge domain	Policy is processed on ingress NPU hosting the Layer 3 interface
BVI ingress	Layer 2 to Layer 3 traffic out the bridge domain	Policy is processed on ingress NPU hosting the Layer 2 interface

Restrictions

QoS on BVI does not support the following:

- Ethernet and SIP 700 linecards (supports only ASR9000 Enhanced Ethernet linecards).
- Bidirectional Forwarding Detection (BFD), Shared Policy Instance, L1 Overhead Accounting.
- VLAN tag, DEI classification and marking.
- Any queue QoS including shape/bandwidth, priority, bandwidth remaining, shaping, queue-limit, and random-detect.
- Percentage policer at lower level without reference policer rate at upper level.
- QoS policy propagation using Border Gateway Protocol (BGP)

Note Queuing can be performed by marking the qos-group and then adding a interface policy that matches the qos-group.

Limitations

- Scale Limitation: 2000 BVI (8 classes per policy)
- Policer Limitation: 8000 policers (per Network Processor)

Classification and Marking for BVI

The following table indicates the QoS fields that are supported on BVI for classification and marking.

	Classification		**Marking**	
	Ingress	Egress	Ingress	Egress
Qos-group	yes	yes	yes	yes
Discard class	yes	yes	yes	yes
Prec (dscp)	yes	yes	yes	yes
vlan	no	no	NA	NA
cos	no	no	no	no
dei	no	no	no	no
src/ DST MAC	yes	no	NA	NA
ipv4 L3 fields	yes	yes	NA	NA
ipv6 L3 fields	yes	yes	NA	NA
cos mark via QG mark/ classify	yes inL2/L3 egress	yes in L2/L3 egress	yes in L2/L3 agress	yes in L2/L3 egress

QoS on IPv4 GRE tunnels

QoS support on IPv4 GRE tunnels enables applying the policy map directly on the IPv4 GRE interface. This enables aggregate policing and marking on the tunnel. The policy can be applied on the ingress side of the tunnel to mark and police payload traffic going into the tunnel. QoS is not supported on traffic egressing out of the tunnel.

For information on GRE tunnels, see the chapter *Implementing Generic Routing Encapsulation* in the*Cisco ASR 9000 Series Aggregation Services Router MPLS Layer 3 VPN Configuration Guide*.

Restrictions

QoS on GRE tunnels does not support the following:

- Any other payload traffic except that of IPv4, IPv6, and MPLS
- Any other GRE tunnel except IPv4 GRE tunnel
- Ethernet and SIP 700 line cards (supports only ASR9000 Enhanced Ethernet line cards)
- Bidirectional Forwarding Detection (BFD), Shared Policy Instance, and L1 Overhead Accounting
- VLAN tag, Drop Eligibility Indicator (DEI) classification, and marking
- Any QoS queuing actions including shape or bandwidth

Note You can perform queuing by applying QoS policy instead on the physical interface.

- Percentage-based policer

Classification and marking for IPv4 GRE tunnel traffic

The following table indicates the support for various payload traffic QoS fields on an IPv4 GRE tunnel for classification and marking.

QoS field	Classification		Marking	
	Ingress	Egress	Ingress	Egress
Precedence	No	Yes	No	Yes
Tunnel Precedence	No	No	No	Yes
VLAN	No	No	No	No
Class of Service (CoS)	No	No	No	No
Drop Eligibility Indicator (DEI)	No	No	No	No
IPv4 L3 field	No	Yes	NA	NA

Example

The following example shows application of a marking output service policy on a GRE tunnel.

```
RP/0/RSP0/CPU0:router# configure terminal
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip1
RP/0/RSP0/CPU0:router(config-if)# service-policy output prec0
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 12.0.0.1 255.255.255.0
RP/0/RSP0/CPU0:router(config-if)# tunnel source TenGigE0/0/0/1
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 15.1.1.2
```

QoS for IPv6 ACLs

The Modular Weapon-X line cards support classification of IPv6 properties based on Source IP, Destination IP, Source Port, Destination Port, Protocol, TOS, Hop Limit, and ACL-based classification.

The supported interfaces are indicated below.

Supported Interface	Ethernet Linecard	Enhanced Ethernet Linecard
L3 main interface	yes	yes
L3 sub-interface	yes	yes
L3 bundle-interface/ sub-interface	yes	yes
L2 main interface	no	yes
L2 sub-interface	no	yes

Supported Interface	Ethernet Linecard	Enhanced Ethernet Linecard
L2 bundle-interface/ sub-interface	no	yes

Policer Granularity and Shaper Granularity

Policer granularity can be configured in the ingress and egress directions. The policer granularity is specified as a permissible percentage variation between the user-configured policer rate, and the hardware programmed policer rate.

Policers applied in either the ingress or egress direction can have any configured rate. However, different line card generations have different granularity as to what rates can be programmed in the hardware. Because of this, a desired rate configured in the policy map may get rounded down to the nearest granularity increment.

Ethernet line cards support a granularity of 64 kbps increments. Hence, if you specify a police rate on Ethernet line cards that is not a multiple of 64, the police rate is rounded down to the nearest 64 kbps increment.

Enhanced Ethernet line cards support a granularity of 8 kbps, so a configured rate is rounded down to the nearest 8 kbps increment.

For all generations of linecards, the minimum police rate is 64 kbps.

To verify the programmed rate of the hardware, run the **show qos interface <interface> <direction>** command. For example:

```
RP/0/RSP0/CPU0:A9K-BNG#show qos interface gigabitEthernet 0/0/0/1 input
Tue Dec 19 16:45:58.260 EDT
….
------------------------------------------------------------------------
Level: 0 Policy: telnet Class: 3play-voip
QueueID: 162 (Port Default)
Policer Profile: 62 (Single)
Conform: 96 kbps(100 kbps) Burst: 1600 bytes (0 Default)
```

Here, the programmed rate is displayed outside the parentheses while the configured rate is displayed within parentheses.

Congestion Management Using DEI

You can manage congestion based on the Drop Eligible Indicator (DEI) bit that is present in 802.1ad frames and 802.1ah frames. Random early detection based on the DEI value is supported on 802.1ad packets for:

- Layer 2 subinterfaces
- Layer 2 main interfaces
- Layer 3 main interfaces
- Ingress and egress

Note If there are any marking actions in the policy, the marked values are used for doing WRED.

How to Configure QoS Congestion Management

Configuring Guaranteed and Remaining Bandwidths

The **bandwidth** command allows you to specify the minimum guaranteed bandwidth to be allocated for a specific class of traffic. MDRR is implemented as the scheduling algorithm.

The **bandwidth remaining** command specifies a weight for the class to the MDRR. The MDRR algorithm derives the weight for each class from the bandwidth remaining value allocated to the class. If you do not configure the **bandwidth remaining** command for any class, the leftover bandwidth is allocated equally to all classes for which **bandwidth remaining** is not explicitly specified.

Guaranteed Service rate of a queue is defined as the bandwidth the queue receives when all the queues are congested. It is defined as:

Guaranteed Service Rate = minimum bandwidth + excess share of the queue

Restrictions

The amount of bandwidth configured should be large enough to also accommodate Layer 2 overhead.

The **bandwidth** command is supported only on policies configured on outgoing interfaces.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** {*rate* [*units*]| **percent** *value*}
5. **bandwidth remaining percent** *value*
6. **exit**
7. **class** *class-name*
8. **bandwidth** {*rate* [*units*] | **percent** *value*}
9. **bandwidth remaining percent** *value*
10. **exit**
11. **exit**
12. **interface** *type interface-path-id*
13. **service-policy** {**input** | **output**} *policy-map*
14. Use the **commit** or **end** command.
15. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 4	**bandwidth** {*rate* [*units*]| **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 50`	Specifies the bandwidth allocated for a class belonging to a policy map and enters the policy map class configuration mode. In this example, class class1 is guaranteed 50 percent of the interface bandwidth.
Step 5	**bandwidth remaining percent** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	Specifies how to allocate leftover bandwidth to various classes. **Note** The remaining bandwidth of 40 percent is shared by class class1 and class2 (see Steps 8 and 9) in a 20:80 ratio: class class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
Step 6	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 7	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of a different class whose policy you want to create or change.
Step 8	**bandwidth** {*rate* [*units*] | **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 10`	Specifies the bandwidth allocated for a class belonging to a policy map. In this example, class class2 is guaranteed 10 percent of the interface bandwidth.
Step 9	**bandwidth remaining percent** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`	Specifies how to allocate leftover bandwidth to various classes. **Note** The remaining bandwidth of 40 percent is shared by class class1 and class2 (see Steps 8 and 9) in a 20:80 ratio: class

	Command or Action	Purpose
		class1 receives 20 percent of the 40 percent, and class class2 receives 80 percent of the 40 percent.
Step 10	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface POS 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 15	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Guaranteed Bandwidth

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}
5. **exit**
6. **class** *class-name*
7. **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}
8. **exit**
9. **class** *class-name*
10. **bandwidth** {*rate* [*units*]| **percent** *percentage-value*}
11. **exit**
12. **exit**
13. **interface** *type interface-path-id*
14. **service-policy** {**input** | **output**} *policy-map*
15. **end** or **commit**
16. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 4	**bandwidth** {*rate* [*units*]| **percent** *percentage-value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 40`	Enters policy map class configuration mode. • Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class1 is guaranteed 40 percent of the interface bandwidth.

	Command or Action	Purpose
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth** {*rate* [*units*]| **percent** *percentage-value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 40`	Enters policy map class configuration mode. • Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class2 is guaranteed 40 percent of the interface bandwidth.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change.
Step 10	**bandwidth** {*rate* [*units*]| **percent** *percentage-value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 20`	Enters policy map class configuration mode. • Specifies the bandwidth allocated for a class belonging to a policy map. • In this example, class class-default is guaranteed 20 percent of the interface bandwidth.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 12	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 13	**interface** *type interface-path-id* **Example:**	Enters interface configuration mode and configures an interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/2/0/0`	
Step 14	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • In this example, the traffic policy evaluates all traffic leaving that interface.

Step 15	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 16	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Bandwidth Remaining

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth remaining percent** *percentage-value*
5. **exit**
6. **class** *class-name*

7. **bandwidth remaining percent** *percentage-value*
8. **exit**
9. **class** *class-name*
10. **bandwidth remaining percent** *percentage-value*
11. **exit**
12. **exit**
13. **interface** *type interface-path-id*
14. **service-policy** {**input** | **output**} *policy-map*
15. **end** or **commit**
16. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 4	**bandwidth remaining percent** *percentage-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 40`	Specifies how to allocate leftover bandwidth for class class1.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
Step 7	**bandwidth remaining percent** *percentage-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 40`	Specifies how to allocate leftover bandwidth for class class2.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change.
Step 10	**bandwidth remaining percent** *percentage-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 20`	Specifies how to allocate leftover bandwidth for class class-default.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 12	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 13	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/2/0/0`	Enters interface configuration mode and configures an interface.
Step 14	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. • In this example, the traffic policy evaluates all traffic leaving that interface.
Step 15	**end** or **commit** **Example:**	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes:

	Command or Action	Purpose

	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 16	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Low-Latency Queueing with Strict Priority Queueing

The **priority** command configures LLQ with strict priority queuing (PQ) that allows delay-sensitive data such as voice to be dequeued and sent before packets in other queues are dequeued. When a class is marked as high priority using the **priority** command, you must configure a policer to limit the priority traffic. This configuration ensures that the priority traffic does not constrain all the other traffic on the line card, which protects low priority traffic from limitations. Use the **police** command to explicitly configure the policer.

Note Eight levels of priorities are supported: priority level 1, priority level 2, priority level 3, priority level 4, priority level 5, priority level 6, priority level 7 and the priority level normal. If no priority level is configured, the default is priority level normal.

Note The output of the show policy-map interface command is inconsistent for priority level 1, priority level 2, priority level 3 that is configured on the bundle interface, when the interface has either of these combinations:

- 3rd generation of ASR 9000 LC with SE and TR versions
- 4th generation of ASR 9000 LC with SE and 3rd generation of ASR 9000 LC with TR versions

Additionally, for priority level 3 configuration the show policy-map interface command output is inconsistent when the bundle interface has a combination of 3rd generation ASR 9000 LC with TR and 4th generation of ASR 9000 LC with TR versions.

In order to display consistent show policy-map interface, use **show policy-map interface bundle-eth** <*number*> **[input | output] member** {*interface type interface-path-id*} command.

Restrictions

- Unused priority queues cannot be used for a different priority level.
- The eight priority levels can be configured only on egress of main physical interface or main bundle interface.
- Eight priority levels work on Cisco ASR 9000 High Density 100GE Ethernet line cards only.
- The policy-map with eight priorities must have only one queuing class at the parent level of the priority class.
- If the policy-map has a parent class, the parent class cannot have bandwidth configured.
- Within a policy map, you can give one or more classes priority status. When multiple classes within a single policy map are configured as priority classes, all traffic from these classes is queued to the same single priority queue.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-burst** *peak-burst* [*burst-units*]] [**peak-rate** *value* [*units*]]
5. **exceed-action** *action*
6. **exit**
7. **priority**[**level** *priority_level*]
8. **exit**
9. **exit**
10. **interface** *type interface-path-id*
11. **service-policy** {**input** | **output**} *policy-map*
12. Use the **commit** or **end** command.
13. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map voice`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class voice`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-burst** *peak-burst* [*burst-units*]] [**peak-rate** *value* [*units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 250`	Configures traffic policing and enters policy map police configuration mode. In this example, the low-latency queue is restricted to 250 kbps to protect low-priority traffic from starvation and to release bandwidth.
Step 5	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit.
Step 6	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map class configuration mode.
Step 7	**priority**[**level** *priority_level*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# priority level 1`	Specifies priority to a class of traffic belonging to a policy map. If no priority level is configured, the default is priority 1.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to Global Configuration mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet`	Enters interface configuration mode, and configures an interface.
Step 11	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 12	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 13	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Shaping

Traffic shaping allows you to control the traffic exiting an interface to match its transmission to the speed of the remote target interface and ensure that the traffic conforms to policies contracted for it.

Shaping performed on incoming and outgoing interfaces is done at the Layer 2 level and includes the Layer 2 header in the rate calculation.

Restrictions

- The bandwidth, priority and shape average commands should not be configured together in the same class.

- A flat port-level shaper requires a child policy with 100% bandwidth explicitly allocated to the class-default.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **shape average** {**percent** *value* | *rate* [*units*]}
5. **exit**
6. **exit**
7. Specifies the name of the class whose policy you want to create or change.**interface** *type interface-path-id*
8. **service-policy** {**input** | **output**} *policy-map*
9. Use the **commit** or **end** command.
10. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**shape average** {**percent** *value*	*rate* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`	Shapes traffic to the indicated bit rate according to average rate shaping in the specified units or as a percentage of the bandwidth.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 6	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 7	Specifies the name of the class whose policy you want to create or change.**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet`	Enters interface configuration mode and configures an interface.
Step 8	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 10	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (Two-Rate Color-Blind)

Traffic policing allows you to control the maximum rate of traffic sent or received on an interface. This section provides the procedure for configuring two-rate color-blind traffic policing.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*

4. **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]]
5. **conform-action** *action*
6. **exceed-action** *action*
7. **exit**
8. **exit**
9. **exit**
10. **interface** *type interface-path-id*
11. **service-policy** {**input** | **output**} *policy-map*
12. Use the **commit** or **end** command.
13. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**police rate** {[*units*]	**percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 250000`	Configures traffic policing and enters policy map police configuration mode. The traffic policing feature works with a token bucket algorithm.
Step 5	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action set mpls experimental topmost 3`	Configures the action to take on packets that conform to the rate limit. The *action* argument is specified by one of these keywords: • **drop**—Drops the packet. • **set**—Has these keywords and arguments:

	Command or Action	Purpose
		discard-class *value*—Sets the discard class value. Range is 0 to 7. **dscp** —Sets the differentiated services code point (DSCP) value and sends the packet. **mpls experimental** {**topmost** | **imposition**} *value*—Sets the experimental (EXP) value of the Multiprotocol Label Switching (MPLS) packet topmost label or imposed label. Range is 0 to 7. **precedence** —Sets the IP precedence and sends the packet. **qos-group**—Sets the QoS group value. Range is from 0 to 511. • **transmit**—Transmits the packets.
Step 6	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action set mpls experimental topmost 4`	Configures the action to take on packets that exceed the rate limit. The *action* argument is specified by one of the keywords specified in Step 5 .
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet`	Enters configuration mode and configures an interface.
Step 11	**service-policy** {**input** | **output**} *policy-map* **Example:**	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	
Step 12	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 13	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Traffic Policing (2R3C)

This section provides the procedure for configuring two-rate three-color traffic policing. It is applicable to SIP 700 line cards on the ingress side only.

SUMMARY STEPS

1. **configure**
2. **class-map** [**match-all**][**match-any**] *class-map-name*
3. **match [not] fr-de***fr-de-bit-value*
4. **policy-map** *policy-name*
5. **class** *class-name*
6. **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]]
7. **conform-color** *class-map-name*
8. **exceed-color** *class-map-name*
9. **conform-action** *action*
10. **exceed-action** *action*
11. **exit**
12. **exit**
13. **exit**
14. **interface** *type interface-path-id*
15. **service-policy** *policy-map*
16. Use the **commit** or **end** command.

17. **show policy-map interface** *type interface-path-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**match-all**][**match-any**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-all match-not-frde`	(Use with SIP 700 line card, ingress only) Creates or modifies a class map that can be attached to one or more interfaces to specify a matching policy and enters the class map configuration mode.
Step 3	**match [not] fr-de***fr-de-bit-value* **Example:** `RP/0/RSP0/CPU0:router(config)# match not  fr-de 1`	(Use with SIP 700 line card, ingress only) Specifies the matching condition: • Match *not* fr-de 1 is typically used to specify a conform-color packet. • Match fr-de 1 is typically used to specify an exceed-color packet.
Step 4	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 5	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 6	**police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 768000 burst 288000 peak-rate 1536000 peak-burst 576000`	Configures traffic policing and enters policy map police configuration mode. The traffic policing feature works with a token bucket algorithm.
Step 7	**conform-color** *class-map-name* **Example:**	(Use with SIP 700 line card, ingress only)

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-color match-not-frde`	Configures the class-map name to assign to conform-color packets.
Step 8	**exceed-color** *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-color match-frde`	(Use with SIP 700 line card, ingress only) Configures the class-map name to assign to exceed-color packets.
Step 9	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action set mpls experimental topmost 3`	Configures the action to take on packets that conform to the rate limit. The *action* argument is specified by one of these keywords: • **drop**—Drops the packet. • **set**—Has these keywords and arguments: **discard-class** *value*—Sets the discard class value. Range is 0 to 7. **dscp** *value*—Sets the differentiated services code point (DSCP) value and sends the packet. **mpls experimental** {**topmost** | **imposition**} *value*—Sets the experimental (EXP) value of the Multiprotocol Label Switching (MPLS) packet topmost label or imposed label. Range is 0 to 7. **precedence** *precedence*—Sets the IP precedence and sends the packet. **qos-group**—Sets the QoS group value. Range is 0 to 63. • **transmit**—Transmits the packets.
Step 10	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action set mpls experimental topmost 4`	Configures the action to take on packets that exceed the rate limit. The *action* argument is specified by one of the keywords specified in Step 5.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 12	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 13	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 14	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface pos 0/5/0/0`	Enters configuration mode and configures an interface.
Step 15	**service-policy** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy policy1`	Attaches a policy map to an input interface to be used as the service policy for that interface.
Step 16	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 17	**show policy-map interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface POS 0/2/0/0`	(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring Hierarchical Policing

Hierarchical policing provides support at two levels:

- Parent level
- Child level

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*

4. **service-policy** *policy-map-name*
5. **police rate percent** *percentage*
6. **conform-action** *action*
7. **exceed-action** *action*
8. **end** or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy child`	Attaches a policy map to an input or output interface to be used as the service policy for that interface.
Step 5	**police rate percent** *percentage* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate percent 50`	Configures traffic policing and enters policy map police configuration mode.
Step 6	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action transmit`	Configures the action to take on packets that conform to the rate limit. The allowed action is: **transmit**—Transmits the packets.
Step 7	**exceed-action** *action* **Example:**	Configures the action to take on packets that exceed the rate limit. The allowed action is: **drop**—Drops the packet.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	

Step 8	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Traffic Policing for BVI

The traffic policy configuration defines the information rate, percentage of link bandwidth and the action taken on the packets (conform/ violate/ exceed) for the BVI. The configured policer rate on the BVI is effective NP-wise. If two interfaces are in one NP, BVI traffic from these two interfaces is under one policer. Traffic from the other interfaces and/ or on another NP is not affected by the policer. You can use the command, **show controller np ports** to check for interfaces on a particular NP.

Note To avoid the problem of system idle in the configuration mode while performing IRB QoS in-place modification, you can remove the QoS policy from the BVI before modifying related class-maps or policy-maps.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]]
5. **conform-action** *action*
6. **exceed-action** *action*
7. **violate-action** *action*

8. **exit**
9. **exit**
10. **exit**
11. **interface** *type interface-path-id*
12. **service-policy** {**input** | **output**} *policy-map*
13. **end** or **commit**
14. **show policy-map interface** *type interface-path-id* [**input** | **output**]*interface-path-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**police rate** {[*units*] | **percent** *percentage*} [**burst** *burst-size* [burst-*units*]] [**peak-burst** *peak-burst* [burst-*units*]] [**peak-rate** *value* [*units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 250000`	Configures traffic policing and enters policy map police configuration mode. The traffic policing feature works with a token bucket algorithm. **Note** **police rate** is more suitable for regular, flat policy maps. You may use the **police percent** command for parent/child policy maps.
Step 5	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action set prec 1`	Configures the action to take on packets that conform to the rate limit. The *action* argument is specified by one of these keywords: • **drop**—Drops the packet. • **set**—Has these keywords and arguments: **discard-class** *value*—Sets the discard class value. Range is 0 to 7. **dscp** —Sets the differentiated services code point (DSCP) value and sends the packet.

	Command or Action	Purpose
		precedence —Sets the IP precedence and sends the packet. **qos-group**—Sets the QoS group value. Range is 0 to 63. • **transmit**—Transmits the packets.
Step 6	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit. The *action* argument is specified by one of the keywords specified in Step 5.
Step 7	**violate-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# violate-action drop`	Configures the action to take on packets that exceed the rate limit. The *action* argument is specified by one of the keywords specified in Step 5.
Step 8	**exit** **Example:** RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit	Returns the router to policy map class configuration mode.
Step 9	**exit** **Example:** RP/0/RSP0/CPU0:router(config-pmap-c)# exit	Returns the router to policy map configuration mode.
Step 10	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 11	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface BVI 10`	Specifies the BVI to which the Qos policy will get attached to .
Step 12	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output BVI to be used as the service policy for that interface. **Note** Policer for BVI is aggregated per Network processor. 500M policer for two interfaces of the same NP results in the total policed rate per NP as 500M. • In this example, the traffic policy evaluates all traffic leaving that interface.

	Command or Action	Purpose

Step 13	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 14	**show policy-map interface** *type interface-path-id* [**input** | **output**]*interface-path-id* **Example:** `RP/0/RSP0/CPU0:router# sh policy-map int BVI 1 input member gig 0/1/0/29`	(Optional) Displays policy configuration information for all classes configured for all service policies on a NP which the specified interface (gig 0/1/0/29) belongs to.

Configuring ECN

ECN helps routers and end hosts to understand that the network is congested and slow down the rate at which packets are transmitted.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** [*percent* | *value*]
5. **random-detect** { **default** | **discard-class** | **dscp** | **precedence** }
6. **random-detect ecn**
7. **exit**
8. **exit**
9. **end** or **commit**
10. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**bandwidth** [*percent* | *value*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth 100`	Specifies or modifies the bandwidth allocated for a class in a specific policy-map. **Note** ECN can be configured with any queuing action, such as , bandwidth, shaping, etc.
Step 5	**random-detect** { **default** | **discard-class** | **dscp** | **precedence** } **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# random-detect dscp 1 1000 packets 2000 packets`	Configures the WRED profile. WRED profile entry is required to apply ECN for a particular class.
Step 6	**random-detect ecn** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# random-detect ecn`	Enables ECN.
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.

	Command or Action	Purpose

Step 9	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# end`
	or
	`RP/0/RSP0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show policy-map interface** *type interface-path-id* [**input** | **output**] **Example:** `RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/2/0/0`	(Optional) Displays statistics for all classes configured for all service policies on the specified interface. If ECN is enabled, displays ECN marking information for the specified interface.

Dynamic and Static Buffer Allocation

Note This feature is supported only on the **IOS XR 64-Bit** operating sytem.

The Cisco ASR 9000 4th Generation QSFP28-based dense 100GE line cards and Cisco ASR 9000 5h Generation line cards allow either dynamic or static buffer allocation for egress queuing. This ensures lesser packet drops or low latency for priority packets across ports sharing the egress buffer as required.

You can allocate (or carve) the buffer space using one of two available options: dynamic or static.

Note Because the buffer spaces must accommodate packet buffering as well as overhead, the actual packet buffering capability depends on the frame size, frame alignment to multiples of buffer particles (32 B), and active queue scale.

Dynamic Buffer Allocation

Dynamic buffer allocation is the default buffer allocation mode on Cisco ASR 9000 4th Generation QSFP28-based dense 100GE line cards and Cisco ASR 9000 5th Generation line cards. On these cards, each network processor has 3 GB of egress queuing memory.The dynamic buffer allocation option manages the 3 GB of egress queuing memory available for packet buffering by defining three logical regions:

- Shared region—2.5 GB of egress queuing memory is available for shared packet buffering among all ports of the network processor. These ports consume the shared region first on a first-come and first-serve basis.
- Reserved region—200 MB of egress queuing memory is reserved for internal ports and critical control protocol protection, yield at 4-ms buffering per port. This region is not available for transit packet buffering.
- Protection region—300 MB of egress queuing memory is available for port guarantees and for priority traffic protection. Ports that have not exhausted their guaranteed 10 ms of buffering can use this region. This region is also used for ports that have not exhausted their 100-ms limit to packet buffer under congestion. Transit priority packets are scheduled first out of any port; so, priority buffering is rare and happens only for transient bursts on a well-designed network.

Advantages

The advantages that the dynamic buffer allocation offers are:

- Priority protection
- Protection of critical control traffic
- Guaranteed 10 ms of port buffering on all ports
- Up to 100 ms of port buffering is available in the shared region when only 50% of the network processor ports are congested.

Restrictions

Dynamic buffer allocation mode does not guarantee per-priority buffering limits and hence does not support 8-priority user policies.

Static Buffer Allocation

The static buffer mode allocates buffers equally, on a per-port or per-port-group basis. This allocation guarantees a predefined and fixed amount of buffer memory for each port under traffic congestion. In static buffer allocation mode, buffers allocated and unused by a port cannot be used by other congested ports on the same network processor.

Advantages

This mode guarantees port buffers, provides priority protection within a port's queues, and supports per priority buffer limits for 2-priority and 8-priority policies.

Configuring the Static Buffer Option

The default option for buffer allocation at egress for the Cisco ASR 9000 4th Generation QSFP28-based dense 100GE line cards and Cisco ASR 9000 5th Generation line cards is dynamic. To use the static buffer allocation option, you must first enable it as detailed in this section.

Configuration Example

You must accomplish the following to enable the static buffer allocation:

1. To enable the static buffer allocation mode, use the **hw-module buffer-carve-mode** command.

Note If WRED is configured, then this buffering feature is not applicable. This is because user-defined WRED takes precedence over this feature.

2. Reload the line cards for the changes to take effect.

```
RP/0/RSP0/CPU0:ios#conf
RP/0/RSP0/CPU0:ios(config)#hw-module buffer-carve-mode location 0/0/CPU0 static
RP/0/RSP0/CPU0:ios(config)#commit
```

Running Configuration

```
RP/0/RSP0/CPU0:ios#sh running-config
Building configuration...
!! IOS XR Configuration 7.2.1.10I
!! Last configuration change at Fri Jan 10 15:11:39 2020 by root
!
!
hw-module buffer-carve-mode location 0/0/CPU0 static
end
RP/0/RSP0/CPU0:ios#
```

Verification

If the show configuration command has the entry **hw-module buffer-carve-mode location 0/0/CPU0 static**, it indicates that the static buffer allocation mode is enabled. Else, the dynamic mode is enabled.

You can also run the **show qoshal default-queue** command to view the buffer carving mode per port.

```
For DYNAMIC MODE:
================

RP/0/RSP0/CPU0:ios#sh qoshal default-queue interface hundredGigE 0/0/0/0
TY Options argc:6
nphal_show_chk -p 32800 default-queue -m HundredGigE0_0_0_0
Done
Interface Default Queues : HundredGigE0_0_0_0
================================================================
Port 0 NP 0 TM Port 0
  Global Inst buffer usage       : 0 Kbytes
  Port Buffer allocate mode      : Dynamic
  Port Inst buffer usage         : 0 Kbytes
  Total active queues            : 4
  Total active non-empty queues  : 0
  Total leaked queues            : 0
```

```
   Egress: QID 0x30 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/0 Priority: Priority
 1 Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2D4/0x2D5
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0

   Egress: QID 0x31 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/1 Priority: Priority
 2 Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2D9/0x2DA
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0

   Egress: QID 0x32 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/2 Priority: Priority
 Normal Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2DE/0x2DF
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0

   Egress: QID 0x33 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/3 Priority: Priority
 Normal Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2E3/0x2E4
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0

RP/0/RSP0/CPU0:ios#

For STATIC MODE:
===============
RP/0/RSP0/CPU0:ios#sh qoshal default-queue interface hundredGigE 0/0/0/0
TY Options argc:6
nphal_show_chk -p 32800 default-queue -m HundredGigE0_0_0_0
Done
Interface Default Queues : HundredGigE0_0_0_0
================================================================
Port 0 NP 0 TM Port 0
  Global Inst buffer usage       : 0 Kbytes
  Port Buffer allocate mode      : Static
  Drop Thresholds: 8P Mode
   P1 : 56ms | P2 : 52ms | P3 : 44ms | P4 : 36ms | P5 : 28ms |   P6 : 20ms | P7/Pn : 10ms
  Drop Thresholds: P1P2Pn Mode
   P1 : 56ms | P2 : 52ms | Pn : 44 ms
  Port Inst buffer usage         : 0 Kbytes
  Total active queues            : 4
  Total active non-empty queues  : 0
  Total leaked queues            : 0

   Egress: QID 0x30 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/0 Priority: Priority
 1 Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2C0/0x2C1
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0

   Egress: QID 0x31 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/1 Priority: Priority
 2 Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2C5/0x2C6
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0
   Egress: QID 0x32 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/2 Priority: Priority
 Normal Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2CA/0x2CB
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0
```

```
   Egress: QID 0x33 Entity[np/tm/chunk/level/index/offset]: 0/0/0/4/6/3 Priority: Priority
 Normal Inst Queue length: 0(bytes) Avg queue len: 0(bytes)
            StatIDs: Commit/Drop: 0x2CF/0x2D0
            Statistics(Pkts/Bytes):
              Total Xmt 0/0 Dropped 0/0

RP/0/RSP0/CPU0:ios#
```

Related Commands **hw-module buffer-carve-mode**

Configuration Examples for Configuring Congestion Management

Service Fragment Configurations: Example

This example shows the service-fragment premium being created.

```
policy-map tsqos-port-policy
    class class-default
        shape 500 mbps
    class dscp1
        shape 1 Gbps
        service-fragment premium
        end-policy
      exit
```

This example shows the service-fragment premium being referred (at the sub-interface):

```
policy-map tsqos-subif-policy-premium
    class class-default
        fragment premium
        shape 20 mbps
        bandwidth remaining ratio 20
        service-policy subif-child
        end-policy
       exit
```

Traffic Policing for BVI: Example

The following example shows how to configure traffic policing for a BVI:

```
policy-map p1
 class c1
   police rate 10
    conform-action set prec 1
    exceed-action drop
   exit
  exit
exit
interface BVI 10
 service-policy output p1
```

Configuration example for L2VPN (sub-interface):

```
interface TE0/2/1/2.1 l2transport
  encapsulation dot1q50
  rewrite ingress tag pop1 symmetric (for dot1q sub)
l2vpn
 bridge group BVI
  bridge-domain BVI
```

```
   interface TE0/2/1/2.1
   !
   routed interface BVI1
  !
 !
```

ECN: Example

The following example shows how to run the **random-detect ecn** command to configure ECN:

```
config
policy-map p1
class c1
bandwidth 100
random-detect dscp 1 1000 packets 2000 packets
random-detect ecn
exit
exit
commit
```

Hierarchical Policing: Example

Additional References

These sections provide references related to implementing QoS congestion management.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Gett Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Mod of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Ro of Cisco Cisco ASR 9000 Series Aggregation Services Re Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

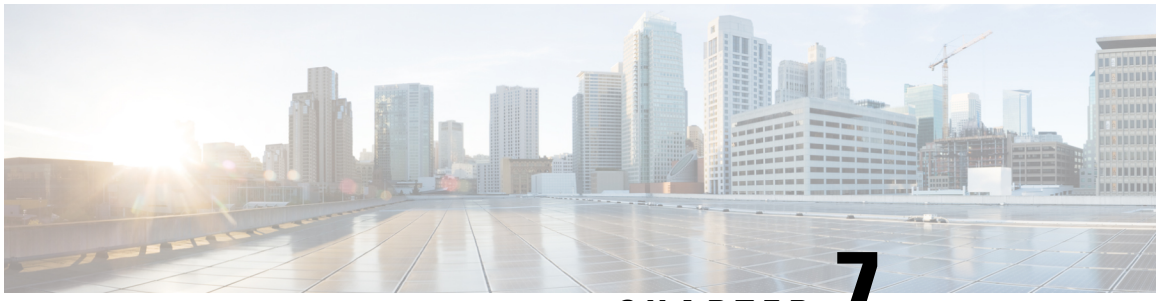
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html
For information about fabric scheduling, virtual output queuing (VOQ), and more, search for “voq” on community.cisco.com.	community.cisco.com
For information about session id BRKSPG-2904 and BRKARC-2003, search on Cisco Live on-demand library.	https://www.ciscolive.com/on-demand/on-demand-library.htr



CHAPTER 7

Configuring Modular QoS Service Packet Classification

Packet 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 module provides the conceptual and configuration information for QoS packet classification.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Classification Based on DEI	yes	no
Class-Based Unconditional Packet Marking	yes	yes
In-Place Policy Modification	yes	yes
IPv6 QoS	yes	yes
Packet Classification and Marking	yes	yes
Policy Inheritance	yes	yes
Port Shape Policies	yes	no
Shared Policy Instance	yes	no

Feature History for Configuring Modular QoS Packet Classification and Marking on Cisco ASR 9000 Series Routers

Release	Modification

Release 3.7.2	The Class-Based Unconditional Packet Marking feature was introduced on ASR 9000 Ethernet Line Cards. The IPv6 QoS feature was introduced on ASR 9000 Ethernet Line Cards. (QoS matching on IPv6 ACLs is not supported.) The Packet Classification and Marking feature was introduced on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The Class-Based Unconditional Packet Marking feature was supported on the SIP 700 for the ASR 9000. The Packet Classification and Marking feature was supported on the SIP 700 for the ASR 9000. The Policy Inheritance feature was introduced on ASR 9000 Ethernet Line Cards and on the SIP 700 for the ASR 9000. The Shared Policy Instance feature was introduced on ASR 9000 Ethernet Line Cards.
Release 4.0.0	The Classification Based on DEI feature was introduced on ASR 9000 Ethernet Line Cards. The In-Place Policy Modification feature was introduced on ASR 9000 Ethernet Line Cards and on the SIP 700 for the ASR 9000. The IPv6 QoS feature was supported on the SIP 700 for the ASR 9000. Support for three stand-alone marking actions and three marking actions as part of a policer action in the same class was added on the SIP 700 for the ASR 9000. (ASR 9000 Ethernet Line Cards support two stand-alone marking actions and two marking actions as part of a policer action in the same class.)
Release 4.0.1	Support for the port shape policies feature was introduced on ASR 9000 Ethernet Line Cards.
Release 4.2.1	QoS on satellite feature was added.
Release 5.1.1	The QoS Offload on satellite feature was added.
	The Inter Class Policer Bucket Sharing feature was added. This feature is applicable to all ASR 9000 Enhanced Ethernet line cards except the first generation and SIP 700 line cards.
Release 5.1.2	The QoS Offload on 901 feature was added.
	Port Shaper Policy Support on L2 Fabric ICL Interface
Release 5.2.0	The QPPB on GRE Tunnel Interfaces feature was added.
Release 6.0	Classification Support for Ethernet-Services ACL was added.
Release 6.1.2	New Chapter, "Configuring QoS on the Satellite System" was created that contains information about QoS Offload.
Release 6.2.1	Support for Multiple QoS Policy feature was added.

Prerequisites for Configuring Modular QoS Packet Classification

These prerequisites are required for configuring modular QoS packet classification and marking on your network:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must be familiar with Cisco IOS XR QoS configuration tasks and concepts.

Information About Configuring Modular 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.

Note IPv6-based classification is supported only on Layer 3 interfaces.

Traffic Class Elements

The purpose of a traffic class is to classify traffic on your router. Use the **class-map** command to define a traffic class.

A traffic class contains three major elements: a name, a series of **match** commands, and, if more than one **match** command exists in the traffic class, an instruction on how to evaluate these **match** commands. The traffic class is named in the **class-map** command. For example, if you use the word *cisco* with the **class-map** command, the traffic class would be named *cisco*.

The **match** commands are used to specify various criteria for classifying packets. Packets are checked to determine whether they match the criteria specified in the **match** commands. If a packet matches the specified criteria, that packet is considered a member of the class and is forwarded according to the QoS specifications set in the traffic policy. Packets that fail to meet any of the matching criteria are classified as members of the default traffic class. See the "Default Traffic Class" section on page 18.

The instruction on how to evaluate these **match** commands needs to be specified if more than one match criterion exists in the traffic class. The evaluation instruction is specified with the **class-map [match-any]** command. If the **match-any** option is specified as the evaluation instruction, the traffic being evaluated by the traffic class must match at least one of the specified criteria.

The function of these commands is described more thoroughly in the Cisco ASR 9000 Series Aggregation Services Routers Modular Quality of Service Command Reference. The traffic class configuration task is described in the "Creating a Traffic Class" section on page 32.

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.

Traffic Policy Elements

The purpose of a traffic policy is to configure the QoS features that should be associated with the traffic that has been classified in a user-specified traffic class or classes. The **policy-map** command is used to create a traffic policy. A traffic policy contains three elements: a name, a traffic class (specified with the **class** command), and the QoS policies. The name of a traffic policy is specified in the policy map Modular Quality of Service (MQC) (for example, the **policy-map** *policy1* command creates a traffic policy named *policy1*). The traffic class that is used to classify traffic to the specified traffic policy is defined in class map configuration mode. After choosing the traffic class that is used to classify traffic to the traffic policy, the user can enter the QoS features to apply to the classified traffic.

The MQC does not necessarily require that users associate only one traffic class to one traffic policy. When packets match to more than one match criterion, as many as 1024 traffic classes can be associated to a single traffic policy. The 1024 class maps include the default class and the classes of the child policies, if any.

The order in which classes are configured in a policy map is important. The match rules of the classes are programmed into the TCAM in the order in which the classes are specified in a policy map. Therefore, if a packet can possibly match multiple classes, only the first matching class is returned and the corresponding policy is applied.

The function of these commands is described more thoroughly in the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*.

The traffic policy configuration task is described in "Creating a Traffic Policy" section on page 38.

Limitation

Fragmented IPv4 packets are subjected to egress QoS policies only on the main interface and not on sub-interfaces. The fragmented IPv4 packets are subjected to the Local Packet Transport Services (LPTS) policer. IPv4 packets are fragmented when the egress interface MTU is smaller than the packet size.

Default Traffic Class

Unclassified traffic (traffic that does not meet the match criteria specified in the traffic classes) is treated as belonging to the default traffic class.

If the user does not configure a default class, packets are still treated as members of the default class. However, by default, the default class has no enabled features. Therefore, packets belonging to a default class with no configured features have no QoS functionality. These packets are then placed into a first in, first out (FIFO) queue and forwarded at a rate determined by the available underlying link bandwidth. This FIFO queue is managed by a congestion avoidance technique called tail drop.

For further information about congestion avoidance techniques, such as tail drop, see the “Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers” module in this guide

Bundle Traffic Policies

When a policy is bound to a bundle, the same policy is programmed on every bundle member (port). For example, if there is a policer or shaper rate, the same rate is configured on every port. Traffic is scheduled to bundle members based on the load balancing algorithm.

A policy can be bound to:

- Bundles
- Bundle Layer 3 subinterfaces
- Bundle Layer 2 subinterfaces (Layer 2 transport)

Both ingress and egress traffic is supported. Percentage-based policies and absolute rate-based policies are supported. However, for ease of use, it is recommended to use percentage-based policies.

Shared Policy Instance

After the traffic class and traffic policy have been created, Shared Policy Instance (SPI) can optionally be used to allow allocation of a single set of QoS resources and share them across a group of subinterfaces, multiple Ethernet flow points (EFPs), or bundle interfaces.

Using SPI, a single instance of qos policy can be shared across multiple subinterfaces, allowing for aggregate shaping of the subinterfaces to one rate. All of the subinterfaces that share the instance of a QoS policy must belong to the same physical interface. The number of subinterfaces sharing the QoS policy instance can range from 2 to the maximum number of subinterfaces on the port.

For bundle interfaces, hardware resources are replicated per bundle member. All subinterfaces that use a common shared policy instance and are configured on a Link Aggregation Control Protocol (LAG) bundle must be load-balanced to the same member link.

When a policy is configured on a bundle EFP, one instance of the policy is configured on each of the bundle member links. When using SPI across multiple bundle EFPs of the same bundle, one shared instance of the

policy is configured on each of the bundle member links. By default, the bundle load balancing algorithm uses hashing to distribute the traffic (that needs to be sent out of the bundle EFPs) among its bundle members. The traffic for single or multiple EFPs can get distributed among multiple bundle members. If multiple EFPs have traffic that needs to be shaped or policed together usingSPI, the bundle load balancing has to be configured to select the same bundle member (hash-select) for traffic to all the EFPs that belong the same shared instance of the policy. This ensures that traffic going out on all the EFPs with same shared instance of the policy use the same policer/shaper Instance.

This is normally used when the same subscriber has many EFPs, for example, one EFP for each service type, and the provider requires shaping and queuing to be implemented together for all the subscriber EFPs.

Policy Inheritance

When a policy map is applied on a physical port, the policy is enforced for all Layer 2 and Layer 3 subinterfaces under that physical port.

Port Shape Policies

When a port shaping policy is applied to a main interface, individual regular service policies can also be applied on its subinterfaces. Port shaping policy maps have these restrictions:

- class-default is the only allowed class map.
- The shape class action is the only allowed class action.
- They can only be configured in the egress direction.
- They can only be applied to main interfaces, not to subinterfaces.
- Two- and three- level policies are not supported. Only one level or flat policies are supported.

If any of the above restrictions are violated, the configured policy map is applied as a regular policy, not a port shaping policy.

Support for 16 Queues

The ASR 9000 traffic manager (TM) for the enhanced Ethernet line cards now supports up to 16 Queues. The extension is from 8 queues to 16 queues at leaf level called L4 in a QoS policy.

The capabilities of each mode are:

- 8 Q-mode—8 L4 flows per L3 class. Up to 32000 L3 classes in TM.
- 16 Q-mode—16 L4 flows per L3 class. Up to 16000 L3 classes in TM.

This table provides the different service profiles supported in different modes:

Mode	Service Profile
8Q	1 priority-1 queue, 1 priority-2 queue, 6 normal priority queue
16Q	1 priority-1 queue, 1 priority-2 queue, 14 normal priority queue
8Q	1 priority-1 queue, 2 priority-2 queues, 5 normal priority queue (BNG Only)

Mode	Service Profile
16Q	1 priority-1 queue, 2 priority-2 queues, 13 normal priority queue (BNG Only)
8Q	1 priority-1 queue, 1 priority-2 queue, 1 priority- 3 queue, 5 normal priority queue
16Q	1 priority-1 queue, 1 priority-2 queue, 1 priority- 3 queue, 13 normal priority queue

The L3, L4 service profiles in 16 Q-mode are similar to that of the 8 Q-mode, with just an increase in the number of normal priority queues.

Restrictions

The support for 16 queues has these restrictions:

- Supports only the enhanced Ethernet line cards.
- When 16Q-mode policy is applied on all interfaces, the number of interface scale will be 4K interface.

Class-based Unconditional Packet Marking Feature and Benefits

This feature provides users with a means for efficient packet marking by which the users can differentiate packets based on the designated markings.

This feature allows users to perform these tasks:

- Mark packets by setting the IP precedence bits or the IP differentiated services code point (DSCP) in the IP ToS byte.
- Mark Multiprotocol Label Switching (MPLS) packets by setting the EXP bits within the imposed or topmost label.
- Mark packets by setting the Layer 2 class-of-service (CoS) value.
- Mark packets by setting inner and outer CoS tags for an IEEE 802.1Q tunneling (QinQ) configuration.
- Mark packets by setting the value of the *qos-group* argument.
- Mark packets by setting the value of the *discard-class* argument.

Note *qos-group* and *discard-class* are variables internal to the router, and are not transmitted.

Note When the router receives multicast traffic from a Multicast Label Distribution Protocol (MLDP) solution, the MPLS label from the received packet is not dispositioned at the ingress line-card. Instead, the label is removed at the egress line-card. As a result, you cannot mark the IP header for incoming multicast traffic in an MLDP scenario. This means that such packets will not be marked with a Differentiated Services Code Point (DSCP) or precedence value. This is expected behavior for the line cards listed below and is applicable for unconditional marking and for packet marking as policer action (also known as conditional marking):

- ASR 9000 Ethernet Line Cards
- Cisco ASR 9000 High Density 100GE Ethernet line cards
- Cisco ASR 9000 fourth Generation family of QSFP-based dense 100GE line cards.

Unconditional packet marking allows you to partition your network into multiple priority levels or classes of service, as follows:

- Use QoS unconditional packet marking to set the IP precedence or IP DSCP values for packets entering the network. Routers within your network can then use the newly marked IP precedence values to determine how the traffic should be treated.

  For example, weighted random early detection (WRED), a congestion avoidance technique, can be used to determine the probability that a packet is dropped. In addition, low-latency queuing (LLQ) can then be configured to put all packets of that mark into the priority queue.

- Use QoS unconditional packet marking to assign packets to a QoS group. To set the QoS group identifier on MPLS packets, use the **set qos-group** command in policy map class configuration mode.

Note Setting the QoS group identifier does not automatically prioritize the packets for transmission. You must first configure an egress policy that uses the QoS group.

- Use CoS unconditional packet marking to assign packets to set the priority value of IEEE 802.1p/ Inter-Switch Link (ISL) packets. The router uses the CoS value to determine how to prioritize packets for transmission and can use this marking to perform Layer 2-to-Layer 3 mapping. To set the Layer 2 CoS value of an outgoing packet, use the **set cos** command in policy map configuration mode.

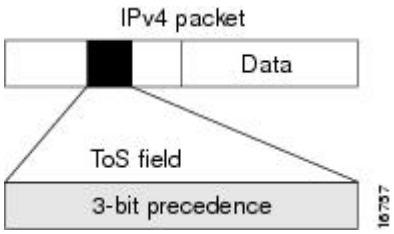
The configuration task is described in the Configuring Class-based Unconditional Packet Marking, page 46.

Note Unless otherwise indicated, the class-based unconditional packet marking for Layer 3 physical interfaces applies to bundle interfaces.

Specification of the CoS for a Packet with IP Precedence

Use of IP precedence allows you to specify the CoS for a packet. You use the three precedence bits in the ToS field of the IP version 4 (IPv4) header for this purpose. This figure shows the ToS field.

Figure 7: IPv4 Packet Type of Service Field



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.

By setting precedence levels on incoming traffic and using them in combination with the Cisco IOS XR QoS queuing features, you can create differentiated service.

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.

The configuration task is described in the "Configuring Class-based Unconditional Packet Marking" section on page 46.

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 2: IP Precedence Values

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

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

IP Precedence Value Settings

By default, Cisco IOS XR software leaves the IP precedence value untouched. This preserves the precedence value set in the header and allows all internal network devices to provide service based on the IP precedence setting. This policy follows the standard approach stipulating that network traffic should be sorted into various types of service at the edge of the network and that those types of service should be implemented in the core of the network. Routers in the core of the network can then use the precedence bits to determine the order of transmission, the likelihood of packet drop, and so on.

Because traffic coming into your network can have the precedence set by outside devices, we recommend that you reset the precedence for all traffic entering your network. By controlling IP precedence settings, you prohibit users that have already set the IP precedence from acquiring better service for their traffic simply by setting a high precedence for all of their packets.

The class-based unconditional packet marking, LLQ, and WRED features can use the IP precedence bits.

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 and egress
- Layer 2 subinterfaces
- Layer 2 main interfaces
- Layer 3 main interfaces

Note The set DEI action is ignored for traffic on interfaces that are not configured for 802.1ad encapsulation.

Default DEI Marking

Incoming Packet		Default DEI on Imposed 802.1ad Headers
802.1q packet	None	0
802.1ad packet	None	DEI of top-most tag of the incoming packet
802.1q packet translated to 802.1ad packet or 802.1ad packet	set dei {0 | 1}	0 or 1 Based on DEI value in the set action

TCP Establishment DSCP Marking/ Set IP Precedence/DSCP for NTP

The Differentiated Services Code Point (DSCP) field in an IP packet which helps enables different levels of service to be assigned to network traffic. Marking is a process, which helps to modify QOS fields incoming and outgoing packets. You can use marking commands in traffic classes, which are referenced in the policy map. You can configure the following marking features:

- DSCP
- IP Precedence
- CoS

Each IP packet is marked with a DSCP code and assigned to corresponding level of service. DSCP is a combination of IP Precedence and Type of Service fields. The TCP Establishment DSCP Marking/Set IP Precedence feature sets Network Time Protocol (NTP) with the DSCP field. NTP packets can be based on either IPv4 and IPV6 based respectively. The NTP sets DSCP/TOS field under either v4 or v6 IP headers. The DSCP level can be configured through the NTP configuration. The configured level will be set across NTP packets throughout IP layer.

IP Precedence Compared to IP DSCP Marking

If you need to mark packets in your network and all your devices support IP DSCP marking, use the IP DSCP marking to mark your packets because the IP DSCP markings provide more unconditional packet marking options. If marking by IP DSCP is undesirable, however, or if you are unsure if the devices in your network support IP DSCP values, use the IP precedence value to mark your packets. The IP precedence value is likely to be supported by all devices in the network.

You can set up to 8 different IP precedence markings and 64 different IP DSCP markings.

Configuring DSCP for source IPv4 address for NTP Packets

To mark a packet by setting the **IP DSCP** value for **NTP** packets, use the following commands (given below) starting in global configuration mode. These commands permit configuring the DSCP for source addresses, to mark **NTP** packets, so that the marked **NTP** packets are treated as per the DSCP markings. There are different code point values available for different services:

SUMMARY STEPS

1. **configure**
2. **ntp** {*ipv4*} **dscp**
3. **end** `or` **commit**
4. **show processes ntpd**
5. **show ntp associations**

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	**configure**
	Example:
	`RP/0/0/CPU0:Router#configure`
	`Mon Aug 10 14:35:04.826 IST`
	`RP/0/0/CPU0:Router(config)#`
		Enters the global configuration mode.

Step 2	**ntp** {*ipv4*} **dscp**
	Example:
	`RP/0/0/CPU0:Router(config)#ntp ipv4 dscp ?`
	`<0-63> Differentiated services codepoint value`
	`af11 Match packets with AF11 dscp (001010)`
	`af12 Match packets with AF12 dscp (001100)`
	`af13 Match packets with AF13 dscp (001110)`
	`af21 Match packets with AF21 dscp (010010)`
	`af22 Match packets with AF22 dscp (010100)`
	`af23 Match packets with AF23 dscp (010110)`
	`af31 Match packets with AF31 dscp (011010)`
	`af32 Match packets with AF32 dscp (011100)`
	`af33 Match packets with AF33 dscp (011110)`
	`af41 Match packets with AF41 dscp (100010)`
	`af42 Match packets with AF42 dscp (100100)`
	`af43 Match packets with AF43 dscp (100110)`
	`cs1 Match packets with CS1(precedence 1) dscp (001000)`
	`cs2 Match packets with CS2(precedence 2) dscp (010000)`
	`cs3 Match packets with CS3(precedence 3) dscp (011000)`
	`cs4 Match packets with CS4(precedence 4) dscp (100000)`
	`cs5 Match packets with CS5(precedence 5) dscp (101000)`
	`cs6 Match packets with CS6(precedence 6) dscp (110000)`
	`cs7 Match packets with CS7(precedence 7) dscp (111000)`
	`default Match packets with default dscp (000000)`
	`ef Match packets with EF dscp (101110)`
		Specifies Differentiated services code point (dscp) value. The range is from 0 to 63. The default value is 0.

Step 3	**end** or **commit** **Example:** `RP/0/RP0/CPU0:Router(config)#commit` **Example:**	Commit- saves the configuration changes and remains within the configuration session. End- prompts the user to take one of these actions: • Yes- Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
	RP/0/0/CPU0:Router#exit (	• No-Exits the configuration session without committing the configuration changes. • Cancel-Remains in the configuration session, without committing the configuration changes.
Step 4	**show processes ntpd** **Example:** RP/0/RP0/CPU0:Router#show processes ntpd Mon Jun 22 20:25:18.026 IST Job Id: 208 PID: 2540 Executable path: /pkg/bin/ntpd Instance #: 1 Version ID: 00.00.0000 Respawn: ON Respawn count: 1 Last started: Fri Jun 19 16:04:14 2015 Process state: Run Package state: Normal Process group: dlrsc core: MAINMEM Max. core: 0 Level: 120 Placement: None startup_path: /pkg/startup/ip_ntp.startup Ready: 2.444s Process cpu time: 0.074s user, 0.031s kernel, 1.005s total PID TID CPU Stack Pri State Run Time CPU use NAME 2540 2978 1 92K 15 Sleeping 3:04:20:59s 0.000s ntpd 2540 2975 4 92K 15 Sleeping 3:04:20:59s 0.001s ntpd 2540 2947 5 92K 15 Sleeping x3:04:20:59s 0.027s chkpt_evm 2540 2943 1 92K 15 Sleeping 3:04:21:00s 0.000s ITAL Server Thr 2540 2914 5 92K 15 Sleeping 3:04:21:00s 0.000s async 2540 2810 3 92K 15 Sleeping 3:04:21:00s 0.000s EnXR internal:mmap_peer_threa 2540 2760 2 92K 15 Sleeping 3:04:21:00s 0.011s ntpd 2540 2540 1 92K 15 Sleeping 3:04:21:03s 0.064s ntpd	Displays the ntpd process
Step 5	**show ntp associations** **Example:**	Shows the status of NTP associations.

	Command or Action	Purpose
	Router# show ntp associations Sat Feb 14 13:53:18.468 UTC address ref clock st when poll reach delay offset disp *~223.255.254.254 171.68.38.65 2 121 128 377 2.44 -0.260 4.537 * sys_peer, # selected, + candidate, - outlayer, x falseticker, ~ configured	

Configure DSCP CS7 (Precedence 7)

Before you begin

The IP DSCP value in the class map command using the following commands, starting with the global configuration mode.

SUMMARY STEPS

1. **configure**
2. **ntp** *ipv4* **dscp** *cs7*
3. **end** `or` **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/0/CPU0:Router#configure Mon Aug 10 14:35:04.826 IST RP/0/0/CPU0:Router(config)#	Enters the global configuration mode.
Step 2	**ntp** *ipv4* **dscp** *cs7*	Configures options in **DSCP** for a particular source address in IPv4 packets.
Step 3	**end** `or` **commit** **Example:** RP/0/RP0/CPU0:Router(config)#commit	Commit Command saves the configuration changes and remains within the configuration session. End Command prompts the user to take one of these actions: • Yes- Saves configuration changes and exits the configuration session. • No-Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• Cancel-Remains in the configuration session, without committing the configuration changes.

Configure IPv4 DSCP Precedence

Before you begin

The following steps help you to configure **DSCP** precedence:

SUMMARY STEPS

1. **configure**
2. **ntp** {*ipv4* } **precedence** *codepoint_value*
3. **ntp** { *ipv4*} **precedence**
4. **end** or **commit**
5. **show ntp status**

DETAILED STEPS

Procedure

	Command or Action	Purpose

Step 1	**configure**
	Example:
	`RP/0/0/CPU0:Router#configure`
	`Mon Aug 10 14:35:04.826 IST`
	`RP/0/0/CPU0:Router(config)#`
		Enters the global configuration mode.

Step 2	**ntp** {*ipv4* } **precedence** *codepoint_value*
	Example:
	`RP/0/0/CPU0:Router(config)#ntp ipv4 precedence ?`
	`<0-7> Precedence value`
	`critical Match packets with critical precedence (5)`
	`flash Match packets with flash precedence (3)`
	`flash-override Match packets with flash override precedence (4)`
	`immediate Match packets with immediate precedence (2)`
	`internet Match packets with internetwork control precedence (6)`
	`network Match packets with network`
		Sets the ntp [IPv4] precedence. It ranges from 0 to 63.

	Command or Action	Purpose

	`control precedence (7)`
	`  priority  Match packets with priority precedence (1)`
	`  routine  Match packets with routine precedence (0)`	

Step 3	**ntp** { *ipv4*} **precedence** **Example:** `RP/0/RP0/CPU0:Router(config)#ntp ipv4 precedence internet`	Sets precedence values.

Step 4	**end** or **commit**
	Example:
	`RP/0/RP0/CPU0:Router(config)#commit`
	Example:
	`RP/0/0/CPU0:Router#exit`
	`(`
		Commit- saves the configuration changes and remains within the configuration session.
		End- prompts the user to take one of these actions:
		• Yes- Saves configuration changes and exits the configuration session.
		• No-Exits the configuration session without committing the configuration changes.
		• Cancel-Remains in the configuration session, without committing the configuration changes.

Step 5	**show ntp status**
	Example:
	`RP/0/RP0/CPU0:Router#show ntp status`
	`Mon Aug 10 14:35:04.826 IST`
	`Clock is synchronized, stratum 3, reference is 223.255.254.254`
	`nominal freq is 1000000000.0000 Hz, actual freq is 30440042.8893 Hz, precision is 2**22`
	`reference time is D889D255.BF525356 (13:55:33.747 UTC Sat Feb 14 2015)`
	`clock offset is -0.413 msec, root delay is 3.569 msec`
	`root dispersion is 20.55 msec, peer dispersion is 4.04 msec`
	`loopfilter state is 'CTRL' (Normal Controlled Loop), drift is 0.0000318515 s/s`
	`system poll interval is 128, last update was 117 sec ago`
		Displays NTP status.

Configure IPv6 DSCP precedence

Before you begin

You can configure **DSCP** precedence:

SUMMARY STEPS

1. **configure**
2. **ntp ,** source { *ipv6* } **dscp** *codepoint_value*
3. **ntp** { *ipv6* } **precedence** *codepoint_value*
4. **end** `or` **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:Router(config)#`	Enters the global configuration mode.

Step 2	**ntp ,** source { *ipv6* } **dscp** *codepoint_value*
	Example:
	`RP/0/0/CPU0:Router(config)#ntp ipv6 dscp ?`
	`<0-63> Differentiated services codepoint value`
	`af11 Match packets with AF11 dscp (001010)`
	`af12 Match packets with AF12 dscp (001100)`
	`af13 Match packets with AF13 dscp (001110)`
	`af21 Match packets with AF21 dscp (010010)`
	`af22 Match packets with AF22 dscp (010100)`
	`af23 Match packets with AF23 dscp (010110)`
	`af31 Match packets with AF31 dscp (011010)`
	`af32 Match packets with AF32 dscp (011100)`
	`af33 Match packets with AF33 dscp (011110)`
	`af41 Match packets with AF41 dscp (100010)`
	`af42 Match packets with AF42 dscp (100100)`
	`af43 Match packets with AF43 dscp (100110)`
	`cs1 Match packets with CS1(precedence 1) dscp (001000)`
	`cs2 Match packets with CS2(precedence 2) dscp (010000)`
	`cs3 Match packets with CS3(precedence 3) dscp (011000)`
	`cs4 Match packets with CS4(precedence 4) dscp (100000)`
	`cs5 Match packets with CS5(precedence 5)`
		Configures options in **DSCP** for a particular source address in IPV6 packets. The value ranges between 0 - 63.

	Command or Action	Purpose

	`dscp (101000)`
	`  cs6      Match packets with CS6(precedence 6)`
	`dscp (110000)`
	`  cs7      Match packets with CS7(precedence 7)`
	`dscp (111000)`
	`  default  Match packets with default dscp (000000)`
	
	`  ef       Match packets with EF dscp (101110)`	

Step 3	**ntp** { *ipv6* } **precedence** *codepoint_value*
	Example:
	`  RP/0/0/CPU0:Router(config)#ntp ipv6 precedence ?`
	`  <0-7>           Precedence value`
	`  critical        Match packets with critical precedence (5)`
	`  flash           Match packets with flash precedence (3)`
	`  flash-override  Match packets with flash override precedence (4)`
	`  immediate       Match packets with immediate precedence (2)`
	`  internet        Match packets with internetwork control precedence (6)`
	`  network         Match packets with network control precedence (7)`
	`  priority        Match packets with priority precedence (1)`
	`  routine         Match packets with routine precedence (0)`
		Sets **ntp** *ipv6* **precedence**

Step 4	**end** `or` **commit**
	Example:
	`RP/0/RP0/CPU0:Router(config)#commit`
	Example:
	`RP/0/0/CPU0:ios#exit`
	`(`
		Commit- saves the configuration changes and remains within the configuration session.
		End- prompts the user to take one of these actions:
		• Yes- Saves configuration changes and exits the configuration session.
		• No-Exits the configuration session without committing the configuration changes.
		• Cancel-Remains in the configuration session, without committing the configuration changes.

QoS Policy Propagation Using Border Gateway Protocol

Packet classification identifies and marks traffic flows that require congestion management or congestion avoidance on a data path. Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) allows you to classify packets by Qos Group ID, based on access lists (ACLs), Border Gateway Protocol (BGP) community lists, BGP autonomous system (AS) paths, Source Prefix address, or Destination Prefix address. After a packet has been classified, you can use other QoS features such as policing and weighted random early detection (WRED) to specify and enforce policies to fit your business model.

QoS Policy Propagation Using BGP (QPPB) allows you to map BGP prefixes and attributes to Cisco Express Forwarding (CEF) parameters that can be used to enforce traffic policing. QPPB allows BGP policy set in one location of the network to be propagated using BGP to other parts of the network, where appropriate QoS policies can be created.

QPPB supports both the IPv4 and IPv6 address-families.

QPPB allows you to classify packets based on:

- Access lists.
- BGP community lists. You can use community lists to create groups of communities to use in a match clause of a route policy. As with access lists, you can create a series of community lists.
- BGP autonomous system paths. You can filter routing updates by specifying an access list on both incoming and outbound updates, based on the BGP autonomous system path.
- Source Prefix address. You can classify a set of prefixes coming from the address of a BGP neighbor(s).
- Destination Prefix address. You can classify a set of BGP prefixes.

Classification can be based on the source or destination address of the traffic. BGP and CEF must be enabled for the QPPB feature to be supported.

QoS on PWHE

QoS on Pseudo-wire Head End (PWHE) enables enhanced L3VPN and L2VPN service on a service-provider-edge router. The available PWHE types are PW-Ether main interfaces, PW-Ether subinterfaces, and PW interworking (IW) interfaces.

Note The PWHE-Ether subinterfaces and PW-IW interfaces are supported from Release 5.1.1 onwards.

Supported Features

Features of QoS on PWHE:

- IPv4 and IPv6 address-families are supported.
- Policy maps on both ingress and egress PWHE. Both ingress and egress support policing, marking, and queuing within hardware limitations.
- Policies at the port for the transit traffic can be applied simultaneously with policies for PWHE interfaces.

- Policy is replicated on all PWHE members. This means the rate specified in the PWHE policy-map is limited to the lowest rate of all the pin down members. For example, if the PWHE interface has both 1G and 10G pin down members, the rate is limited to 1G. if the 10G member has a shaper of 900 mbps, the rate of the PWHE interface policy is limited to 900 mbps.
- Port shaping policy on the member interface will impact the PWHE traffic passing through that port.
- Policy maps can be applied on PW-Ether subinterface.
- PW-Ether subinterfaces inherit policy on its main PW-Ether interface.
- PW-Ether subinterface can have policy configured as shared policy instance (SPI).
- PW-Ether main and subinterface policies may co-exist.
- L2 multicast and flood over PW-Ether interface are supported.
- L3 multicast over PW-Ether interface are supported.
- Independent of line card co-existence mode, percentage based rate at the lowest policy level in PW-Ether main and subinterface policies is supported.

Note In the same policy, the grand parent level is lower than parent level, and parent level is lower than child level.

Limitations for QoS on PWHE

Supported Interfaces

You can configure PWHE in Layer 3 mode on PWHE-Ether main and subinterfaces.

QoS Accounting Scope

QoS accounting, which measures and records the packet length when performing QoS functions such as policing, shaping, and gathering statistics, doesn't include PW headers.

QoS Configuration Rules

- **match** commands are optional when configuring QoS on PWHE. However, you must configure at least one match criterion for a class.
- When using the **match access-group** command to configure the match criteria for a class map on the basis of the specified access control list (ACL), QoS classification based on the packet length or time to live (TTL) field in the IPv4 and IPv6 headers is not supported.

L2 Header Based Classification and Marking Scope

L2 header classification and marking are not supported on L3 PWHE interfaces.

Note The classification and marking applied on PW-Ether main interface are inherited by its subinterfaces without policy.

Bandwidth Distribution

PWHE and non-PWHE traffic on the same pin down member share scheduling resources. It is recommended to configure bandwidth remaining in the parent class-default of PWHE policies to control the distribution of excess bandwidth between PWHE and non-PWHE traffic.

Bandwidth remaining command can be used in the parent default class of PWHE policies allowing user to control the distribution of excess bandwidth between various PWHE interfaces and physical interface.

QoS Accounting

- The packet length when performing QoS functions (policing, shaping, statistics, etc.) will be based on the customer IP packet, customer L2 header and the configured additional overhead.
- QoS statistics will include the customer IP packet, customer L2 header and configured additional overhead.

Note For PW-IW interfaces, the packet length used for QoS accounting does not contain customer L2 header.

- Outer MPLS headers (VC label, transport labels, etc.) and outer L2 header (Layer 2 encap of the underlying physical interface) will not be included in the packet length when performing QoS on the PWHE virtual interface.

Classification and Marking Support

Marking for PW-Ether in ingress and egress direction

- Marking of customer IP header, qos-group and discard-class will be supported.
- Marking of EXP bits for all imposed MPLS labels will be supported for PWHE main interface and PW-Ether subinterfaces.
- EXP for imposed labels can be set in an ingress or an egress policy attached to a PWHE interface.

Note For non-PWHE interfaces, EXP for imposed labels can only be set in an ingress policy. This is an exception made for PWHE interfaces because more labels are imposed on the customer IP packet after processing the egress QoS policy.

- For unconditional markings in ingress direction, the following fields can be marked - DSCP/precedence, EXP for imposed labels, qos-group and discard-class.
- For unconditional markings in egress direction, the following fields can be marked - DSCP/precedence, discard-class and EXP for imposed labels.
- For conditional policer markings in ingress direction, at most two of the following fields can be marked - DSCP/precedence, EXP for imposed labels, qos-group and discard-class.
- For conditional policer markings in egress direction, the following fields can be marked - DSCP/precedence, discard-class and EXP for imposed labels.

L2 header based classification and marking support

The Table-1, Table-2 and Table-3 summarizes the L2 header based classification and marking support on different PWHE interfaces.

Table 3: Supported L2 header based classification and marking for PW-Ether VC type 4 interface

PW-Ether VC type 4		
Classification	**Ingress**	**Egress**
SRC MAC	Yes	No
DEST MAC	Yes	No
DEI	Yes	No
DEI Inner	No	No
COS	Yes	No
COS Inner	No	No
VLAN	Yes	No
VLAN Inner	No	No
Marking		
DEI	No	No
COS	No	No
COS Inner	No	No

Table 4: Supported L2 header based classification and marking for PW-Ether VC type 5 interface

PW-Ether VC type 5		
Classification	**Ingress**	**Egress**
SRC MAC	Yes	No
DEST MAC	Yes	No
DEI	Yes	Yes
DEI Inner	No	No
COS	Yes	Yes
COS Inner	Yes	Yes
VLAN	No	No
VLAN Inner	No	No
Marking		
DEI	No	Yes

COS	No	Yes
COS Inner	No	Yes

Note The classification and marking applied on PW-Ether main interface are inherited by its subinterfaces without policy.

Table 5: Supported L2 header based classification and marking for PW-Ether L3 subinterface VC type 5

PW-Ether L3 subinterface VC type 5		
Classification	**Ingress**	**Egress**
SRC MAC	Yes	No
DEST MAC	Yes	No
DEI	Yes	Yes
DEI Inner	No	No
COS	Yes	Yes
COS Inner	Yes	Yes
VLAN	Yes	Yes
VLAN Inner	Yes	Yes
Marking		
DEI	No	Yes
COS	No	Yes
COS Inner	No	Yes

For PW-Ether L2 subinterface VC type 5, all classification and marking are supported.

L2 classification and marking are not supported for PW-IW interface VC type 11.

Policing and Queuing support

All the policing features supported on normal L3 interfaces will be supported on PWHE main interface and subinterface too.

Queuing

	Ingress and Egress Queues	Ingress and Egress Policers
PWHE interface with no policy map	Each PWHE member has per port default queues. Both the ingress and egress traffic will use the members port default queue.	Not applicable

	Ingress and Egress Queues	Ingress and Egress Policers
PWHE interface with a policy map	Any ingress and egress queues in the policymaps would be replicated on each PWHE member.	Any ingress and egress policer in the policymaps would be replicated per each PWHE member.

Note If PWHE member is a bundle, policy maps will be replicated on bundle members.

Statistics

Show commands of a PWHE virtual interface and PWHE subinterface QoS policy will provide ingress / egress statistics;

- per pin down member.
- per bundle member if the bundle is a pin down member.
- aggregated stats on the whole PWHE interface.
- shared policy instance per pin down member.
- aggregated stats on the whole bundle if the bundle is a pin down member.
- PWHE aggregate shaper stats aggregates all queuing stats of all subinterfaces.

Co-existence of PWHE Main and Subinterface Policies

A line card (LC) can be configured to allow PWHE aggregate shaper policy to co-exist with subinterface policies. This mode is known as co-existence mode. The PWHE aggregate shaper policy will only have a class-default with shape and bandwidth remaining actions. If no PWHE subinterface policy exists, PWHE main interface can have up to 3 level-queuing hierarchical policy.

The co-existence mode with subinterface queuing policies is known as co-existence queuing mode. The co-existence mode with subinterface non-queuing policies is known as co-existence non-queuing mode.

As shown in below examples, PWHE aggregate shaper policy can have:

- only shape action
- only bandwidth remaining action
- shape and bandwidth remaining actions.

Here is the example for PWHE aggregate shaper policy with only shape action:

```
policy-map pwhe-aggregate-shaper
class class-default
shape average 1 gbps
!
end-policy-map
!
end
```

Here is the example for PWHE aggregate shaper policy with only bandwidth remaining action:

```
policy-map pwhe-aggregate-shaper
class class-default
```

```
bandwidth remaining ratio 20
!
end-policy-map
!
end
```

Here is the example for PWHE aggregate shaper policy with shape and bandwidth remaining actions:

```
policy-map pwhe-aggregate-shaper
class class-default
shape average 1 gbps
bandwidth remaining ratio 20
!
end-policy-map
!
end
```

Note It is recommended to configure shape and bandwidth remaining actions for PWHE aggregate shaper policy.

Restrictions

These restrictions apply while configuring co-existence mode:

- If co-existence mode is configured for all LCs in ingress direction then co-existence mode configuration for specified LC in ingress will be rejected. But co-existence configuration for specified LC in egress will be accepted provided there is no co-existence mode configured for all LCs in egress direction.
- If any PWHE main or subinterface has policy configured on a LC, configuring or not configuring co-existence mode will take effect after the LC reloads.
- If no PWHE main or subinterface has policy configured on a LC, configuring or not configuring co-existence mode will take effect immediately on the LC. It is recommended to commit the co-existence mode change before adding QoS policies on the PWHE main or subinterfaces.
- In the co-existence queuing mode, policy applied on PWHE subinterface will have up to 2-levels of queuing. Configuring a 3-level queuing policy on PWHE subinterface will be rejected.
- In the co-existence non-queuing mode, only non-queuing policies on subinterfaces are allowed to co-exist with the PWHE aggregate shaper. If PWHE main interface does not have policy, then subinterface policy can have up to 2-level of queuing.
- When a LC is not in co-existence mode, the PWHE main interface and subinterfaces cannot have policies at the same time. But each can have policy if the other does not.
- The traffic for PWHE main interface and subinterfaces without queuing policy will use the pin down interface default queue. The behavior is consistent whether the LC is in co-existence mode or not.
- In co-existence queuing, non-queuing mode or co-existence disabled (default) mode, applying a non-aggregate shaper policy on PWHE main interface is allowed if subinterface policy does not exists. The non-aggregate shaper policy can have up to 3-levels of queuing. If non-aggregate shaper policy applied on PWHE main interface is a queuing policy, it impacts traffic on the PWHE main interface and subinterfaces because the traffic is moving from the port default queues to the new queues created for the PWHE.
- After PWHE subinterface policies are applied, in-place modification of the PWHE aggregate shaper is also allowed but after the modification the policy should still be a PWHE aggregate shaper.

- PWHE subinterface policy co-existing with PWHE aggregate shaper is allowed to be configured as SPI.

PW-Ether Subinterface Policy

QoS policies can be applied on PW-Ether subinterfaces when there is no policy applied on the main PW-Ether interface.

Restrictions

- When the LC is not in co-existence mode, policies supported on regular subinterface are supported on PW-Ether subinterface too.
- Percentage based rate on the lowest level is supported on policy applied on PW-Ether subinterface.

Note In the same policy, the grand parent level is lower than parent level, and parent level is lower than child level.

- When LC is not in co-existence mode, service-policy on the PW-Ether main interface is rejected if there is a service-policy already applied on any of its PW-Ether subinterfaces .

PW-Ether Subinterface Shared Policy Instance

PW-Ether subinterface supports shared policy instance (SPI). SPI on PW-Ether subinterface functions similarly to SPI on bundle subinterfaces.

Restrictions

- SPI is only supported on PW-Ether subinterfaces. Configuring SPI on PWHE main interface will be rejected.
- When a policy is applied on PW-Ether subinterface with the SPI, a single instance of the same policy is created on each pin down member.
- SPI name is unique across all PW-Ether main interfaces and bundle interfaces.

Scale Information

QoS on PWHE supports:

- 8000 PWHE interface per system.
- 1792 PWHE interface per line card (LC).
- 8 physical or bundle interfaces per generic interface list.
- 4096 sub-interfaces per PW-Ether interface.
- 20,000 total subinterfaces per LC.

Note The scale numbers are supported if configuration is applied properly so that queuing resource is not exhausted.

Policy Instantiation

The various scenarios of QoS on PWHE are discussed here:

- If any member interface has policies applied to them, only non PWHE traffic will be subjected to those policies. An exception to this is a configured port shaper.
- QoS policy applied on the PWHE main interface or PWHE subinterface is instantiated on pin-down member. If the pin-down member is a bundle, then the policy is instantiated on each bundle member .
- The supported policy combinations on PWHE main and subinterfaces for line card (LC) in any mode are:
  - Non-queuing policy on PWHE main interface and no policy on subinterfaces.
  - No policy on PWHE main interface and no policy or non-queuing policy on subinterfaces.
  - No policy on PWHE main interface. 2-level queuing policies on subinterface with or without SPI.
  - 1, 2, or 3-level queuing policy on PWHE main interface. No policies on subinterface.
- The supported policy combinations on PWHE main and subinterfaces for LC not in the co-existence mode are:
  - No policy on PWHE main interface. 3-level queuing policies on subinterfaces with or without SPI.

Note In ingress direction, policies with priority but no queuing actions in the policy-map will use the member port default queues. In egress direction, priority is treated as queuing action so dedicated queue is created for it.

- The supported policy combinations on PWHE main and subinterfaces for LC in the co-existence queuing mode are:
  - PWHE aggregate shaper policy on the PWHE main interface. Non-queuing policies on subinterfaces with or without SPI.
  - PWHE aggregate shaper policy on the PWHE main interface. Up to 2 level queuing policies on subinterfaces with or without SPI.

Note In the ingress direction, the PWHE subinterface polices with priority but no queuing action in the policy-map will use the queues created for the PWHE main interface. In the egress direction, priority is treated as queuing action so dedicated queues will be created for the subinterface. If the PWHE main interface does not have queuing policy, its subinterface with non-queuing policies will use the pindown interface default queues.

- The supported policy combination on PWHE main and subinterfaces for LC in the co-existence non-queuing mode is:
  - PWHE aggregate shaper on the PWHE main interface. Non-queuing policies on subinterfaces.

Note In ingress and egress direction, the PWHE subinterface policies with priority but no queuing action in the policy-map will use the queues created for the PWHE main interface. If the PWHE main interface does not have queuing policy, its subinterface policies with priority but no queuing action will use the pin-down interface default queues.

Note When PWHE interface is created, and no PWHE QoS policy is applied on it, PWHE traffic will pass through the member interface default queues.

PWHE without QoS policy

The following two cases represent the default behavior of the PWHE interfaces:

- PWHE ingress to core facing egress (access to core) - DSCP/ precedence value from customer IP packet is copied to EXP of all imposed labels (VPN and transport)in the core-facing direction.
- PWHE egress (core to access) - DSCP/precedence value from customer IP packet is copied to EXP of all imposed labels (VC and transport) in the access-facing direction.

Configuring QoS on PWHE: Example.

The example shows how to configure QoS on PWHE main interface or subinterfaces. The example configuration can not be applied on PWHE main and subinterfaces at the same time.

```
policy-map pw_child_in
class voip
 priority level 1
  police rate percent 1
  !
!
class video
  police rate percent 10
  !
  priority level 2
!
class data
  police rate percent 70 peak-rate percent 100
   exceed-action transmit
   violate-action drop
  !
!
class class-default
  police rate percent 19 peak-rate percent 100
   exceed-action transmit
   violate-action drop
  !
!
end-policy-map
!
policy-map pw_parent_in
class class-default
  service-policy pw_child_in
  police rate 100 mbps
   child-conform-aware
  !
!
end-policy-map
!

policy-map pw_child_out
class voip
  priority level 1
```

```
  police rate 1 mbps
  !
!
class data
  bandwidth remaining percent 70
  random-detect discard-class 3 40 ms 50 ms
!
class video
  priority level 2
  police rate 10 mbps
  !
!
class class-default
  random-detect discard-class 1 20 ms 30 ms
!
end-policy-map
!
policy-map pw_parent_out
class class-default
  service-policy pw_child_out
  shape average 100 mbps
!
end-policy-map
!

interface pw-ether 1
service-policy input pw_parent_in
service-policy output pw_parent_out
!
```

The example shows how to apply the PWHE aggregate shaper on PWHE main interface and another policy on its subinterface when the LC is in co-existence mode:

Note

- Use the **hw-module qos-mode pwhe-aggregate-shaper sub-interface** { **queuing** | **non-queuing** } { **ingress** | **egress** } command to enable co-existence mode on the LC.
- For the following example to work, the LC must be in co-existence queuing mode.
- When LC is in co-existence mode, apply only PWHE aggregate shaper policy on PWHE main interface.

```
policy-map pwhe-aggregate-shaper
class class-default
shape average 1 gbps
bandwidth remaining ratio 20
!
end-policy-map
!
policy-map pw_parent_out
class class-default
service-policy pw_child_out
shape average 100 mbps
!
end-policy-map
!

interface pw-ether 1
service-policy output pwhe-aggregate-shaper
!
```

```
interface pw-ether 1.1
service-policy output pw_parent_out
```

For other PWHE related information, please refer the *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers*

Port Shaper Policy Support on L2 Fabric ICL Interface

In L2 fabric mode, a port shaper policy can be applied on the inter-chassis link (ICL) sub-interface. The port shaper policy applied on the ICL sub-interface helps to control the traffic going out on all satellite access ports, and efficiently handles the oversubscribed backhaul ethernet virtual circuits (EVC). This port shaper policy applies to all the satellite interfaces hosted under the ICL sub-interface.

Restrictions

- Support to add or remove the port shaper policy is implemented only when the nV satellite configuration is present on the ICL sub-interface.
- Only the port shaper policy can be configured under the L2 Fabric ICL in egress direction.
- Ability to shape all satellite ports under a single satellite in a simple ring mode is not supported.
- When operating in L2 fabric and simple ring mode, only upto 2 levels of QoS policies are supported on the satellite access ports. The user-defined class configuration is not supported at parent level.
- Pseudowire Headend (PW-HE) and Broadband Network Gateway (BNG) configurations are not supported under the satellite interfaces in L2 fabric or simple ring mode.

Configuring Port Shaper Policy on the ICL Interface in L2 Fabric Mode

Perform this task to apply the port shaper policy on the L2 fabric inter-chassis link (ICL) ethernet virtual circuits (EVC). This procedure applies the port shaper in the egress direction of the ICL EVC.

Before you begin

The nV satellite configuration must be applied on the inter-chassis link (ICL) interface.

SUMMARY STEPS

1. **configure**
2. **policy-map** [**type qos**] *policy-name*
3. **class** *class-name*
4. **shape** {*shape* [*units*] | **average** *value*}
5. **exit**
6. **end-policy-map**
7. **interface** *type interface-path-id*
8. **service-policy output** *policy-map*
9. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map icl_ps`	Creates or modifies a policy map that can be attached to one or more ICL interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change.
Step 4	**shape** {*shape* [*units*] | **average** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average 400 mbps`	Specifies the port shape allocated for a class belonging to a policy map.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 6	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 7	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface TenGigabitEthernet 0/1/0/0.1`	Configures an interface and enters the sub-interface configuration mode.
Step 8	**service-policy output** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# service-policy output icl_ps`	Attaches a policy map to an output interface to be used as the service policy for the ICL interface.

	Command or Action	Purpose
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Functionality Differences Between ASR 9000 Series High Density Ethernet Line Card Generations

The table lists some fundamental differences in functionalities between the third, fourth, and fifth generations of the ASR 9000 Series High Density Ethernet line cards. See the data sheets for more information on these line cards.

Functionality	Third Generation of ASR 9000 Series High Density Ethernet Line Cards	Fourth and Fifth Generations of ASR 9000 Series High Density Ethernet Line Cards
Ingress queuing	Supported. See **Ingress Queuing support**for details.	Not supported
Pre-configured profile values	Supported	Not supported
Default queue limit	100 ms	20 ms
Queue length measurement unit	packets or particles	KB
Queue length for Weighted Random Early Detection (WRED)	Instantaneous queue length. Run the **show policy-map interface** command to view the counter readings.	Average queue length. Run the **show policy-map interface** command to view the counter readings.

Functionality	Third Generation of ASR 9000 Series High Density Ethernet Line Cards	Fourth and Fifth Generations of ASR 9000 Series High Density Ethernet Line Cards
Shaper granularity	8 kbps for all scheduler hierarchies	• 100 kbps for scheduler hierachies L2—L4 • 400 kbps for scheduler hierachy L1 In certain use cases, traffic shaper rates may be rounded down. For example, in a two or three-level Hierarchical QOS policy with only a shaper action in the parent or grandparent policy applied to the main interface, if the configured shaper rate is • Non-multiples of 4—shaper rate is rounded down to the nearest multiple of 4 due to L1 shaper granularity of 400 kbps. • Multiples of 4—configured shaper rate is honored. For values in non-multiples of 4, to ensure the configured shaper rate is honored, apply the policy to L2 or L3 sub-interfaces, or add a marking action to the parent or grandparent policy when applying it to the main interface.
Policer granularity	8 kbps	1 kbps
Default burst size for policer	100 ms	100 ms
Default burst size for shaper	100 ms	1 ms
Particle granularity	256 bytes	32 bytes

Functionality	Third Generation of ASR 9000 Series High Density Ethernet Line Cards	Fourth and Fifth Generations of ASR 9000 Series High Density Ethernet Line Cards
Combination for priority and normal queues	Support for two modes: • 3 priority queues and 5 normal queues • 2 priority queues and 6 normal queues	2 priority queues and 6 normal queues. No support for priority level 3.
QoS classification format ID 5	Supported	Not supported
Traffic drop	Packet drops on account of a full egress queue, or drops by WRED are performed in the Traffic Manager after the packet has exited the NP pipeline. The following commands account for these drops: • **show controllers np tm counters** • **show policy-map interface**	Because of better integration of the NP pipeline with the Traffic Manager, the egress queue state is verified while the NP pipeline is still processing the packet. The following commands account for drops because of a full egress queue or drops by WRED: • **show controllers np tm counters** • **show policy-map interface** • **show controllers np counters**
Multi-rate interfaces	Supported	Not supported

Prioritize BFD traffic over logical bundle with QoS on third and fifth generation of ASR 9000 Series high density ethernet line cards

Bidirectional forwarding detection over logical bundle (BLB) is a BFD implementation that:

- is implemented over VLAN interfaces
- uses a single BFD session for the entire bundle interface, monitoring the path as a whole, and
- monitors the health of a bundle interface as a single entity, rather than monitoring each physical link individually.

Automatic line card assignment for BFD sessions

Users cannot decide which line card should handle a BFD session on an ASR 9000 router with a mix of third and fifth generation high density ethernet line cards within a logical bundle. The router automatically assigns the session to one of these physical links in the bundle based on the availability of the line cards.

This table provides information about how to handle BFD traffic on an ASR 9000 router with a mix of third and fifth generation high density ethernet line cards.

Table 6: Handling of BFD Traffic

If...	Then...
the router has ASR 9000 third generation high density ethernet line cards in a logical bundle with QoS and there is congestion at the egress interface,	the BFD packets are dropped.
the router has ASR 9000 fifth generation high density ethernet line cards in a logical bundle with QoS,	the line cards bypass the QoS processing for BFD traffic.
the BFD session moves from a third generation to a fifth generation high density ethernet line card within the bundle due to a line card failure or system restart,	any previously set CoS or precedence values are bypassed, and QoS does not apply anymore.
the BFD session moves from a fifth generation to a third generation high density ethernet line card,	the CoS or precedence values are applied and the packets are dropped if there is congestion.

Apply QoS on the egress interface to prevent BFD traffic drops

Configure QoS at the egress interface to ensure that the BFD packets are prioritized, regardless of whether they originate from a third or a fifth generation high density ethernet line card in the bundle.

Ingress Queuing Support

Ingress queuing is disabled for some line cards.

The tables below list out the ingress queuing support for fixed port and modular line cards.

Note Ingress queuing is not supported on ASR9K-SIP-700 line cards.

Fixed port Line Card

LC type	Ingress Queuing Support
A9K-24X10GE-TR/- SE	Yes
A9K-8X100G-LB-SE / -TR	No
A9K-8X100GE-SE / -TR	No
A99-8X100GE-SE / -TR	No
A9K-4X100GE-SE / -TR	Yes
A99-12X100GE	No
A9K-4X100GE	No
A9K-48X10GE-1G-SE/-TR	No
A9K-24X10GE-1G-SE/-TR	No

LC type	Ingress Queuing Support
A99-48X10GE-1G-SE/-TR	No

Modular Line Card

Note The A9K-MOD400-SE/TR line cards are supported from Cisco IOS XR Release 5.3.2, and the A9K-MOD200-SE/TR line cards are supported from Cisco IOS XR Release 6.0.1.

For minimum software release versions of the new MPAs that are supported on the Cisco ASR 9000 Series 400G (A9K-MOD400-SE/TR) and 200G Modular Line Cards (A9K-MOD200-SE/TR), see Table 5 and Table 6 respectively.

LC type	EP type	Ingress Queuing Support
A9K-MOD80-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD80-TR/ -SE	A9K-MPA-4X10GE	No **Note** To enable ingress queueing on these line cards, run the command **hw-module all qos-mode ingress-queue-enable**.
A9K-MOD80-TR/ -SE	A9K-MPA-2X10GE	Yes
A9K-MOD80-TR/ -SE	A9K-MPA-1X40GE	No
A9K-MOD400-SE	A9K-MPA-8X10GE	Yes
A9K-MOD400-SE	A9K-MPA-20X10GE	No **Note** To enable ingress queueing on these line cards, run the command **hw-module all qos-mode ingress-queue-enable**.
A9K-MOD400-SE	A9K-MPA-2X100GE	No
A9K-MOD400-SE	A9K-MPA-1X100GE	Yes
A9K-MOD400-SE	A9K-MPA-2X100GE	Yes
A9K-MOD400-SE	A9K-MPA-32X1GE	Yes

LC type	EP type	Ingress Queuing Support
A9K-MOD200-SE/ -TR	A9K-MPA-8X10GE	No **Note** To enable ingress queueing on these line cards, run the command **hw-module all qos-mode ingress-queue-enable**.
A9K-MOD200-SE	A9K-MPA-4X10GE	Yes
A9K-MOD200-SE	A9K-MPA-2X10GE	Yes
A9K-MOD200-SE	A9K-MPA-2X40GE	No
A9K-MOD200-SE	A9K-MPA-1X40GE	Yes
A9K-MOD200-SE	A9K-MPA-20X1GE	Yes
A9K-MOD200-SE	A9K-MPA-32X1GE	Yes
A9K-MOD200-SE	A9K-MPA-1X100GE	No
A9K-MOD200-SE	A9K-MPA-10X10GE	No
A9K-24X10GE-1G	4X1GE , 4X10GE	No
A9K-48X10GE-1G	4X1GE , 4X10GE	No
A99-12X100GE/A9K-4X100GE	QSFP-4X10G	No
A99-12X100GE/A9K-4X100GE	QSFP-1X40G	No
A99-12X100GE/A9K-4X100GE	QSFP-1x100G	No
ASR9901	All types	No
A9K-MOD160-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD160-TR/ -SE	A9K-MPA-4X10GE	Yes
A9K-MOD160-TR/ -SE	A9K-MPA-2X10GE	Yes
A9K-MOD160-TR/ -SE	A9K-MPA-1X40GE	No
A9K-MOD160-TR/ -SE	A9K-MPA-2X40GE	No
A9K-MOD160-TR/ -SE	A9K-MPA-8X10GE	No
A9K-MOD200-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD200-TR/ -SE	A9K-MPA-4X10GE	Yes
A9K-MOD200-TR/ -SE	A9K-MPA-2X10GE	Yes

LC type	EP type	Ingress Queuing Support
A9K-MOD200-TR/ -SE	A9K-MPA-1X40GE	Yes
A9K-MOD200-TR/ -SE	A9K-MPA-2X40GE	No
A9K-MOD200-TR/ -SE	A9K-MPA-8X10GE	No
A9K-MOD400-TR/ -SE	A9K-MPA-20X1GE	Yes
A9K-MOD400-TR/ -SE	A9K-MPA-4X10GE	Yes
A9K-MOD400-TR/ -SE	A9K-MPA-2X10GE	Yes
A9K-MOD400-TR/ -SE	A9K-MPA-1X40GE	Yes
A9K-MOD400-TR/ -SE	A9K-MPA-2X40GE	No
A9K-MOD400-TR	A9K-MPA-8X10GE	Yes
A9K-MOD400-TR	A9K-MPA-20X10GE	No
A9K-MOD400-TR	A9K-MPA-2X100GE	No
A9K-MOD400-TR	A9K-MPA-1X100GE	Yes

In-Place Policy Modification

The In-Place Policy Modification feature allows you to modify a QoS policy even when the QoS policy is attached to one or more interfaces. When you modify the QoS policy attached to one or more interfaces, the QoS policy is automatically modified on all the interfaces to which the QoS policy is attached. A modified policy is subject to the same checks that a new policy is subject to when it is bound to an interface.

If the policy-modification is successful, the modified policy takes effect on all the interfaces to which the policy is attached. The configuration session is blocked until the policy modification is complete.

However, if the policy modification fails on any one of the interfaces, an automatic rollback is initiated to ensure that the pre-modification policy is in effect on all the interfaces. The configuration session is blocked until the rollback is complete on all affected interfaces.

If unrecoverable errors occur during in-place policy modification, the policy is put into an inconsistent state on target interfaces. Use the **show qos inconsistency** command to view inconsistency in each location. (This command is supported only on ASR 9000 Ethernet Line Cards). The configuration session is blocked until the modified policy is effective on all interfaces that are using the policy. No new configuration is possible until the configuration session is unblocked.

When a QoS policy attached to an interface is modified, there might not be any policy in effect on the interfaces in which the modified policy is used for a short period of time.

Note The QoS statistics for the policy that is attached to an interface are lost (reset to 0) when the policy is modified.

Recommendations for Using In-Place Policy Modification

For a short period of time while a QoS policy is being modified, there might not be any policy in effect on the interfaces in which the modified policy is used. For this reason, modify QoS policies that affect the fewest number of interfaces at a time. Use the **show policy-map targets** command to identify the number of interfaces that will be affected during policy map modification.

Dynamic Modification of Interface Bandwidth

This section describes the dynamic modification of interface bandwidth feature.

Policy States

- Verification—This state indicates an incompatibility of the configured QoS policy with respect to the new interface bandwidth value. The system handles traffic on a best-efforts basis and some traffic drops can occur.

Inter-Class Policer Bucket Sharing

Based on different classification criteria, inter-class policer bucket sharing feature allows policer bucket sharing among different classes in a hierarchical QoS model, within the modular quality of service command line (MQC) construct, to achieve multirate policing of the same packet. In this feature, the classification of the incoming packet happens only once. However, the policer bucket is shared among classes; that is the same token bucket is used even though a match happens against different classes.

This feature includes following components:

Policer Bucket Shared

The policer bucket shared feature defines and shares a policer node entity. The defined policer bucket is shared among multiple classes.

Here is a sample configuration that defines and shares policer bucket instance *sp1*:

```
policy-map parent
        class long-distance
          police bucket shared sp1 rate 1 mbps
```

In this configuration, a policy-map for class long-distance traffic type is created to police at 1Mbps and the policer bucket is shared.

Policer Bucket Referred

The policer bucket referred feature refers a defined policer bucket instance. The reference to the policer bucket could be across policy level, a parent can refer a child policer, or vice versa, and one policer node can be referred by multiple classes across a policy map.

Here is a sample configuration that refers shared policer bucket instance *sp1*:

```
policy-map voip-child
        class long-distance-voip
         police bucket referred sp1
```

In this configuration, a policy-map for class long-distance-voip traffic type is created and the shared policer bucket *sp1* is referred.

Interface Support

Inter-class policer bucket sharing feature is supported in both the egress and ingress directions. This section describes supported and non-supported interfaces for inter-class policer bucket sharing feature.

Table 7: Supported and non-supported interfaces

Supported Interfaces	1G/10G/100GE Physical interfaces
	L2 and L3 sub-interfaces
	Bundle ports
	Bundle sub-interfaces
Non-supported Interfaces	Bridge Virtual Interface (BVI)
	Satellite interfaces
	Pseudowire Headend (PWHE) interfaces

Note Inter-class policer bucket sharing feature is supported only on the ASR 9000 Enhanced Ethernet Line Card.

Classification Support for Ethernet-Services ACL

You can configure class of service (QoS) classification based on a match for partial MAC address (such as Organizationally Unique Identifier (OUI)) using the **match access-group ethernet-service** command. This command creates a match criteria for a class map based on the specified ethernet-service access control list (ACL) containing MAC addresses.

For example, you can create an ethernet-service ACL such as the following:

```
ethernet-services access-list acl1
 20 permit 2222.3300.0000 0000.00ff.ffff any
 30 permit 1111.2200.0000 0000.00ff.ffff any
 40 permit 1212.2300.0000 0000.00ff.ffff any
!
```

The ethernet-service ACL can be used in the class map to match the MAC addresses as follows:

```
class-map NID-123
  match access-group ethernet-service acl1
end-class-map
```

Note

- You can provide multiple values for the **ethernet-service** match type in a configuration; only the first value is considered for the match criteria. Subsequent values indicated in the match statement are ignored for classification.
- The capture statements in an ethernet-service ACL are ignored.
- An ethernet-service ACL should have only permit statements. If there are any deny statements, the policy is rejected.
- If you specify a value for the **Ether-Type** keyword using the **match access-group ethernet-service** command, the value is ignored.

How to Configure Modular QoS Packet Classification

Creating a Traffic Class

To create a traffic class 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.

For conceptual information, see the Traffic Class Elements, page 16.

Restrictions

- All **match** commands specified in this configuration task are considered optional, but you must configure at least one match criterion for a class.
- For the **match access-group** command, QoS classification based on the packet length or TTL (time to live) field in the IPv4 and IPv6 headers is not supported.
- For the **match access-group** command, when an ACL list is used within a class-map, the deny action of the ACL is ignored and the traffic is classified based on the specified ACL match parameters.

  An empty ACL (contains no rules, only remarks) within a QoS policy-map permits all traffic by default, and the implicit deny condition doesn't work with an empty ACL. Within a QoS policy map, the corresponding class-map matches all traffic not yet matched by the preceding traffic classes. In such a case, any explicit deny rule in the ACL leads to configuration commit failure.
- The **match discard-class** command is not supported on the Asynchronous Transfer Mode (ATM) interfaces.
- When QoS policy-maps use ACLs to classify traffic, ACEs of ACLs consume some amount of TCAM memory of the line card. Each QoS policy-map for ASR9000 supports up to a maximum of 3072 TCAM IPv4 entries. If you cross the limit, IOS XR fails to apply this policy-map with the insufficient memory available error. If you encounter this error, decrease the number of ACEs in ACLs for the policy-map.

This error typically appears when using nested policy-maps, where ACEs in ACLs on different levels are multiplied.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match [not] access-group** [**ipv4| ipv6| ethernet-service**] *access-group-name*
4. **match** [**not**] **cos** [*cos-value*] [*cos-value0 ... cos-value7*]
5. **match** [**not**] **cos inner** [*inner-cos-value*] [*inner-cos-value0...inner-cos-value7*]
6. **match destination-address mac** *destination-mac-address*
7. **match source-address mac** *source-mac-address*
8. **match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1 ... discard-class-value6*]
9. **match** [**not**] **dscp** [**ipv4** | **ipv6**] *dscp-value* [*dscp-value ... dscp-value*]
10. **match** [**not**] **mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*]
11. **match** [**not**] **precedence** [**ipv4** | **ipv6**] *precedence-value* [*precedence-value1 ... precedence-value6*]
12. **match** [**not**] **protocol** *protocol-value* [*protocol-value1 ... protocol-value7*]
13. **match** [**not**] **qos-group** [*qos-group-value1 ... qos-group-value8*]
14. **match vlan [inner]** *vlanid* [vlanid1 ... vlanid7]
15. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(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 [not] access-group** [**ipv4| ipv6| ethernet-service**] *access-group-name* **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match access-group ipv4 map1`	(Optional) Configures the match criteria for a class map based on the specified access control list (ACL) name. **Note** You can provide multiple values in a configuration; only the first value is considered for the match criteria. The subsequent values indicated in the match statement are ignored for classification.

	Command or Action	Purpose
Step 4	**match** [**not**] **cos** [*cos-value*] [*cos-value0 ... cos-value7*] **Example:** `RP/0/RSP0/CPU0:router(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 5	**match** [**not**] **cos inner** [*inner-cos-value*] [*inner-cos-value0...inner-cos-value7*] **Example:** `RP/0/RSP0/CPU0:router match cos inner 7`	(Optional) Specifies an *inner-cos-value* in a class map to match packets. The inner-cos-value arguments are specified as an integer from 0 to 7.
Step 6	**match destination-address mac** *destination-mac-address* **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match destination-address mac 00.00.00`	(Optional) Configures the match criteria for a class map based on the specified destination MAC address.
Step 7	**match source-address mac** *source-mac-address* **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match source-address mac 00.00.00`	(Optional) Configures the match criteria for a class map based on the specified source MAC address.
Step 8	**match** [**not**] **discard-class** *discard-class-value* [*discard-class-value1 ... discard-class-value6*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match discard-class 5`	(Optional) Specifies a *discard-class-value* in a class map to match packets. The *discard-class-value* argument is specified as an integer from 0 to 7. The **match discard-class** command is supported only for an egress policy. The **match discard-class** command is not supported on the Asynchronous Transfer Mode (ATM) interfaces.
Step 9	**match** [**not**] **dscp** [**ipv4** | **ipv6**] *dscp-value* [*dscp-value ... dscp-value*] **Example:** `RP/0/RSP0/CPU0:router(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 10	**match** [**not**] **mpls experimental topmost** *exp-value* [*exp-value1 ... exp-value7*] **Example:** `RP/0/RSP0/CPU0:router(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.

	Command or Action	Purpose
Step 11	**match** [**not**] **precedence** [**ipv4** | **ipv6**] *precedence-value* [*precedence-value1 ... precedence-value6*] **Example:** `RP/0/RSP0/CPU0:router(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 12	**match** [**not**] **protocol** *protocol-value* [*protocol-value1 ... protocol-value7*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match protocol igmp`	(Optional) Configures the match criteria for a class map on the basis of the specified protocol.
Step 13	**match** [**not**] **qos-group** [*qos-group-value1 ... qos-group-value8*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match qos-group 1 2 3 4 5 6 7 8`	(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 63. • Up to eight values (separated by spaces) can be entered in one match statement. • **match qos-group** command is supported only for an egress policy.
Step 14	**match vlan [inner]** *vlanid* [vlanid1 ... vlanid7] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match vlan vlanid vlanid1`	(Optional) Specifies a VLAN ID or range of VLAN IDs in a class map to match packets. • *vlanid* is specified as an exact value or range of values from 1 to 4094. • Total number of supported VLAN values or ranges is 8.
Step 15	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Creating a Traffic Policy

To create a traffic policy, use the **policy-map** command to specify the traffic policy name.

The traffic class is associated with the traffic policy when the **class** command is used. The **class** command must be issued after you enter the policy map configuration mode. After entering the **class** command, the router is automatically in policy map class configuration mode, which is where the QoS policies for the traffic policy are defined.

These class-actions are supported:

- bandwidth—Configures the bandwidth for the class. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.
- police—Police traffic. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.
- priority—Assigns priority to the class. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.
- queue-limit—Configures queue-limit (tail drop threshold) for the class. See the "Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers" module in this guide.
- random-detect—Enables Random Early Detection. See the "Configuring Modular QoS Congestion Avoidance on Cisco ASR 9000 Series Routers" module in this guide.
- service-policy—Configures a child service policy.
- set—Configures marking for this class. See the "Class-based Unconditional Packet Marking Feature and Benefits" section on page 20.
- shape—Configures shaping for the class. See the "Configuring Modular Quality of Service Congestion Management on Cisco ASR 9000 Series Routers" module in this guide.

For additional commands that can be entered as match criteria, see the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*.

For conceptual information, see "Traffic Policy Elements" section on page 17.

SUMMARY STEPS

1. **configure**
2. **policy-map** [**type qos**] *policy-name*
3. **class** class-name
4. **set precedence** [**tunnel**] *precedence-value*
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** class-name **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 4	**set precedence** [**tunnel**] *precedence-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set precedence 3`	Sets the precedence value in the IP header.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Attaching a Traffic Policy to an Interface

After the traffic class and traffic policy are created, you must use the service-policy interface configuration command to attach a traffic policy to an interface, and to specify the direction in which the policy should be applied (either on packets coming into the interface or packets leaving the interface).

For additional commands that can be entered in policy map class configuration mode, see the Cisco ASR 9000 Series Aggregation Services RoutersModular Quality of Service Command Reference..

Prerequisites

A traffic class and traffic policy must be created before attaching a traffic policy to an interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **service-policy** {**input** | **output**} *policy-map*
4. Use the **commit** or **end** command.
5. **show policy-map interface** *type interface-path-id* [**input** | **output**]

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/1/0/9`	Configures an interface and enters the interface configuration mode.
Step 3	**service-policy** {**input**	**output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show policy-map interface** *type interface-path-id* [**input**	**output**] **Example:**	(Optional) Displays statistics for the policy on the specified interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# show policy-map interface gigabitethernet 0/1/0/9`	

Attaching a Shared Policy Instance to Multiple Subinterfaces

After the traffic class and traffic policy are created, you can optionally use the **service-policy (interface)** configuration command to attach a shared policy instance to multiple subinterfaces, and to specify the direction in which the policy should be applied (either on packets coming into or leaving the subinterface).

Note A shared policy can include a combination of Layer 2 and Layer 3 subinterfaces.

For additional commands that can be entered in policy map class configuration mode, see the Cisco ASR 9000 Series Aggregation Services Routers Modular Quality of Service Command Reference.

Prerequisites

A traffic class and traffic policy must be created before attaching a shared policy instance to a subinterface.

Restrictions

Shared policy instance across multiple physical interfaces is not supported.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **service-policy** {**input** | **output**} *policy-map* [**shared-policy-instance** instance-name]
4. Use the **commit** or **end** command.
5. **show policy-map shared-policy-instance** *instance-name* [**input** | **output**] **location** *rack/slot/module*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/1/0/0.1`	Enters interface configuration mode and configures a subinterface.

	Command or Action	Purpose
Step 3	**service-policy** {**input** | **output**} *policy-map* [**shared-policy-instance** instance-name] **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1 shared-policy-instance Customer1`	Attaches a policy map to an input or output subinterface to be used as the service policy for that subinterface. • In this example, the traffic policy evaluates all traffic leaving that interface.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show policy-map shared-policy-instance** *instance-name* [**input** | **output**] **location** *rack/slot/module* **Example:** `RP/0/RSP0/CPU0:router# show policy-map shared-policy-instance Customer1 location 0/1/0/7.1`	(Optional) Displays statistics for the policy on the specified shared policy instance subinterface.

Attaching a Shared Policy Instance to Bundle Interfaces or EFP Bundles

After the traffic class and traffic policy are created, you can optionally use the **service-policy (interface)** configuration command to attach a shared policy instance to bundle interfaces and to bundle EFPs, and to specify the direction in which the policy should be applied (either on packets coming into or leaving the subinterface).

For additional commands that can be entered in policy map class configuration mode, see the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Command Reference*.

Prerequisites

A traffic class and traffic policy must be created before attaching a shared policy instance to bundle interfaces or EFP bundles.

Restrictions

Shared policy instance across multiple physical interfaces is not supported.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **service-policy** {**input** | **output**} *policy-map* [**shared-policy-instance** instance-name]
4. Use the **commit** or **end** command.

5. **show policy-map shared-policy-instance** *instance-name* [**input** | **output**] **location** *location-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface Bundle-Ether** *bundle-id* **Example:** `RP/0/RP1/CPU0:router(config)# interface Bundle-Ether 100.1 l2transport`	Enters interface configuration mode and configures a bundle interface.
Step 3	**service-policy** {**input** | **output**} *policy-map* [**shared-policy-instance** instance-name] **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1 shared-policy-instance Customer1`	Attaches a policy map to an input or output bundle interface to be used as the service policy for that subinterface. • In this example, the traffic policy evaluates all traffic leaving that interface.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show policy-map shared-policy-instance** *instance-name* [**input** | **output**] **location** *location-id* **Example:** `RP/0/RSP0/CPU0:router# show policy-map shared-policy-instance Customer1 location 0/rsp0/cpu0`	(Optional) Displays statistics for the policy at the specified shared policy instance location.

Configuring Class-based Unconditional Packet Marking

This configuration task explains how to configure the following class-based unconditional packet marking features on your router:

- IP precedence value
- IP DSCP value
- QoS group value (ingress only)
- CoS value (egress only on Layer 3 subinterfaces)
- MPLS experimental value
- Discard class

Note IPv4 and IPv6 QoS actions applied to MPLS tagged packets are not supported. The configuration is accepted, but no action is taken.

Note Choose only two **set** commands per class.

Procedure

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **policy-map** *policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# policy-map policy1
```

Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

Step 3 **class** *class-name*

Example:

```
RP/0/RSP0/CPU0:router(config-pmap)# class class1
```

Specifies the name of the class whose policy you want to create or change and enters the policy class map configuration mode.

Step 4 **set precedence**

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set precedence 1
```

Sets the precedence value in the IP header.

Step 5 **set dscp**

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set dscp 5
```

Marks a packet by setting the DSCP in the ToS byte.

Step 6 **set qos-group** *qos-group-value*

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set qos-group 31
```

Sets the QoS group identifiers on IPv4 or MPLS packets.

The **set qos-group** command is supported only on an ingress policy.

Step 7 **set cos** *cos-value*

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set cos 7
```

Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7.

Sets the Layer 2 CoS value of an outgoing packet.

- This command should be used by a router if a user wants to mark a packet that is being sent to a switch. Switches can leverage Layer 2 header information, including a CoS value marking.
- Packets entering an interface cannot be set with a CoS value.

Step 8 **set cos** [**inner**] *cos-value*

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set cos 7
```

Sets the specific IEEE 802.1Q Layer 2 CoS value of an outgoing packet. Values are from 0 to7.

Sets the Layer 2 CoS value of an outgoing packet.

- This command should be used by a router if a user wants to mark a packet that is being sent to a switch. Switches can leverage Layer 2 header information, including a CoS value marking.
- For Layer 2 interfaces, the set cos command:

  Is rejected on ingress or egress policies on a main interface.

  Is accepted but ignored on ingress policies on a subinterface.

  Is supported on egress policies on a subinterface.

- For Layer 3 interfaces, the set cos command:

  Is ignored on ingress policies on a main interface.

  Is rejected on ingress policies on a subinterface.

  Is supported on egress policies on main interfaces and subinterfaces.

Step 9 **set mpls experimental** {**imposition** | **topmost**} *exp-value*

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set mpls experimental imposition 3
```

Sets the experimental value of the MPLS packet top-most or imposition labels.

Note
The **imposition** keyword can be used only in service policies that are attached in the ingress policy.

Step 10 **set srp-priority** *priority-value*

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set srp-priority 3
```

Sets the spatial reuse protocol (SRP) priority value of an outgoing packet.

Note
This command can be used only in service policies that are attached in the output direction of an interface.

Step 11 **set discard-class** *discard-class-value*

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set discard-class 3
```

Sets the discard class on IP Version 4 (IPv4) or Multiprotocol Label Switching (MPLS) packets.

Note
This command can be used only in service policies that are attached in the ingress policy.

Step 12 **set atm-clp**

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# set atm-clp
```

Sets the cell loss priority (CLP) bit.

Step 13 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-pmap-c)# exit
```

Returns the router to policy map configuration mode.

Step 14 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-pmap)# exit
```

Returns the router to global configuration mode.

Step 15 **interface** *type* interface-path-id

Example:

```
RP/0/RSP0/CPU0:router(config)# interface pos 0/2/0/0
```

Configures an interface and enters the interface configuration mode.

Step 16 **service-policy** {**input** | **output**]} *policy-map*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1
```

Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.

Step 17 Use the **commit** or **end** command.

commit —Saves the configuration changes, and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Step 18 **show policy-map interface** *type interface-path-id* [**input** | **output**]

Example:

```
RP/0/RSP0/CPU0:router# show policy-map interface pos 0/2/0/0
```

(Optional) Displays policy configuration information for all classes configured for all service policies on the specified interface.

Configuring QoS Policy Propagation Using Border Gateway Protocol

This section explains how to configure Policy Propagation Using Border Gateway Protocol (BGP) on a router based on BGP community lists, BGP autonomous system paths, access lists, source prefix address, or destination prefix address.

Policy Propagation Using BGP Configuration Task List

Policy propagation using BGP allows you to classify packets by IP precedence and/or QoS group ID, based on BGP community lists, BGP autonomous system paths, access lists, source prefix address and destination prefix address. After a packet has been classified, you can use other quality-of-service features such as weighted random early detection (WRED) to specify and enforce policies to fit your business model.

Overview of Tasks

To configure Policy Propagation Using BGP, perform these basic tasks:

- Configure BGP and Cisco Express Forwarding (CEF). To configure BGP, see Cisco IOS XR Routing Configuration Guide . To configure CEF, see Cisco IOS XR IP Address and Services Configuration Guide .
- Configure a BGP community list or access list.
- Define the route policy. Set the IP precedence and/or QoS group ID, based on the BGP community list, BGP autonomous system path, access list, source prefix address or destination prefix address.
- Apply the route policy to BGP.
- Configure QPPB on the desired interfaces or configure QPPB on the GRE Tunnel interfaces.
- Configure and enable a QoS Policy to use the above classification (IP precedence or QoS group ID). To configure committed access rate (CAR), WRED and tail drop, see the Configuring Modular QoS Congestion Avoidance on Cisco IOS XR Software module.

Defining the Route Policy

This task defines the route policy used to classify BGP prefixes with IP precedence or QoS group ID.

Prerequisites

Configure the BGP community list, or access list, for use in the route policy.

Restrictions

- IPv4 and IPv6 QPPB with egress QoS policy is supported on all Ethernet and SIP-700 line cards.
- IPv4 and IPv6 QPPB with ingress QoS policy is supported on the first generation ASR9000 Ethernet line cards.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **set qos-group** *qos-group-value*
4. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**route-policy** *name* **Example:** `RP/0/RSP0/CPU0:router(config)# route-policy r1`	Enters route policy configuration mode and specifies the name of the route policy to be configured.
Step 3	**set qos-group** *qos-group-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set qos-group 30`	Sets the QoS group identifiers. The set qos-group command is supported only on an ingress policy.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Applying the Route Policy to BGP

This task applies the route policy to BGP.

Prerequisites

Configure BGP and CEF.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** { **ipv4** | **ipv6** } *address-family-modifier*
4. **table-policy** *policy-name*
5. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**router bgp** *as-number* **Example:** `RP/0/RSP0/CPU0:router(config)# router bgp 120`	Enters BGP configuration mode.
Step 3	**address-family** { **ipv4** | **ipv6**} *address-family-modifier* **Example:** `RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast`	Enters address-family configuration mode, allowing you to configure an address family.
Step 4	**table-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-bgp-af) # table-policy qppb a1`	Configures the routing policy for installation of routes to RIB.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring QPPB on the Desired Interfaces

This task applies QPPB to a specified interface. The traffic begins to be classified, based on matching prefixes in the route policy. The source or destination IP address of the traffic can be used to match the route policy.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **ipv4 | ipv6 bgp policy propagation input**{**ip-precedence**|**qos-group**} {**destination**[*ip-precedence* {*destination*|*source*}] {**source**[*ip-precedence* {*destination*|*source*}]
4. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface pos 0/2/0/0`	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 3	**ipv4 | ipv6 bgp policy propagation input**{**ip-precedence**|**qos-group**} {**destination**[*ip-precedence* {*destination*|*source*}] {**source**[*ip-precedence* {*destination*|*source*}] **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination`	Enables QPPB on an interface
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Configuring QPPB on the GRE Tunnel Interfaces

This task applies QPPB to a GRE tunnel interface. The traffic begins to be classified, based on matching prefixes in the route policy. The source or destination IP address of the traffic can be used to match the route policy.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip***number*
3. **ipv4 address** *ipv4-address subnet-mask*

4. **ipv6 address** *ipv6-prefix/prefix-length*
5. **ipv4 | ipv6 bgp policy propagation input**{**ip-precedence**|**qos-group**} {**destination**[*ip-precedence* {*destination*|*source*}] {**source**[*ip-precedence* {*destination*|*source*}]
6. **tunnel source** *type path-id*
7. **tunnel destination** *ip-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface tunnel-ip***number* **Example:** `RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 4000`	Enters interface configuration mode and associates one or more interfaces to the VRF.
Step 3	**ipv4 address** *ipv4-address subnet-mask* **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.255.255.0`	Assigns an IP address and subnet mask to the tunnel interface.
Step 4	**ipv6 address** *ipv6-prefix/prefix-length* **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv6 address 100:1:1:1::1/64`	Specifies an IPv6 network assigned to the interface.
Step 5	**ipv4 | ipv6 bgp policy propagation input**{**ip-precedence**|**qos-group**} {**destination**[*ip-precedence* {*destination*|*source*}] {**source**[*ip-precedence* {*destination*|*source*}] **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 bgp policy propagation input qos-group destination`	Enables QPPB on the GRE tunnel interface
Step 6	**tunnel source** *type path-id* **Example:** `RP/0/RSP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1`	Specifies the source of the tunnel interface.

	Command or Action	Purpose
Step 7	**tunnel destination** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-if)# tunnel destination 100.100.100.20`	Defines the tunnel destination.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

QPPB Scenario

Consider a scenario where in traffic is moving from Network1 to Network2 through (a single) router port1 and port2. If QPPB is enabled on port1, then

- for qos on ingress: attach an ingress policy on the interface port1.
- for qos on egress: attach an egress policy on interface port2.

Configuring Hierarchical Ingress Policing

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **service-policy** *policy-name*
5. **police rate percent** *percentage*
6. **conform-action** *action*
7. **exceed-action** *action*
8. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map parent`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change and enters the policy map class configuration mode.
Step 4	**service-policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy child`	Specifies the service-policy as a QoS policy within a policy map.
Step 5	**police rate percent** *percentage* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate percent 50`	Configures traffic policing and enters policy map police configuration mode.
Step 6	**conform-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action transmit`	Configures the action to take on packets that conform to the rate limit. The allowed action is **transmit** that transmits the packets.
Step 7	**exceed-action** *action* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit. The allowed action is **drop** that drops the packet.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Policer Bucket Sharing

Perform these tasks to enable policer bucket sharing in both the egress and ingress directions.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match precedence**[*number* | *name*]
4. **end-class-map**
5. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
6. **match precedence**[*number* | *name*]
7. **end-class-map**
8. **policy-map** [**type qos**] *policy-name*
9. **class** *class-name*
10. **police bucket shared** *policer instance name***rate***value*
11. **exit**
12. **class** *class-name*
13. **police bucket referred** *policer instance name*
14. **exit**
15. **end-policy-map**
16. **interface** *type interface-path-id*
17. **service-policy input** *policy-map*
18. **service-policy output** *policy-map*
19. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map class1`	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode. If you specify **match-any**, any one 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 match criteria.
Step 3	**match precedence**[*number* | *name*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match precedence 5`	Identifies IP precedence values as match criteria. The range is from 0 to 7. Reserved keywords can be specified, instead of numeric values.
Step 4	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map class2`	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode. If you specify **match-any**, any one 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 match criteria.
Step 6	**match precedence**[*number* | *name*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match precedence 1`	Identifies IP precedence values as match criteria. The range is from 0 to 7. Reserved keywords can be specified, instead of numeric values.
Step 7	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 8	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

	Command or Action	Purpose
Step 9	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 10	**police bucket shared** *policer instance name***rate***value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# policer bucket shared policy1 rate 2Mbps`	Defines and shares a policer bucket. In this example, shared policer bucket *policy1* is created to rate limit traffic at 2Mbps.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 12	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change.
Step 13	**police bucket referred** *policer instance name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# policer bucket referred policy1`	Refers a shared policer bucket. In this example, policer bucket *policy1* is referred.
Step 14	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 15	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 16	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 100/0/0/0`	Configures an interface and enters the interface configuration mode.
Step 17	**service-policy input** *policy-map* **Example:**	Attaches a policy map to an input interface to be used as the service policy for that interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-if)# service-policy input policy1`	
Step 18	**service-policy output** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output policy1`	Attaches a policy map to an output interface to be used as the service policy for that interface.
Step 19	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Overview of Multiple QoS Policy Support

In Cisco Common Classification Policy Language (C3PL), the order of precedence of a class in a policy is based on the position of the class in the policy, that is, the class-map configuration which appears first in a policy-map has higher precedence. Also, the actions to be performed by the classified traffic are defined inline rather than using action templates. As a result of these two characteristics, aggregated actions cannot be applied to traffic that matches different classes.

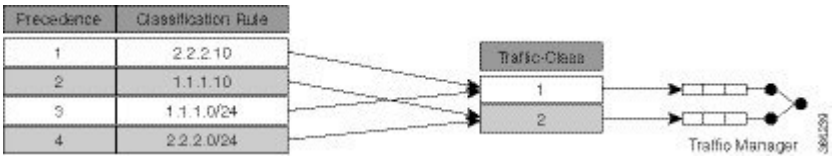
In order to overcome this limitation, the “Multiple QoS Policy Support” feature is introduced. This feature enables the users to apply aggregated actions to various classes of traffic and apply multiple QoS policies on an interface.

Use Case — Multiple QoS Policy Support

Consider a scenario where:

- The classification rules must be applied at different precedence levels.
- Each classification rule must be associated with non-queueing actions (that is, policing/marking).
- Multiple classification rules at different precedence levels must be mapped to a traffic-class.
- Each traffic-class or a group of traffic-classes must be associated with a single queue.

The figure below provides a detailed explanation of the above explained scenario—



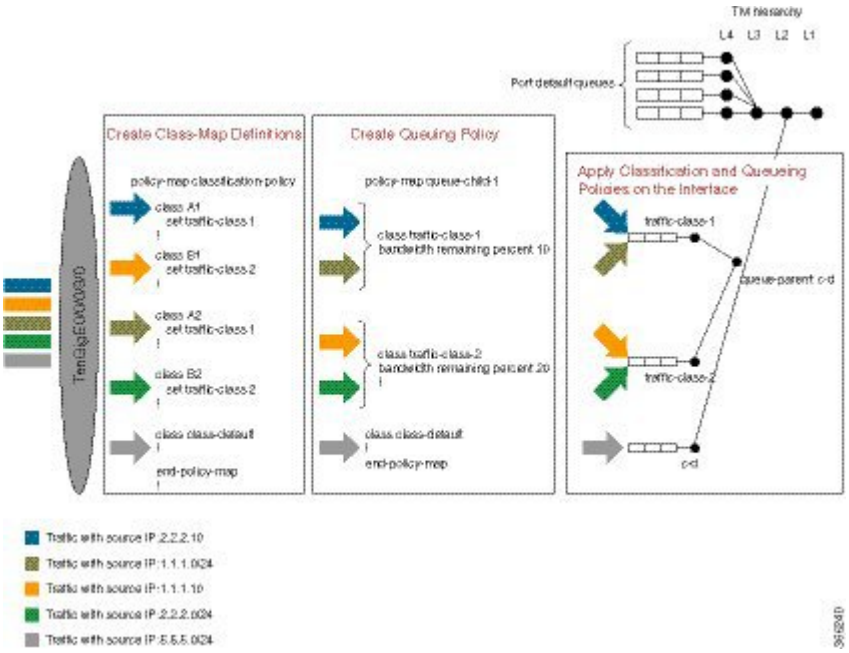
In this example, if the traffic packet matches 2.2.2.10 or 1.1.1.0/24, then the traffic packet is forwarded to the queue that is associated with traffic-class 1. And if the traffic packet matches 1.1.1.10 or 2.2.2.0/24, then the traffic packet is forwarded to the queue that is associated with traffic class 2.

With the existing Modular Quality of Service, we have the following limitations in order to achieve the above mentioned requirement—

1. Packets are matched in the order of precedence that is defined based on the position of the class-maps. There is no way to explicitly specify precedence for a class-map.
2. A queuing action under a class-map in a policy-map, creates a queue for that class.
3. Queues cannot be shared across class-maps.

These limitations can be overcome by separating classification from queuing. By doing this, it is possible to reorder the class-map from higher precedence to lower precedence and also share queues with multiple class-maps.

The example below depicts the implementation—



In this example, 4 classes A1, A2, B1, and B2 are created. Later, classification policies and queueing policies for these classes (A1, A2, B1, and B2) are created. After this, both the classification and queuing policies are applied to the interface. The detailed configuration steps are explained in the following section.

Configuring Multiple QoS Policy Support

In brief, configuring Multiple QoS policy support involves the following steps—

1. Configure Class Map—In this procedure, the traffic classes are defined.

```
/*Defining ACLs for Traffic Filtering*/
ipv4 access-list acl-a1
 10 permit ipv4 host 2.2.2.10 any
ipv4 access-list acl-b1
 10 permit ipv4 host 1.1.1.10 any
ipv4 access-list acl-a2
 10 permit ipv4 1.1.1.0/24 any
ipv4 access-list acl-b2
 10 permit ipv4 2.2.2.0/24 any
!
/*Creating Class Maps*/
class-map match-any A1
 match access-group ipv4 acl-a1
class-map match-any B1
 match access-group ipv4 acl-b1
class-map match-any A2
 match access-group ipv4 acl-a2
class-map match-any B2
 match access-group ipv4 acl-b2

class-map match-any traffic-class-1
 match traffic-class 1
class-map match-any traffic-class-2
 match traffic-class 2
```

2. Configure Policy—In this procedure, the classification and the queuing policies are created.

```
/*Creating Classification Policy*/
policy-map classification-policy
class A1
  set traffic-class 1
 class B1
  set traffic-class 2
class A2
  set traffic-class 1
 class B2
  set traffic-class 2
class class-default

!
/*Creating Queuing Policy*/
policy-map queue-parent
 class class-default
  service-policy queue-child
  shape average 50 mbps
policy-map queue-child
 class traffic-class-1
  bandwidth remaining percent 10
 class traffic-class-2
  bandwidth remaining percent 20
 !
 class class-default
 !
 end-policy-map
```

3. Apply Multiple Services on an Interface—In this procedure, the classification and queuing policies are applied on the interface.

```
/*Applying Policies on an Interface*/
Interface TenGigE0/0/0/3/0
service-policy output classification-policy
service-policy output queue-parent
```

To summarize, two policies (classification and queuing policies) are applied in the Egress direction. The classification policy executes first and classifies traffic at different precedence levels and marks the traffic-class field. The queuing policy executes second, matches on the traffic-class field to select the queue. For traffic matching in different classification precedence to share the same queue, mark the traffic-class field with the same value.

Verification

The **show qos interface** *interface-name* **output** command displays:

- per class per output policy QoS configuration values
- queuing policy followed by the classification policy
- traffic-classes matched by each class in queuing-policy

```
Router#show qos interface TenGigE 0/0/0/3/0 output
Interface: TenGigE0/0/0/3/0 output
Bandwidth configured: 50000 kbps Bandwidth programed: 50000 kbps
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 50000 kbps
Policy: queue-parent Total number of classes: 4
------------------------------------------------------------------------
Level: 0 Policy: queue-parent Class: class-default
Matches: traffic-classes : { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
 63,} and no traffic-class
QueueID: N/A
Shape CIR : NONE
Shape PIR Profile : 8 (Grid) Scale: 134 PIR: 49920 kbps  PBS: 624000 bytes
WFQ Profile: 3/9 Committed Weight: 10 Excess Weight: 10
Bandwidth: 0 kbps, BW sum for Level 0: 0 kbps, Excess Ratio: 1
------------------------------------------------------------------------
Level: 1 Policy: queue-child Class: traffic-class-1
Matches: traffic-classes : { 1}
Parent Policy: queue-parent Class: class-default
QueueID: 1040402 (Priority Normal)
Queue Limit: 66 kbytes Abs-Index: 19 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/19 Committed Weight: 20 Excess Weight: 20
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 10
------------------------------------------------------------------------
Level: 1 Policy: queue-child Class: traffic-class-2
Matches: traffic-classes : {2}
Parent Policy: queue-parent Class: class-default
QueueID: 1040403 (Priority Normal)
Queue Limit: 126 kbytes Abs-Index: 29 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/39 Committed Weight: 40 Excess Weight: 40
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 20
------------------------------------------------------------------------
Level: 1 Policy: queue-child Class: class-default
Matches: traffic-classes : { 0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,}
and no traffic-class
Parent Policy: queue-parent Class: class-default
QueueID: 1040404 (Priority Normal)
Queue Limit: 446 kbytes Abs-Index: 52 Template: 0 Curve: 0
Shape CIR Profile: INVALID
WFQ Profile: 3/98 Committed Weight: 139 Excess Weight: 139
```

```
Bandwidth: 0 kbps, BW sum for Level 1: 0 kbps, Excess Ratio: 70
------------------------------------------------------------------------
Interface: TenGigE0/0/0/3/0 output
Bandwidth configured: 10000000 kbps Bandwidth programed: 10000000 kbps
ANCP user configured: 0 kbps ANCP programed in HW: 0 kbps
Port Shaper programed in HW: 0 kbps
Policy: classification-policy Total number of classes: 5
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A1
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 59 (Single)
Conform: 100000 kbps (100 mbps) Burst: 1250000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B1
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 60 (Single)
Conform: 200000 kbps (200 mbps) Burst: 2500000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: A2
Set traffic-class : 1
QueueID: 0 (Port Default)
Policer Profile: 61 (Single)
Conform: 300000 kbps (300 mbps) Burst: 3750000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: B2
Set traffic-class : 2
QueueID: 0 (Port Default)
Policer Profile: 62 (Single)
Conform: 400000 kbps (400 mbps) Burst: 5000000 bytes (0 Default)
Child Policer Conform: TX
Child Policer Exceed: DROP
Child Policer Violate: DROP
------------------------------------------------------------------------
Level: 0 Policy: classification-policy Class: class-default
QueueID: 0 (Port Default)
------------------------------------------------------------------------
```

Restrictions for Multiple QoS Policy Support

Policy Classification Restrictions

- Classification policy must always be executed before the queuing policy. Also, queuing actions are not supported within a classification policy.
- Classification policy supports unconditional set traffic-class actions. The valid values for **set traffic-class** are 0 – 63.
- In a conditional policer action, **set traffic-class** action is not supported.

- At least one **set traffic-class** action must be present for a policy to be considered a classification policy in the multi policy context.
- Only two additional packet fields can be unconditionally set along with **set traffic-class**.
- Class-maps in a classification policy cannot be used to match on traffic-class.
- Only one **set traffic-class** action is permitted in a hierarchy (either parent or child).
- Flow aware and shared policers are not supported.
- In a three-level policy, **set traffic-class** action is permitted only at the lowest two-levels.
- In a policer action, conditional **set traffic-class** is not supported.

Queuing Policy Restrictions

- Queuing policy can only classify on **traffic-class** field.
  - Valid values for **match traffic-class** are 0-63.
  - Class-maps can match up to 8 discreet traffic-class values or traffic-class ranges.
- At least one class-map with **match traffic-class** must be present for a policy to be considered a queuing policy in the multiple qos policy support feature.
- Class-map with match **not** traffic-class is not supported.
- Non-queuing actions like policer and set are not supported.
- Since policer is not supported in queuing policy, when priority level 1 queue is used, the service rate computed for lower priority queues is very low (with priority 1 utilizing all the bandwidth, the bandwidth remaining for lower priority queues is very low). Due to the same reason, **minimum bandwidth** is also not be supported with priority level 1. However, **bandwidth remaining ratio** may be used instead of **minimum bandwidth**. Since the **default queue-limit** and **time based queue-limit** configurations use service-rate to calculate **queue-limit** in bytes, it is recommended to explicitly configure queue-limit in bytes when using priority 1 queue.

Applying Multiple Services on an Interface Restrictions

- Applying multiple polices is supported only when one policy is a classification policy and the other policy is a queuing policy.
- Applying multiple polices (not more than 2 policies) is supported only in the egress direction. Applying more than 1 policy in the ingress direction is not supported.
- Applying multiple policies is supported only on the following interfaces:
  - Main-interface
  - Sub-interface
  - Bundle interface
  - Bundle sub-interface
- Applying Multi policies is not supported on the following interfaces:

- PWHE
- GRE
- BVI
- Satellite interfaces

- Multi policies are only supported on Cisco ASR 9000 High Density 100GE Ethernet line cards, Cisco ASR 9000 Enhanced Ethernet line cards, and Cisco ASR 9000 Ethernet line cards.
- The same classification policy cannot be applied with different queuing policies on a different interface of the same line card.
- Classification policy and queuing policy cannot be applied with any of the following feature options
  - account
  - service-fragment-parent
  - shared-policy-instance
  - subscriber-parent

Policy Combinations

The different policy combinations are displayed in the below table:

Policies Already Applied on the Interface			**Policies that are yet to be Applied on the Interface**			**Accepted**
Regular Policy (no set/match traffic-class)	**Classification Policy**	**Queuing Policy**	**Regular Policy (no set/match traffic-class)**	**Classification Policy**	**Queuing Policy**	
Yes	No	No	Any combination			No
No	Yes	No	No	No	No	No
			No	No	No	No
			Yes	No	No	No
			No	No	Yes	Yes
			No	Yes	Yes	No
			No	Yes	Yes	No

Policies Already Applied on the Interface			Policies that are yet to be Applied on the Interface			Accepted
No	No	Yes	No	Yes	No	Yes
			Yes	Yes	No	No
			No	No	Yes	No
			No	Yes	Yes	No
			No	Yes	Yes	No
No	Yes	Yes	Any combination			No

Note To change a policy to a different policy of the same type you must first remove the existing policy and then apply the new policy.

Multi Policy and Interface Hierarchy

Multi Policy and Interface Hierarchy is displayed in the below table:

Main/Bundle Interface			Sub/Bundle Sub Interface			Comments
Regular Policy (no set/match traffic-class)	**Classification Policy**	**Queuing Policy**	**Regular Policy**	**Classification Policy**	**Queuing Policy**	
Non Port Shaper Policy	No	No	No policy allowed on child interfaces			The policy is enabled and is inherited by all the child interfaces. The same policy executes on the main interface and all its child interface traffic.
No	Yes	No	No policy allowed on child interfaces			Policy is disabled
No	Yes	No	No policy allowed on child interfaces			Policy is disabled

Main/Bundle Interface			Sub/Bundle Sub Interface	Comments
No	Yes	Yes	No policy allowed on child interfaces	Both policies are enabled and are inherited by all the child interfaces. The classification policy is executed first, followed by the queuing policy on the main interface and all its child interface traffic.

Main/Bundle Interface			Sub/Bundle Sub Interface			Comments
Port Shaper Policy	No	No	Yes	No	No	Main interface policy enabled. Sub interface policy is enabled and uses the port shaper rate as the reference bandwidth. If port shaper is applied after sub interface policy, then the applied sub interface policy will be updated with the new reference bandwidth. If the port shaper rate is lower than any sub interface policy rate, then the port shaper policy is rejected.
			No	Yes	No	Main interface policy enabled. Sub interface policy is disabled
			No	No	Yes	Main interface policy enabled. Sub interface policy is disabled

Main/Bundle Interface			Sub/Bundle Sub Interface			Comments
			No	Yes	Yes	Main interface policy enabled. Both the sub interface policies are enabled and both the policies use the port shaper rate as the reference bandwidth. If port shaper is applied after sub interface policies, then both the applied sub interface policies will be updated with the new reference bandwidth. If the port shaper rate is lower than any sub interface policy rate, then the port shaper policy is rejected.

Main/Bundle Interface	Sub/Bundle Sub Interface			Comments
Non port shaper policy not allowed on main interface	Yes	No	No	Policy is enabled
	No	Yes	No	Policy is disabled
	No	No	Yes	Policy is disabled
	No	Yes	Yes	Both policies are enabled and the classification policy is executed first followed by the queuing policy.

Statistics

Users can retrieve and verify the classification and queuing policy statistics per interface (per direction) in a multi-policy configuration, using the show policy-map interface interface-name output pmap-name command.

The **show policy-map interface all**, **show policy-map interface** *interface-name*, and **show policy-map interface** *interface-name* output displays statistics for all the policies in the each direction on an interface.

Classification Policy

- Statistics counters are allocated for every leaf class and updated for every packet match – match counters.
- Statistics counters are allocated for each policer used in the policy and updated during policing operation.
- There are no queue counters.

Queuing policy

- Each queue has a transmit and drop statistics counter associated with it which is updated for every queuing operation.
- There is a separate drop counter for each WRED color/curve in a queuing class.
- No match counters are allocated for a class. Instead, match counters is derived by adding the queue transmit statistics and all the queue drop statistics.

Example: Egress Policy Classification Statistics

```
Router# show policy-map interface TenGigE 0/0/0/3/9.1 output pmap-name classification

TenGigE0/0/0/3/9.1 output: classification
Class A1
Classification statistics          (packets/bytes)     (rate - kbps)
```

```
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class B1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class A2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class B2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :            83714645/83714645000          100006
    Transmitted         : N/A
    Total Dropped       : N/A
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         : N/A
    Total Dropped       : N/A
```

Example: Egress Queuing Policy Statistics

```
Router# show policy-map interface TenGigE 0/0/0/3/9.1 output pmap-name queueing

TenGigE0/0/0/3/9.1 output: queueing
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          534226989/534226989000         400067
    Transmitted         :          355884870/355884870000         280381
    Total Dropped       :          106961210/106961210000         119726
  Policy queueing-child Class traffic-class-1
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :          178155114/178155114000         200014
      Transmitted         :          178155114/178155114000         200014
      Total Dropped       :                  0/0                    0
    Queueing statistics
      Queue ID                              : 647264
      High watermark                        : N/A
      Inst-queue-len  (packets)             : 0
      Avg-queue-len                         : N/A
      Taildropped(packets/bytes)            : 0/0
      Queue(conform)      :          178155114/178155114000         200014
      Queue(exceed)       :                  0/0                    0
      RED random drops(packets/bytes)       : 0/0
  Policy queueing-child Class traffic-class-2
    Classification statistics          (packets/bytes)     (rate - kbps)
      Matched             :          178098546/178098546000         200111
      Transmitted         :           71137336/71137336000          80385
      Total Dropped       :          106961210/106961210000         119726
    Queueing statistics
      Queue ID                              : 647265
      High watermark                        : N/A
      Inst-queue-len  (packets)             : 1620
      Avg-queue-len                         : N/A
      Taildropped(packets/bytes)            : 106961210/106961210000
      Queue(conform)      :           71137336/71137336000          80385
      Queue(exceed)       :                  0/0                    0
      RED random drops(packets/bytes)       : 0/0
Policy egress-queueing-child Class class-default
    Classification statistics          (packets/bytes)     (rate - kbps)
```

```
  Matched             :                  0/0                    0
  Transmitted         :                  0/0                    0
  Total Dropped       :                  0/0                    0
Queueing statistics
  Queue ID                            : 647266
  High watermark                      : N/A
  Inst-queue-len  (packets)           : 0
  Avg-queue-len                       : N/A
  Taildropped(packets/bytes)          : 0/0
  Queue(conform)      :                  0/0                    0
  Queue(exceed)       :                  0/0                    0
  RED random drops(packets/bytes)     : 0/0
```

Restrictions for Statistics

- The clear counters all is not supported for multi policy.
- The match statistics in a queuing policy are derived from the queue statistics. Therefore, there is no match statistics available for classes, which do not have a dedicated queue. Statistics for packets matching such classes (with no dedicated queue) shows up in the match statistics in the corresponding queuing class.
- Per classification class queue transmit and drop statistics are not available; only aggregated queue transmit and drop statistics are available.

Policy Modification

Modifying a policy when it is already applied on the interface, which is referred to as “In-place modification” is supported for both classification policy and queuing policy.

When a classification policy (or an ACL used in a classification policy) is modified, the previously applied classification policy and the corresponding queueing policy are removed from all interfaces. Then, the modified version of the classification policy is applied and the configured queuing policy is reapplied on all interfaces. If there is an error on any interface when applying the modified version of the classification policy, then all changes are reverted. That is, the modified version is removed from all interfaces on which it was applied and the previous (original, unmodified) version of both policies are reapplied on all interfaces. The modification attempt is terminated.

This modification process is the same for any modifications of the queuing policy. The previously applied queuing policy is removed and the modified version is applied (along with a reapplication of the corresponding classification policy.) In cases of error, the modification attempt is terminated and the previous versions of both policies are reapplied on all interfaces.

Since both classification and queuing polices are removed and then reapplied when either policy is modified, statistic counters in both policies is reset after a successful or failed modification.

Policy Modification Restrictions

- When a classification policy is applied on an interface, any modification, which changes it to a non-classification policy, for example, removing all set traffic-class actions or adding a class that matches on traffic-class, is rejected.

  So, in order to modify a classification policy to a non-classification policy, users must first remove the policy from all the interfaces and then modify.

- When a queuing policy is applied on an interface, any modification, which changes it to a non-queuing policy, for example, removing all classes that match on traffic-class, or adding a non-queuing action

(police or set), is rejected. So, in order to modify a queuing policy to a non-queuing policy, users must first remove the policy from all the interfaces and then modify.

Supported Features by Multi Policies

The following table displays the features supported and not supported by Multi policies—

Feature	Multi Policy- Classification	Multi Policy- Queuing
Classification	Except traffic-class field, all other fields that are currently supported	Only on traffic-class field
Unconditional Marking	Traffic-class and all other fields that are currently supported	No
1R2C	Yes	
1R3C		
2R3C		
Policer/Conditional Marking	Except traffic-class field, all other fields that are currently supported	
Grand Parent Policer	Yes	
Color Aware Policer		
Conform Aware Policer		
Shared Policer	No	
Flow Aware Policer	No	
Priority	No	
Shape		
Bandwidth		
Bandwidth Remaining		Yes
WRED		Supported but no WRED classification on traffic-class
Statistics	Match counters, policer exceed/conform/violate counters	Match, queue transmit, queue drop, WRED drop counters
Rate Calculation	Match and policer statistics	Match and queue statistics
SPI	No	No
Port Shaper	Yes	Yes
Policy Inheritance	Yes	Yes

Configuration Examples for Configuring Modular QoS Packet Classification

Traffic Classes Defined: Example

In this example, two traffic classes are created and their match criteria are defined. For the first traffic class called class1, ACL 101 is used as the match criterion. For the second traffic class called class2, ACL 102 is used as the match criterion. Packets are checked against the contents of these ACLs to determine if they belong to the class.

```
class-map class1
  match access-group ipv4 101
  exit
!
class-map class2
  match access-group ipv4 102
  exit
```

Use the **not** keyword with the **match** command to perform a match based on the values of a field that are not specified. The following example includes all packets in the class qos_example with a DSCP value other than 4, 8, or 10.

```
class-map match-any qos_example
  match not dscp 4 8 10
!
end
```

Traffic Policy Created: Example

In this example, a traffic policy called policy1 is defined to contain policy specifications for the two classes—class1 and class2. The match criteria for these classes were defined in the traffic classes created in the "Traffic Classes Defined: Example" section on page 68.

For class1, the policy includes a bandwidth allocation request and a maximum byte limit for the queue reserved for the class. For class2, the policy specifies only a bandwidth allocation request.

```
policy-map policy1
  class class1
    bandwidth 3000 kbps
    queue-limit 1000 packets
!
  class class2
    bandwidth 2000 kbps
!
  class class-default
!
end-policy-map
!
end
```

Traffic Policy Attached to an Interface: Example

This example shows how to attach an existing traffic policy to an interface (see the "Traffic Classes Defined: Example" section on page 68). After you define a traffic policy with the policy-map command, you can attach it to one or more interfaces to specify the traffic policy for those interfaces by using the **service-policy** command in interface configuration mode. Although you can assign the same traffic policy to multiple interfaces, each interface can have only one traffic policy attached at the input and only one traffic policy attached at the output.

```
interface gigabitethernet 0/1/0/9
  service-policy output policy1
  exit
!
```

Traffic Policy Attached to Multiple Subinterfaces: Example

The following example shows how to attach an existing traffic policy to multiple subinterfaces. After you define a traffic policy with the **policy-map** command, you can attach it to one or more subinterfaces using the service policy command in subinterface configuration mode.

```
interface gigabitethernet 0/1/0/0.1
  service-policy input policy1 shared-policy-instance ethernet101
  exit
!
interface gigabitethernet 0/1/0/0.2
  service-policy input policy1 shared-policy-instance ethernet101
  exit
```

Traffic Policy Attached to a Bundle Interface: Example

The following example shows how to attach an existing traffic policy to a bundle interface. After you define a traffic policy with the **policy-map** command, you can attach it to one or more bundle subinterfaces using the service policy command in subinterface configuration mode.

```
interface Bundle-Ether 100.1
  service-policy tripleplaypolicy shared-policy-instance subscriber1
  exit
!
interface Bundle-Ether 100.2
  service-policy output tripleplaypolicy shared-policy instance subscriber1
  exit
```

EFP Load Balancing with Shared Policy Instance: Example

The following examples show how to configure load balancing of an EFP when SPI is implemented. For additional information on EFP load balancing on link bundles, see the Cisco IOS XR Interface and Hardware Component Configuration Guide.

Configuring a Bundle Interface: Example

```
interface Bundle-Ether 50
interface gigabitethernet 0/1/0/5
```

```
  bundle id 50 mode active
interface gigabitethernet 0/1/0/8
  bundle id 50 mode active
```

Configuring Two Bundle EFPs with the Load Balance Options: Example

This example configures the traffic for two bundle EFPs go over the same physical member link.

```
interface Bundle-Ether 50.25 l2transport
  encapsulation dot1q 25
  bundle load-balance hash-select 2
!
interface Bundle-Ether 50.36 l2transport
  encapsulation dot1q 36
  bundle load-balance hash-select 2
```

Default Traffic Class Configuration: Example

This example shows how to configure a traffic policy for the default class of the traffic policy called policy1. The default class is named class-default, consists of all other traffic, and is being shaped at 60 percent of the interface bandwidth.

```
policy-map policy1
  class class-default
    shape average percent 60
```

class-map match-any Command Configuration: Example

This example illustrates how packets are evaluated when multiple match criteria exist. Only one match criterion must be met for the packet in the **class-map match-any** command to be classified as a member of the traffic class (a logical OR operator). In the example, protocol IP OR QoS group 4 OR access group 101 have to be successful match criteria:

```
class-map match-any class1
  match protocol ipv4
  match qos-group 4
  match access-group ipv4 101
```

In the traffic class called class1, the match criteria are evaluated consecutively until a successful match criterion is located. Each matching criterion is evaluated to see if the packet matches that criterion. If the packet matches at least one of the specified criteria, the packet is classified as a member of the traffic class.

Note The **match qos-group** command is supported only on egress policies.

Class-based Unconditional Packet Marking: Examples

These are typical class-based unconditional packet marking examples:

IP Precedence Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a previously defined class map called *class1* through the use of the **class** command, and then the service policy is attached to the output POS interface 0/1/0/0. The IP precedence bit in the ToS byte is set to 1:

```
policy-map policy1
  class class1
    set precedence 1
!
interface pos 0/1/0/0
  service-policy output policy1
```

IP DSCP Marking Configuration: Example

In this example, a service policy called policy1 is created. This service policy is associated to a previously defined class map through the use of the **class** command. In this example, it is assumed that a class map called class1 was previously configured and new class map called class2 is created.

In this example, the IP DSCP value in the ToS byte is set to 5:

```
policy-map policy1
  class class1
    set dscp 5

  class class2
    set dscp ef
```

After you configure the settings shown for voice packets at the edge, all intermediate routers are configured to provide low-latency treatment to the voice packets, as follows:

```
class-map voice
  match dscp ef
policy-map qos-policy
  class voice
    priority level 1
    police rate percent 10
```

QoS Group Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the input direction on a GigabitEthernet interface 0/1/0/9. The qos-group value is set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set qos-group 1
  !
interface GigabitEthernet 0/1/0/9
  service-policy input policy1
```

Note The **set qos-group** command is supported only on an ingress policy.

CoS Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the output direction on a 10-Gigabit Ethernet interface, TenGigE0/1/0/0. The IEEE 802.1p (CoS) bits in the Layer 2 header are set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set cos 1
  !
interface TenGigE0/1/0/0
interface TenGigE0/1/0/0.100
  service-policy output policy1
```

MPLS Experimental Bit Imposition Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the input direction on a 10-Gigabit Ethernet interface, TenGigE0/1/0/0. The MPLS EXP bits of all imposed labels are set to 1.

```
class-map match-any class1
  match protocol ipv4
  match access-group ipv4 101

policy-map policy1
  class class1
    set mpls exp imposition 1
 !
interface TenGigE0/1/0/0
  service-policy input policy1
```

Note The **set mpls exp imposition** command is supported only on an ingress policy.

MPLS Experimental Topmost Marking Configuration: Example

In this example, a service policy called *policy1* is created. This service policy is associated to a class map called *class1* through the use of the **class** command, and then the service policy is attached in the output direction on a 10-Gigabit Ethernet interface, TenGigE0/1/0/0. The MPLS EXP bits on the TOPMOST label are set to 1:

```
class-map match-any class1
  match mpls exp topmost 2

policy-map policy1
```

```
  class class1
    set mpls exp topmost 1
  !
interface TenGigE0/1/0/0
  service-policy output policy1
```

QoS Policy Propagation using BGP: Examples

These are the IPv4 and IPv6 QPPB examples:

Applying Route Policy: Example

In this example, BGP is being configured for the IPv4 address family:

```
router bgp 100
 bgp router-id 19.19.19.19
 address-family ipv4 unicast
  table-policy qppbv4_dest
 !
 neighbor 10.10.10.10
  remote-as 8000
  address-family ipv4 unicast
   route-policy pass-all in
   route-policy pass-all out
```

In this example, BGP is being configured for the IPv6 address family:

```
router bgp 100
 bgp router-id 19.19.19.19
 address-family ipv6 unicast
  table-policy qppbv6_dest
 !
 neighbor 1906:255::2
  remote-as 8000
  address-family ipv6 unicast
   route-policy pass-all in
   route-policy pass-all out
```

Applying QPPB on a Specific Interface: Example

This example shows applying QPPBv4 (address-family IPv4) for a desired interface:

```
config
interface POS0/0/0/0
ipv4 address 10.1.1.1
ipv4 bgp policy propagation input qos-group destination
end
commit
!
```

This example shows applying QPPBv6 (address-family IPv6) for a desired interface:

```
config
interface POS0/0/0/0
ipv6 address 1906:255::1/64
ipv6 bgp policy propagation input qos-group destination
end
commit
!
```

Applying QPPB on a GRE Tunnel Interface: Example

This example shows applying QPPBv4 (address-family IPv4) for a GRE tunnel interface:

```
config
interface tunnel-ip 4000
ipv4 address 10.1.1.1
ipv4 bgp policy propagation input qos-group destination
tunnel source TenGigE0/2/0/1
tunnel destination 145.12.5.2
end
commit
!
```

This example shows applying QPPBv6 (address-family IPv6) for a GRE tunnel interface:

```
config
interface tunnel-ip 3000
ipv6 address 1906:255::1/64
ipv6 bgp policy propagation input qos-group destination
tunnel source TenGigE0/2/0/1
tunnel destination 145.12.5.2
end
commit
!
```

In-Place Policy Modification: Example

In this example, the precedence is changed from 3 to 5 after the policy is defined and attached to an interface:

Define a class:

```
class-map match-any class1
 match cos 7
 end-class-map
```

Define a policy map that uses the class:

```
policy-map policy1
 class class1
 set precedence 3
```

Attach the policy map to an interface:

```
interface gigabitethernet 0/6/0/1
 service-policy output policy1
 commit
```

Modify the precedence value of the policy map:

```
policy-map policy1
 class class1
 set precedence 5
 commit
```

Note The modified policy *policy1* takes effect on all the interfaces to which the policy is attached. Also, you can modify any class map used in the policy map. The changes made to the class map take effect on all the interfaces to which the policy is attached.

Output from the **show policy-map targets** command indicates that the Gigabit Ethernet interface 0/1/0/0 has one policy map attached as a main policy (as opposed to being attached to a child policy in a hierarchical QoS configuration). Outgoing traffic on this interface is affected if the policy is modified:

```
show policy-map targets

Fri Jul 16 16:38:24.789 DST
1) Policymap: policy1    Type: qos
     Targets (applied as main policy):
       GigabitEthernet0/1/0/0 output
     Total targets: 1

     Targets (applied as child policy):
     Total targets: 0
```

Configuring Inter Class Policer Bucket Sharing: Example

In this example, policer bucket *policy1* is defined and shared by class *class1*. The shared policer bucket *policy1* is referred by class *class2*.

```
configure
class-map class1
 match precedence 5
 !
class-map class2
 match precedence 1
 !
policy-map parent
  class class1
    police bucket shared policy1 rate 2 mbps
  class class2
    police bucket referred policy1
end-policy-map
!
```

Additional References

These sections provide references related to implementing packet classification.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Gett Guide*

Related Topic	Document Title
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Modular of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Route of Cisco Cisco ASR 9000 Series Aggregation Services Route Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

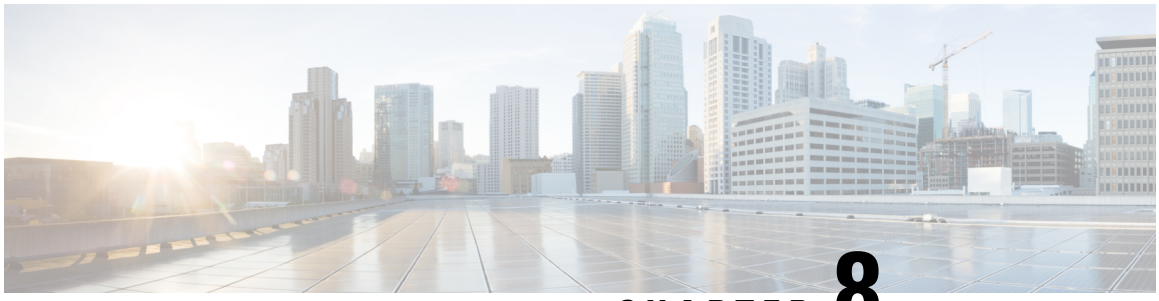
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 8

Modular QoS Deployment Scenarios

This module provides deployment scenarios use cases for specific QoS features or for QoS implementations of features that are described in other technology guides such as L2VPN or MPLS.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
802.1ad DEI	yes	no
Frame Relay QoS	no	yes
IPHC QoS	no	2-Port Channelized OC-12c/DS0 SPA only
L2VPN QoS	yes	yes
MLPPP/MLFR QoS	no	2-Port Channelized OC-12c/DS0 SPA only
MPLS QoS	yes	yes
QoS on Multicast VPN	yes	yes
QoS on NxDS0 Interfaces	no	2-Port Channelized OC-12c/DS0 SPA only

Feature History for QoS Deployment Scenarios on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.2	The L2VPN QoS feature was introduced on ASR 9000 Ethernet Line Cards. The MPLS QoS feature was introduced on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The MLPPP QoS feature was introduced on the SIP 700 for the ASR 9000.
Release 3.9.1	The QoS on Multicast VPN feature was introduced on ASR 9000 Ethernet Line Cards.

Release	Modification
Release 4.0.0	The 802.1ad DEI feature was introduced on the SIP 700 for the ASR 9000. The Frame Relay QoS feature was introduced on the SIP 700 for the ASR 9000. The IP Header Compression QoS feature was introduced on the SIP 700 for the ASR 9000. The L2VPN QoS feature was supported on the SIP 700 for the ASR 9000. The MLFR QoS feature was introduced on the SIP 700 for the ASR 9000. The suspend/resume approach was added for MLPPP and MLFR interfaces. The MPLS QoS feature was supported on the SIP 700 for the ASR 9000. The QoS on NxDS0 Interfaces feature was introduced on the SIP 700 for the ASR 9000.
Release 4.1.0	The VPLS and VPWS QoS feature was introduced.

802.1ad 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. DEI support includes the ability to:

- Police to a certain rate and, based on whether the traffic is conforming or exceeding, mark the DEI as 0 or 1.
- On ingress, police and set up the discard class (even on an interface that is not configured for 802.1ad encapsulation).
- On egress, mark the DEI based on the discard class value (802.1ad interfaces only).

You can manage congestion based on the Drop Eligible Indicator (DEI) bit that is present in 802.1ad frames and 802.1ah frames. DEI support includes the ability to:

- Do weighted random early detection (WRED) based on the value of the DEI bit.
- Do active queue management during traffic congestion on an interface by giving preferential treatment to traffic (bigger thresholds) or set up smaller thresholds for out-of-profile traffic based on a DEI value.

Mark DEI Based on a Policing Action: Example

In this example, the police rate is set to 5 Mbps. Conforming traffic is marked with a DEI value of 0; traffic that exceeds the police rate is marked with a DEI value of 1.

```
policy-map 1ad-mark-dei
class c1
police rate 5 mbps
      conform-action set dei 0
      exceed-action set dei 1
end-policy-map
```

Mark DEI Based on Incoming Fields: Example

In this example, 802.1ad CoS plus DEI is derived from the incoming 802.1q CoS. Packets with a CoS value of 0 are remarked with a DEI value of 1.

```
class-map match-any remark-cos
  match cos 0
end-class-map

policy-map p1
  class remark-cos
  set dei 1
end-policy-map

interface GigabitEthernet0/4/0/39.1 l2transport
 encapsulation dot1q 1
 rewrite ingress tag push dot1ad 5 symmetric
 service-policy input p1
!
```

Congestion Management Using DEI: Example

In this example, congestion is managed by dropping packets with a DEI value of 1 before dropping packets with a DEI value of 0.

```
policy-map dei-sample
  class class-default
    random-detect dei 1 1000 6000
    random-detect dei 0 5000 10000
end-policy-map
```

Frame Relay QoS

The main difference between Frame Relay QoS and other interface types is that you can perform:

- Frame Relay DLCI classification
- Frame Relay DE classification
- Frame Relay DE marking

Note A QoS policy can be applied only to a PVC under a Frame Relay subinterface; it cannot be applied directly to a Frame Relay subinterface.

Frame Relay DLCI Classification

This configuration allows users to match on the Frame Relay DLCI value of packets encapsulated in Frame Relay. Packets that are not Frame Relay encapsulated do not match this configuration.

```
class-map foo
            match frame-relay list of dlci-values
```

The list of DLCI values can contain ranges as well as individual values, as in this example:

```
class-map foo
            match frame-relay dlci 1-100 150 200-300
```

Note DLCI matching is supported only on main interfaces.

Frame Relay DE Classification

This configuration allows the user to match Frame Relay packets that have the discard eligible (DE) bit set in the Frame Relay header:

```
class-map fr_class
  match fr-de 1
```

To match Frame Relay DE bit 0, use this configuration:

```
class-map match-not-fr-de
  match not fr-de 1
```

Note DE bit classification is not supported on Layer 3 interfaces.

Frame Relay DE Marking

In this example, the fr-de bit is set when traffic exceeds the policing committed information rate, so the downward system (when experiencing congestion) discards traffic with the fr-de bit set to 1.

```
policy-map fr_de_marking
 class class-default
  police rate percent 50
   conform-action transmit
   exceed-action set fr-de 1
```

```
 !
!
end-policy-map
```

Note DE bit marking is not supported on Layer 3 interfaces.

Frame Relay QoS: Example

In this example, parent_policy is applied to the Multilink Frame Relay main interface. There are two classes in parent_policy, which match on Frame Relay DLCIs. The Multilink Frame Relay main interface has two Frame Relay PVCs configured (DLCI 16, DLCI 17).

```
show run int multi 0/2/1/0/1
Mon Aug  2 11:34:31.019 UTC
interface Multilink0/2/1/0/1
 service-policy output parent_policy
 encapsulation frame-relay
 frame-relay intf-type dce
!

show run policy-map parent_policy
Mon Aug  2 11:34:36.118 UTC
policy-map parent_policy
 class parentQ_1
  service-policy child_queuing_policy
  shape average 64 kbps
 !
 class parentQ_2
  service-policy child_queuing_policy
  shape average 1 mbps
 !
 class class-default
 !
 end-policy-map
!

show run class-map  parentQ_1 <----- class map parent class dlci=16
Mon Aug  2 11:34:43.363 UTC
class-map match-any parentQ_1
 match frame-relay dlci 16
 end-class-map
!

show run class-map  parentQ_2 <----- class map parent class dlci=17
Mon Aug  2 11:34:45.647 UTC
class-map match-any parentQ_2
 match frame-relay dlci 17
 end-class-map
!

show run int multi 0/2/1/0/1.16 <------ dlci 16 pvc config
Mon Aug  2 11:34:53.988 UTC
interface Multilink0/2/1/0/1.16 point-to-point
 ipv4 address 192.1.1.1 255.255.255.0
 pvc 16
  encap cisco
 !
!
```

```
show run int multi 0/2/1/0/1.17  <------ dlci 17 pvc config
Mon Aug  2 11:34:56.862 UTC
interface Multilink0/2/1/0/1.17 point-to-point
 ipv4 address 192.1.2.1 255.255.255.0
 pvc 17
  encap cisco
 !
!
show run policy-map child_queuing_policy <--------- child policy-map
Mon Aug  2 11:35:05.821 UTC
policy-map child_queuing_policy
 class voice-ip
  priority level 1
  police rate percent 20
  !
 !
 class video
  bandwidth percent 40
 !
 class premium
  service-policy gchild_policy
  bandwidth percent 10
  random-detect discard-class 2 10 ms 100 ms
  random-detect discard-class 3 20 ms 200 ms
  queue-limit 200 ms
 !
 class best-effort
  bandwidth percent 20
  queue-limit 200 ms
 !
 class class-default
 !
 end-policy-map
!

show run policy-map gchild_policy  <-------- grandchild policy map
Mon Aug  2 11:35:15.428 UTC
policy-map gchild_policy
 class premium_g1
  police rate percent 10
  !
  set discard-class 2
 !
 class premium_g2
  police rate percent 50
  !
  set discard-class 3
 !
 class class-default
 !
 end-policy-map
!

show run class-map  <----------- shows all class map configs
Mon Aug  2 11:35:19.479 UTC
class-map match-any video
 match precedence 1
 end-class-map
!
class-map match-any premium
 match precedence 2 3
 end-class-map
!
class-map match-any voice-ip
```

```
 match precedence 0
 end-class-map
!
class-map match-any parentQ_1
 match frame-relay dlci 16
 end-class-map
!
class-map match-any parentQ_2
 match frame-relay dlci 17
 end-class-map
!
class-map match-any premium_g1
 match precedence 2
 end-class-map
!
class-map match-any premium_g2
 match precedence 3
 end-class-map
!
class-map match-any best-effort
 match precedence 4
 end-class-map
!
```

IP Header Compression QoS

An IP Header Compression (IPHC) profile can be enabled on an interface so that the IPHC profile applies only to packets that match a QoS service policy. In this case, the QoS service-policy class attributes determine which packets are compressed. This allows users to fine tune IPHC with greater granularity.

Policy maps are attached to an interface using the **service-policy** command. IPHC action applies only to output service policies. IPHC is not supported on input service policies. (IPHC is supported in the input direction but there is no use case to configure IPHC in an input policy.)

You can configure IPHC using QoS as follows:

- Create a QoS **policy** with the **compress header ip** action.
- Attach the IPHC profile to the interface using the **ipv4 iphc profile** *profile_name* **mode service-policy** command.
- Attach the QoS **policy** with **compress header ip** action using the **service-policy output** command.

You can also display IPHC statistics using the **show policy-map interface** command, as shown in the following example:

```
show policy-map interface Serial0/0/3/0/3:0 output

show policy-map int  Serial0/0/3/0/3:0  output
Mon May 18 22:06:14.698 UTC
Serial0/0/3/0/3:0 output: p1
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                  0/0                   0
    Transmitted         :                  0/0                   0
    Total Dropped       :                  0/0                   0
  Queueing statistics
    Queue ID                              : 0
    High watermark  (Unknown)             : 0
```

```
    Inst-queue-len  (packets)            : 0
    Avg-queue-len   (packets)            : 0
    Taildropped(packets/bytes)           : 0/0
  Compression Statistics
    Header ip rtp
    Sent Total       (packets)           : 880
    Sent Compressed  (packets)           : 877
    Sent full header (packets)           : 342
    Saved            (bytes)             : 31570
    Sent             (bytes)             : 24750
    Efficiency improvement factor        : 2.27
```

IP Header Compression QoS: Example

In this example, IPHC is configured through QoS as an action under the class map using the **compress header ip** command.

The packets are classified according to the criteria in the class maps. The policy map specifies which behavior to apply to which classes. IPHC is enabled using the **compress header ip** action for the class. An IPHC profile with a QoS service policy is attached to a serial interface.

```
class-map match-all voice1
  match precedence 2
class-map match-all voice2
  match access-group acl_iphc

access-list acl_iphc permit udp any range lower-bound src udp port 5000 upper-bound src udp
 port15000 any lower-bound udp dst port 5000 upper-bound dst udp port 15000

ipv4 access-list acl_iphc permit udp any range 5000 15000 any range 5000 15000

policy-map iphc_policy
 class iphc_class_1
  compress header ip
 class iphc_class_2
  compress header ip

interface serial 0/1/0/1:1
 ipv4 iphc profile Profile_3 mode service-policy
  service-policy output iphc_policy

interface Serial 0/2/0/0/1/1/1:1
  ipv4 address 10.0.0.1 255.255.255.252
  ipv4 iphc profile Profile_3 mode service-policy
  service-policy output iphc_policy
  encapsulation ppp
```

L2VPN QoS

This section describes the following Frame Relay L2VPN deployment scenarios:

- Frame Relay <-> Frame Relay over pseudowire
- Frame Relay <-> Ethernet over pseudowire

Note There are local-connect variants of these scenarios that do not go over a pseudowire. This discussion focuses on the pseudowire scenarios.

Frame Relay - Frame Relay Over Pseudowire: Example

This example shows that you can match based on the Frame Relay DLCI on the ingress Frame Relay interface on router PE1 and set the fr-de value. This configuration is carried over the L2VPN pseudowire. When the Frame Relay packet exits router PE2 through the Frame Relay l2transport interface, the fr-de value is intact.

This configuration allows you to manipulate and carry over the Frame Relay QoS values across L2VPN. This figure shows the network topology.

Figure 8: Frame Relay Over MPLS

CE1

```
interface pos0/2/0/0.26
 pvc 26
 ipv4 add 10.0.0.1 255.0.0.0
```

PE1

```
interface pos0/2/0/0.26 l2transport
 pvc 26

l2vpn
 xconnect group frfr
 p2p p1
interface pos0/2/0/0.26
 neighbor y.y.y.y pw-id 1001

!QoS Policy
class-map matchdlci
 match frame-relay dlci 26

policy-map setde1
 class matchdlci
  set fr-de 1

interface pos0/2/0/0
 service-policy input setde1
```

PE2

```
interface pos0/3/0/0.26 l2transport
 pvc 26
```

```
l2vpn
 xconnect group frfr
 p2p p1
interface pos0/3/0/0.26
 neighbor x.x.x.x pw-id 1001
```

CE2

```
interface pos0/3/0/0.26
 pvc 26
 ipv4 add 10.0.0.2 255.0.0.0
```

Frame Relay - Ethernet Over Pseudowire: Example

This example shows that you can match based on the fr-de value on the ingress Frame Relay l2transport interface on router PE1 and set a specific MPLS EXP value. When the MPLS packet exits the PE1 core interface, this EXP value is set. When the packet exits router PE2 through the Ethernet l2transport interface, this value is part of the Ethernet packet CoS field.

This configuration allows you to carry over or map the QoS field from the Frame Relay network to the Ethernet network. This figure shows the network topology.

Figure 9: IP Interworking Over MPLS

CE1

```
interface pos0/2/0/0.26
 pvc 26
 ipv4 add 10.0.0.1 255.0.0.0
```

PE1

```
interface pos0/2/0/0.26 l2transport
 pvc 26

l2vpn
 xconnect group freth
 p2p p1
interface pos0/2/0/0.26
 neighbor y.y.y.y pw-id 1001
 interworking ipv4

!QoS Policy
class-map matchfrde
 match fr-de 1

policy-map setexp
 class matchfrde
  set mpls exp imposition 5

interface pos0/2/0/0.26 l2transport
```

```
 pvc 26
 service-policy input setexp
```

PE2

```
interface gig0/4/0/0.26 l2transport
 encapsulation dot1q 100

l2vpn
 xconnect group freth
 p2p p1
interface gig0/4/0/0.26
 neighbor x.x.x.x pw-id 1001
 interworking ipv4
```

CE2

```
interface gig0/4/0/0.26
 encapsulation dot1q 100
 ipv4 add 10.0.0.2 255.0.0.0
```

MLPPP QoS/MLFR QoS

Multilink provides a mechanism for aggregating multiple serial links into a bundle. Bundles support more bandwidth, load balancing between links, and improved service availability by protecting against single points of failure. The service allows users to increase bandwidth by aggregating multiple low speed links, which can be more cost-effective than upgrading to a single higher speed link. This provides a cost-effective solution for users requiring leased line service with bandwidth greater than T1 rates but below T3 rates.

Multilink interfaces can be configured with PPP encapsulation (MLPPP) or with Frame Relay encapsulation (MLFR). When a multilink interface is configured with Frame Relay encapsulation, subinterfaces can be configured below it.

The total bandwidth available for the multilink interface can change dynamically when links are added or removed to or from a multilink interface. The total bandwidth available can also change if the member links change state operationally to up or down, or by modifying the suspended condition of the policy. QoS policies applied on such interfaces need to be updated based on the bandwidth changes. In this case, one of the following actions is taken:

- Suspend the policy—Policy is suspended if the bandwidth requirements of the attached policy are more than the available bandwidth (which is reduced due to a member link going operationally down). Once the policy is suspended, any incoming or outgoing packets on that interface are not subject to QoS.

  A policy is suspended on ingress under these conditions:

  - In Enhanced Hierarchical Ingress Policing, when the sum of child police rates is greater than the parent police conform rate
  - Police peak rate is less than the police conform rate

    A policy is suspended on egress under these conditions:

  - Minimum bandwidth rate + priority class police rate is greater than the interface rate
  - Shape rate is less than the minimum bandwidth rate

- Priority class police conform rate is greater than the interface rate
- Priority class police peak rate is greater than the interface rate
- Police peak rate is less than the police conform rate

- Resume the policy—Policy is resumed if the bandwidth requirements of the attached policy are less than or equal to the available bandwidth, which increased due to a member link going operationally up. A suspended policy can also be resumed by modifying the suspended condition of the policy map without any change in the member link status.
- Update the policy—Active policy rates are updated to reflect the new available bandwidth. The available bandwidth could have increased or decreased, but the applied policy's bandwidth requirements can still be satisfied.

QoS statistics are not retained for the policy that transitions from an active state to a suspended state. If the policy is reactivated, all the previously collected statistics are lost and only the packets that pass through the interface after the reactivation are counted. The suspended policy can be modified to reduce its bandwidth requirements, so that it can be reactivated. A suspended policy can be modified while still attached to the interface.

Multiclass MLPPP with QoS

Multiclass Multilink Point-to-Point Protocol (MLPPP) can be used with QoS and configured using the **encap-sequence** command under a class in a policy map. The **encap-sequence** command specifies the MLPPP MCMP class ID for the packets in an MQC defined class.

The valid values for the **encap-sequence** ID number are **none**, 1, 2, or 3. The **none** value is applicable only when the **priority level** is 1 and indicates that there is no MLPPP encapsulation. The values 1, 2, or 3 can be used with priority 1 or 2 classes or other classes with queuing actions. An **encap-sequence** ID number of zero (0) is used by the system and is reserved for the default class; it cannot be specified in any other classes.

Note The **encap-sequence** ID numbers must be configured in numeric order. For example, you cannot assign an ID number of 3 unless you have already assigned 1 and 2.

The number of **encap-sequence** ID numbers must be less than the number of MLPPP classes that are negotiated between the peers via the multilink header. The user must ensure that the configuration is consistent as the system does not verify this.

The **ppp multilink multiclass remote apply** command provides a way to ensure this. You can ensure that the number of classes using an **encap-sequence** ID number (including the default of 0) is less than the min-numb*er* value in the **ppp multilink multiclass remote apply** command. For example, if the min-number value is 4, you can only have three or fewer classes with encap-sequence ID numbers.

The QoS policy validates the following conditions. If these conditions are not met, the policy is rejected:

- The **encap-sequence** ID number is within the allowed values of 1 to 3.
- When **encap-sequence** is configured for any class in a policy map, all classes in that policy map with **priority level 1** must also contain an **encap-sequence** ID number.
- The **encap-sequence none** configuration is restricted to classes with **priority level** 1.

- The class-default does not contain an **encap-sequence** configuration.
- Only classes containing a queuing action have the **encap-sequence** configuration.

Note Classes that share the same **encap-sequence** ID number must have the same priority.

A QoS policy map is configured as follows:

```
config
 policy-map type qos policy-name  class class-name   action   action   action
. . .
```

The following example shows how to configure a policy map for MLPPP:

```
config
 policy-map foo
  class ip-prec-1
   encap-sequence none
   police rate percent 10
   priority level 1
 !
  class ip-prec-2
   encap-sequence 1
   shape average percent 80
 !
  class ip-prec-3
   encap-sequence 1
   bandwidth percent 10
 !
  class class-default
 !
 end-policy-map
 !
```

MLPPP QoS/MLFR QoS: Example

Because a bundle interface dynamically changes its bandwidth as the member links go up or down, QoS policies applied on such interfaces need to be updated based on the bandwidth changes.

MPLS QoS

Note The introductory text and topology diagrams are taken from “MPLS Fundamentals,” Luc De Ghein, Copyright 2007, Cisco Systems, Inc.

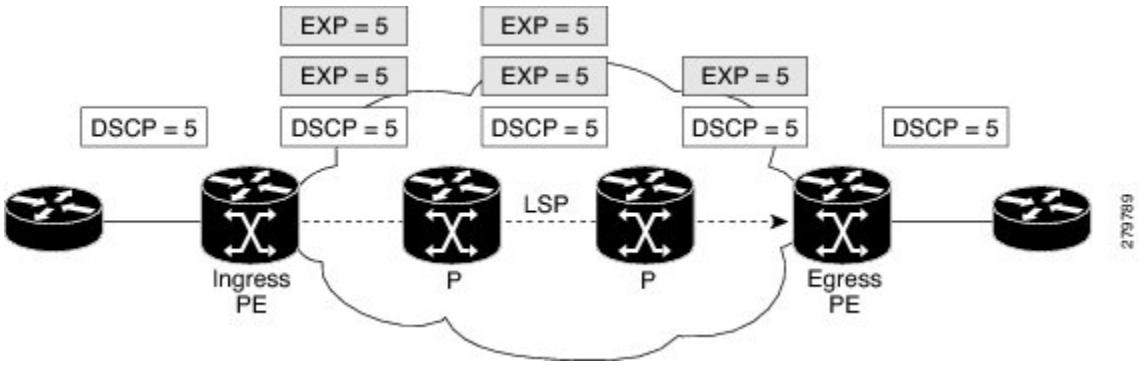
For MPLS QoS, there are three deployment scenarios based on tunneling model: uniform mode, pipe mode, and short pipe mode. Table 2 shows an overview of the tunneling models.

Tunneling Mode	IP-to-Label	Label-to-Label	Label-to-IP
Uniform	Copy IP precedence /DiffServ to MPLS EXP	MPLS EXP copied	Copy MPLS EXP to IP precedence/DiffServ
Pipe	MPLS EXP set according to service provider policy	MPLS EXP copied	Preserve IP precedence /DiffServ Forwarding treatment based on MPLS EXP
Short Pipe	MPLS EXP set according to service provider policy	MPLS EXP copied	Preserve IP precedence /DiffServ Forwarding treatment based on IP precedence/DiffServ

MPLS Uniform Mode

In uniform mode (as shown in following figure), there is only one DiffServ marking that is relevant for a packet when traversing the MPLS network. If the DiffServ marking of the packet is modified within the MPLS network, the updates information is the one considered meaningful at the egress of the LSP. Any changes to the packet marking within the MPLS network are permanent and get propagated when the packet leaves the MPLS network.

Figure 10: Uniform Mode

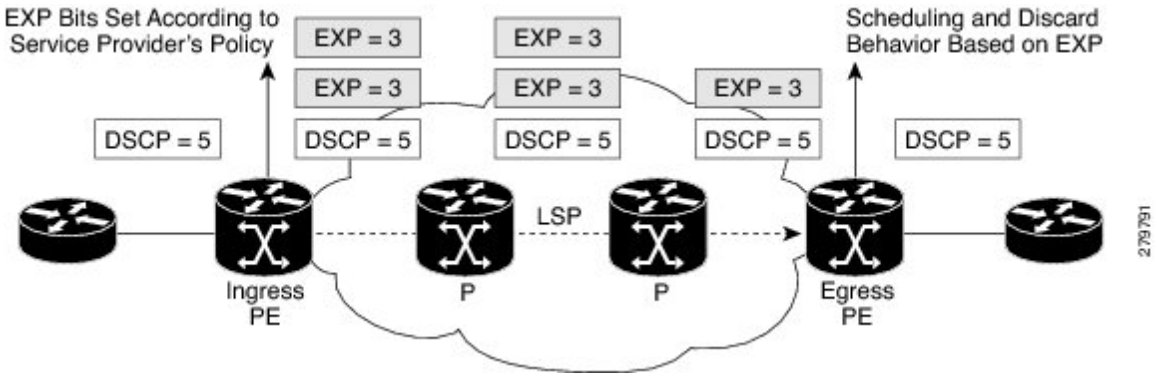


MPLS Pipe Mode

In pipe mode (as shown in the following figure), two markings are relevant for a packet when traversing the MPLS network. First, the marking used by intermediate nodes along the LSP span including the egress LSR. Second, the original marking carried by the packet before entering the MPLS network that will continue to be used once the packet leaves the MPLS network. Any changes to the packet marking within the MPLS network are not permanent and do not get propagated when the packet leaves the MPLS network.

Note that the egress LSR still uses the marking that was used by intermediate LSRs. However, the egress LSR has to remove all labels imposed on the original packet. In order to preserve this marking carried in the labels, the edge LSR keeps an internal copy of the marking before removing the labels. This internal copy is used to classify the packet on the outbound interface (facing the CE) once the labels are removed. This is usually achieved using the **set qos-group** and **match qos-group** commands.

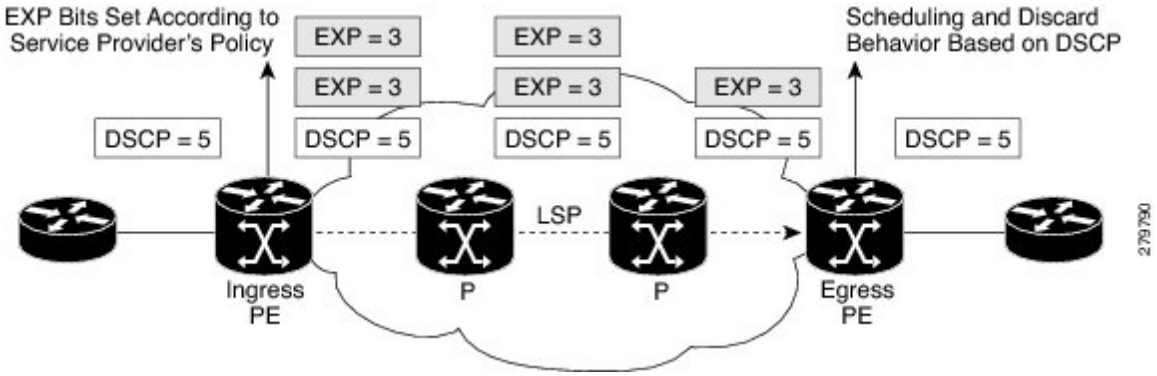
Figure 11: Pipe Mode



MPLS Short Pipe Mode

The short pipe mode, is a slight variation of the pipe mode. The only difference is that the egress LSR uses the original packet marking instead of using the marking used by the intermediate LSRs.

Figure 12: Short Pipe Mode



Uniform, Pipe, Short Pipe Modes: Ingress PE Example

This example shows how to implement the MPLS DiffServ and demonstrates the configuration needed on the ingress PE. Only precedence 4 is being matched. Precedence 4 is mapped to EXP bits value 4 by the policer, unless the bandwidth is exceeded, in which case the EXP bits are recolored to the value 2. The egress interface configuration is not needed for the MPLS DiffServ uniform model, but it is added to show how to perform QoS on the EXP bits.

```
!Ingress interface:
class-map prec4
match precedence 4
!
policy-map set-MPLS-PHB
class prec4
police rate 8000 kbps
conform-action set mpls experimental imposition 4
exceed-action set mpls experimental imposition 2
!
```

```
interface GigabitEthernet0/0/0/1
service-policy input set-MPLS-PHB

!Egress interface:
class-map exp2and4
match mpls experimental topmost 2 4
!
policy-map output-qos
class exp2and4
bandwidth percent 40
random-detect default
!
interface GigabitEthernet0/0/0/2
service-policy output output-qos
```

Uniform Mode: Egress PE Example

On the egress PE, the EXP bits are copied to the precedence bits using the **set qos-group** and **match qos-group** commands.

```
!Ingress interface:
class-map exp2
match mpls experimental topmost 2
!
class-map exp4
match mpls experimental topmost 4
!
policy-map policy2
class exp2
set qos-group 2
class exp4
set qos-group 4
!
interface GigabitEthernet0/0/0/2
service-policy input policy2

!Egress interface:
class-map qos2
match qos-group 2
class-map qos4
match qos-group 4
!
policy-map policy3
class qos2
set precedence 2
bandwidth percent 20
random-detect default
class qos4
set precedence 4
bandwidth percent 20
random-detect default
!
interface GigabitEthernet0/0/0/1
service-policy output policy3
```

Pipe Mode: Egress PE Example

This example shows the configuration of the egress PE for the MPLS DiffServ pipe mode. The egress LSR does not copy the EXP bits to the precedence bits of the outgoing IP packet. The scheduling of the packets

on the egress interface is done indirectly on the EXP bits using the **set qos-group** and **match qos-group** commands.

```
!Ingress interface:
class-map exp2
match mpls experimental topmost 2
!
class-map exp4
match mpls experimental topmost 4
!
policy-map policy2
class exp2
set qos-group 2
class exp4
set qos-group 4
!
interface GigabitEthernet0/0/0/2
service-policy input policy2

!Egress interface:
class-map qos2
match qos-group 2
class-map qos4
match qos-group 4
!
policy-map policy3
class qos2
bandwidth percent 20
random-detect default
class qos4
bandwidth percent 20
random-detect default
!
interface GigabitEthernet0/0/0/1
service-policy output policy3
```

Short Pipe Mode: Egress PE Example

This example shows the configuration of the egress PE for the MPLS DiffServ short pipe mode. The egress LSR forwards the packet based on the precedence or differentiated services code point (DSCP) bits of the IP packet after removing the labels. The egress LSR does not copy the EXP bits to the precedence bits of the outgoing IP packet.

```
! Configuration is not needed for ingress interface

!Egress interface:
class-map prec4
match precedence 4
!
policy-map policy3
class prec4
bandwidth percent 40
random-detect precedence 4 100 ms 200 ms
!
interface GigabitEthernet0/0/0/1
service-policy output policy3
```

QoS on Multicast VPN

QoS on Multicast VPN: Example

Supporting QoS in an mVPN-enabled network requires conditional and unconditional marking of the DSCP or precedence bits onto the tunnel header. Unconditional marking marks the DSCP or precedence tunnel as a policy action. Conditional marking marks the DSCP or precedence values on the tunnel header as a policer action (conform, exceed, or violate).

Unconditional Marking

```
class-map c1
  match vlan 1-10

policy-map p1
 class c1
  set precedence tunnel 3
```

Conditional Marking

```
policy-map p2
 class c1

  police rate percent 50
  conform action set dscp tunnel af11
  exceed action set dscp tunnel af12
```

SIP 700 for the ASR 9000

The **set precendence tunnel** and **set dscp tunnel** commands are not supported but general Multicast VPN is supported, as shown in the following example.

QoS on Multicast VPN: Example

In this example, there are three services offered across the network: mobile, enterprise, and other services. Mobile traffic is classified as broadband 2G mobile traffic and 3G mobile traffic.

Control traffic needs the highest priority and has priority level 1. Broadband 2G mobile traffic has priority level 2. A priority queue is associated with each of these traffic classes. Traffic in these classes is policed at a rate of 100 percent, which means that full line rate bandwidth is dedicated to these traffic classes.

Remaining bandwidth is distributed across the Mcast_BBTV_Traffic class, Enterprise_Traffic class, and Enterprise_Low_Traffic class.

```
policy-map CompanyA-Profile
 class Control_Traffic
  priority level 1
  police rate percent 100
  !
 !
 class BB_2GMobile_Traffic
  priority level 2
```

```
 police rate percent 100
 !
!
class Mcast_BBTV_Traffic
 bandwidth remaining ratio 1000
!
class 3GMobile_Traffic
 bandwidth remaining ratio 100
!
class Enterprise_Traffic
 bandwidth remaining ratio 10
!
class Enterprise_Low_Traffic
 bandwidth remaining ratio 1
!
class class-default
!
end-policy-map
```

QoS on NxDS0 Interfaces

For QoS on NxDS0 interfaces, the shape, police, and queuing minimum rate is 8 kbps and granularity is 1 kbps. When QoS is applied to a low speed NxDS0 link, frame relay fragmentation (frf12) configuration is also recommended in order to provide low delay for real-time priority traffic. The common configurations on NxDS0 interfaces are:

- One-level policy applied to a main interface without Frame Relay configured
- Two-level policy applied to a subinterface with Frame Relay configured

One-Level Policy Applied to Main Interface: Example

```
show run int Serial0/2/1/0/1/1:0

Mon Aug  9 11:29:50.721 UTC
interface Serial0/2/1/0/1/1:0
 service-policy output fractional_T1_E1_policy □--------policy applied to serial interface
 encapsulation frame-relay
!

RP/0/RSP1/CPU0:viking-1#show run policy-map
policy-map fractional_T1_E1_policy
 class Conversational
  priority level 1
  police rate 64 kbps
  !
 !
 class Streaming-Interactive
  bandwidth remaining percent 35
 !
 class Background
  bandwidth remaining percent 15
 !
 class TCP-traffic
  bandwidth remaining percent 10
 !
 class class-default
 bandwidth remaining percent 40
```

```
 !
 end-policy-map
```

Two-Level Policy Applied to a Subinterface: Example

```
show run int Serial0/2/1/0/1/1:0

Mon Aug  9 11:29:50.721 UTC
interface Serial0/2/1/0/1/1:0
 encapsulation frame-relay
 frame-relay intf-type dce
!

Mon Aug  9 11:29:37.150 UTC
interface Serial0/2/1/0/1/1:0.16 point-to-point
 ipv4 address 192.1.1.1 255.255.255.0
 pvc 16
 service-policy output parent_policy □--------policy applied to serial subinterface
  encap cisco
  fragment end-to-end 350  □-------------------frf12 enabled
  !
 !
!

show run policy-map

policy-map parent_policy
 class class-default
  shape average rate 768 kbps

show run policy-map

policy-map fractional_T1_E1_policy
 class Conversational
  priority level 1
  police rate 64 kbps
  !
 !
 class Streaming-Interactive
  bandwidth remaining percent 35
 !
 class Background
  bandwidth remaining percent 15
 !
 class TCP-traffic
  bandwidth remaining percent 10
 !
 class class-default
 bandwidth remaining percent 40
 !
 end-policy-map
```

VPLS and VPWS QoS

To support QoS on virtual private LAN service (VPLS)-enabled and virtual private wire service (VPWS)-enabled networks, packets can be classified based on these match criteria:

- Match on vpls broadcast (applicable to VPLS)

- Match on vpls multicast (applicable to VPLS)
- Match on vpls control (applicable to VPLS)
- Match on ethertype arp (applicable to both VPLS and VPWS)

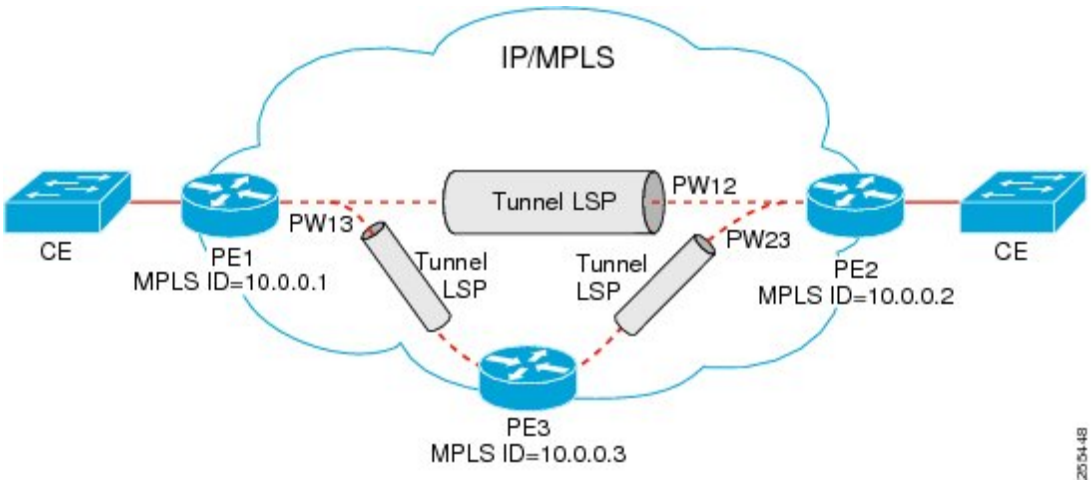
Note VPLS-specific and VPWS-specific classification are performed only in the ingress direction.

These guidelines apply to the VPLS and VPWS QoS feature:

- Supported on ingress Layer 2 bundle and nonbundle subinterfaces.
- Not supported on Layer 3 subinterfaces, but supported on ports with port inheritance policy. The system ignores VPLS classification on Layer 3 subinterfaces associated with the port.
- Match VPLS <control | multicast | broadcast> and match ethertype arp can be applied on a Layer 2 interface regardless of the Layer 2 service type, however VPLS <control | multicast | broadcast> classification is ignored on a non-VPLS Layer 2 interface type.

The following figure illustrates a typical VPLS topology. The VPLS network is a mesh of pseudowires (PWs) interconnected to bridge domains in the routers. Each of the provider edge (PE) routers has a bridge domain. Each PW is a bridge port into the bridge domain. The customer edge (CE) connection into each PE router is an attachment circuit (AC) bridge port into the same bridge domain. QoS configuration commands are applied to the AC that connects to the CE router on the one end and the bridge domain of the PE router on the other.

Figure 13: Typical VPLS Network Topology



VPLS and VPWS QoS: Example

This section contains a configuration example based on the components shown in VPLS and VPWS QoS, on page 220, and explains how the network matches packets based on the configured values.

The policy-map and PE-to-CE connection are configured as follows on the PE1 router:

```
class c1
  match vpls multicast
!
class c2
  match vpls broadcast
!
class c3
  match vpls control
!
class c4
  match ethertype arp
!
policy-map p1
  class c1
   set qos-group 3
   set mpls experimental imposition 4
   shape average percent 40
!
  class c2
   bandwidth remaining percent 10
   set mpls experimental imposition 5
!
  class c3
   police rate percent 10
   set mpls experimental imposition 6
!
  class c4
   bandwidth remaining percent 10
   set mpls experimental imposition 7
!
class class-default
!
end policy-map

interface GigabitEthernet0/2/0/0 l2transport
  description PE to CE connection
  service-policy input p1
!

l2vpn
 bridge group examples
  bridge-domain vpls-bridge
   interface GigabitEthernet0/2/0/0
   !
   vfi pe12link
    neighbor 10.0.0.2 pw-id 12
    !
   !
   vfi pe13link
    neighbor 10.0.0.3 pw-id 13
    !
   !
  !
 !
!
```

In the network designed and configured according to this example, and with VPLS and VPWS enabled, the packets that meet the match criteria receive QoS treatment according to the policy actions defined in the policy:

- If a VPLS multicast packet arrives on the ingress interface of the PE router, it matches class c1.

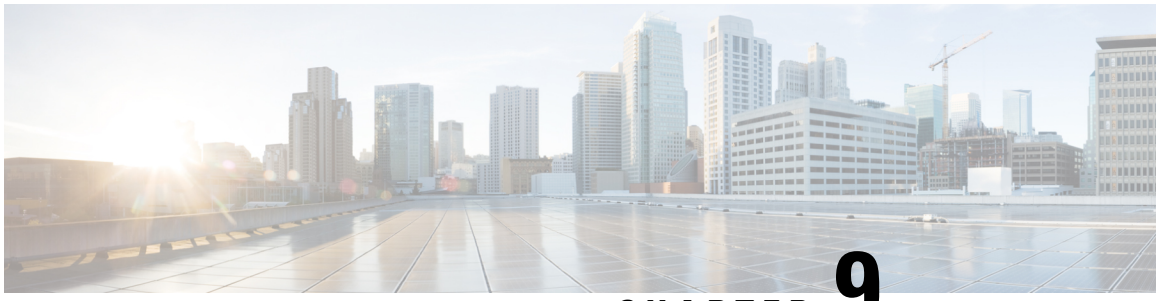
- If a VPLS broadcast packet arrives on the ingress interface of the PE router, it matches class c2.
- If a VPLS control packet arrives on the ingress interface of the PE router with MAC address ranging from 01-80-C2-00-00-00 to 01-80-C2-00-00-3F, it matches class c3.
- If an ARP packet arrives on the ingress interface of the PE router, it matches class c4.

Related Information

The information in this module focuses on the QoS implementation of features that are described in other technology guides. This table indicates the guides where you can find more information about these features.

Feature	Guide
802.1ad DEI	"Configuring Modular QoS Packet Classification and Marking" and "Configuring Modular QoS Congestion Management" in this guide
Frame Relay	*Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide* *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference*
IP Header Compression	*Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide* *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference*
L2VPN	*Cisco ASR 9000 Series Aggregation Services Router L2VPN and Ethernet Services Configuration Guide* *Cisco ASR 9000 Series Aggregation Services Router L2VPN and Ethernet Services Command Reference*
MLPPP/MLFR	*Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide* *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference*
MPLS	*Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide* *Cisco ASR 9000 Series Aggregation Services Router MPLS Command Reference*
QoS on Multicast VPN	*Cisco ASR 9000 Series Aggregation Services Router Multicast Configuration Guide* *Cisco ASR 9000 Series Aggregation Services Router Multicast Command Reference*
QoS on NxDS0 Interfaces	*Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide* *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference*

Related Information



CHAPTER 9

Configuring Hierarchical Modular QoS

Hierarchical QoS allows you to specify QoS behavior at multiple policy levels, which provides a high degree of granularity in traffic management.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
Enhanced Hierarchical Ingress Policing	no	yes
Hierarchical Policing	yes	yes
Hierarchical QoS	yes	yes
Three-Parameter Scheduler	yes	yes

Feature History for Hierarchical QoS on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.1	The Hierarchical Policing feature was introduced on Cisco ASR 9000 Series Routers on ASR 9000 Ethernet Line Cards. The Hierarchical QoS feature was introduced on Cisco ASR 9000 Series Routers on ASR 9000 Ethernet Line Cards. The Three-Parameter Scheduler feature was introduced on Cisco ASR 9000 Series Routers on ASR 9000 Ethernet Line Cards.
Release 3.9.0	The Hierarchical QoS feature was supported on the SIP 700 for the ASR 9000. (two-level policies only)

Release 4.0.0	The Enhanced Hierarchical Ingress Policing feature was introduced on Cisco ASR 9000 Series Routers on the SIP 700 for the ASR 9000. The Hierarchical Policing feature was supported on Cisco ASR 9000 Series Routers on the SIP 700 for the ASR 9000. For the Hierarchical QoS feature, support was added for three-level policies on the SIP 700 for the ASR 9000. The Three-Parameter Scheduler feature was supported on the SIP 700 for the ASR 9000.

How to Configure Hierarchical QoS

When configuring hierarchical QoS, consider the following guidelines:

- When defining polices, start at the bottom level of the hierarchy. For example, for a two-level hierarchical policy, define the bottom-level policy and then the top-level policy. For a three-level hierarchical policy, define the bottom-level policy, the middle-level policy, and then the top-level policy.
- Do not specify the input or output keyword in the service-policy command when configuring a bottom-level policy within a top-level policy.
- Configure bottom-level policies only in middle-level and top-level policies.
- When you attach an undefined policy as a child policy, a policy-map (with only class-default) is created.

Service Fragment on LACP

- Supports only physical and bundle interfaces. No support on BVI, Satellite, and BNG.
- All sub interface policys in a port with service-fragment policy must refer to one of the service fragments in port policy.
- You must perform removal of sub-interface policy before port policy.

Port policy configurations - Defining a service fragment

This configuration task explains how to define a service fragment in a port policy. The **service-fragment** command, in the policy map configuration mode helps define the service fragment.

Aspects need to be considered while defining a service-fragment are:

- All service fragment names must be unique in a port policy. However, same names can be reused across policies.
- A class in a port policy which defines a service fragment can only specify shape, BWRR (Budgeted Weighted Round Robin), and child policy actions. Only flat policies are supported at port level.

- In a 2-level policy, only a child policy can define service fragments. A parent policy can not define service fragments and should have one class with only shape actions.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-map-name*
3. **class** *class-name*
4. **service-fragment** *name*
5. **exit**
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Enters policy map class configuration mode. • Specifies the name of the class whose policy you want to create or change.
Step 4	**service-fragment** *name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)#` **`service-fragment s1`**	Defines a service-fragment. The defined service fragment (s1) will be referred to for the sub-interface policy configuration.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)#` **`exit`**	Returns the router to policy map configuration mode.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Configuring sub-interface policy

This configuration task explains configuring sub-interface policy using the **fragment** command. The **fragment** command refers to the previously configured service-fragment and has to be applied on the corresponding port.

Sub-interface policy limitations:

- Sub-interface policies need to refer to a service-fragment in the parent policy in a 2-level sub-interface policy.
- The sub-interface policy actions in a parent policy should not have shape, policy, bandwidth actions in percentages (only in absolute numbers).

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-map-name*
3. **class** *class-name*
4. **fragment** *name*
5. **exit**
6. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Enters policy map configuration mode. • Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name*	Enters policy map class configuration mode.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	• Specifies the name of the class whose policy you want to create or change.
Step 4	**fragment** *name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# fragment s1`	Refers to a previously defined service-fragment (here, s1 is the defined service-fragment).
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 6	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Applying a service fragment policy on a physical interface

To apply a qos policy on an interface, use the **service-fragment-parent** command. This can be used only after a service-fragment policy is defined on a port.

SUMMARY STEPS

1. **configure**
2. **interface** *interface-path-id*
3. **service-policy** { **input** | **output** | **type** } **service-fragment-parent**
4. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**interface** *interface-path-id* **Example:** `RP/0/RSP0/CPU0:router (config) # interface gig 0/1/0/22`	Specifies the interface for which the service-policy is being defined.
Step 3	**service-policy** { **input** | **output** | **type** } **service-fragment-parent** **Example:** `RP/0/RSP0/CPU0:router (config-if) # service-policy input s1 service-fragment-parent`	Applies the service policy on the defined service-fragment.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes, and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration mode, without committing the configuration changes.

Configuring the Three-Parameter Scheduler

When configuring the Three-Parameter Scheduler, consider the following guidelines:

- To use the three-parameter scheduler, a queueing class must be enabled. To enable a queueing class, you must configure at least one of the three parameters. When at least one parameter is configured, a queue is assigned to the class.
- If you configure only one parameter, the scheduler uses default values for the other two parameters.
- You can configure all 3 parameters in the same class.
- Minimum bandwidth must be less than maximum bandwidth.
- You can configure the three-parameter scheduler on the second generation of ASR 9000 Series Carrier Ethernet line cards and the third generation of ASR 9000 Series High Density Ethernet line cards.

ASR 9000 Ethernet Line Cards

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*

3. **class** *class-name*
4. **shape average** {**percent** *percentage* | *rate* [*units*]}
5. **exit**
6. **policy-map** *policy-name*
7. **class class-default**
8. **bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
9. **service-policy** *policy-map-name*
10. **end**
11. or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map bottom-child`	Creates or modifies the bottom-level policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class Bronze`	Assigns the traffic class that you specify to the policy map. Enters policy map class configuration mode.
Step 4	**shape average** {**percent** *percentage*	*rate* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average 1 mbps`	Shapes traffic to the indicated bit rate.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Exits policy map class configuration mode.
Step 6	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# policy-map Top-Parent`	Creates or modifies the top-level policy.

	Command or Action	Purpose
Step 7	**class class-default** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Configures or modifies the parent class-default class. **Note** • You can configure only the class-default class in a parent policy. Do not configure any other traffic class.

Step 8	**bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`
		Specifies the minimum bandwidth allocated to a class as a percentage of link bandwidth.
		Specifies how to allocate excess bandwidth to a class.
		Specifies maximum bandwidth as a percentage of link bandwidth (when other classes are not using all of their bandwidth share).
		Note
		• You must configure at least one of the three parameters.

Step 9	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy Bottom-Child`	Applies a bottom-level policy to the top-level class-default class.
Step 10	**end**	

Step 11	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

SIP 700 for the ASR 9000

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
5. **exit**
6. **policy-map** *policy-name*
7. **class class-default**
8. **shape average** {**percent** *percentage* | *rate* [*units*]}
9. **service-policy** *policy-map-name*
10. **end**
11. or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map bottom-child`	Creates or modifies the bottom-level policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class Bronze`	Assigns the traffic class that you specify to the policy map. Enters policy map class configuration mode.

Step 4	**bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*] **or shape average** {**percent** *percentage* | *rate* [*units*]}
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	`or`
		Specifies the minimum bandwidth allocated to a class as a percentage of link bandwidth.
		Specifies how to allocate excess bandwidth to a class.
		Specifies maximum bandwidth as a percentage of link bandwidth (when other classes are not using all of their bandwidth share).
		Note

	Command or Action	Purpose

	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# shape average percent 50`
		• You must configure at least one of the three parameters.

Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Exits policy map class configuration mode.
Step 6	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# policy-map Top-Parent`	Creates or modifies the top-level policy.
Step 7	**class class-default** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Configures or modifies the parent class-default class. **Note** • You can configure only the class-default class in a parent policy. Do not configure any other traffic class.
Step 8	**shape average** {**percent** *percentage* | *rate* [*units*]} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average 1 mbps`	(Optional) Shapes traffic to the indicated bit rate.
Step 9	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy Bottom-Child`	Applies a bottom-level policy to the top-level class-default class.
Step 10	**end**	

Step 11	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

	Command or Action	Purpose
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Attaching Hierarchical Policies to Physical and Virtual Links

To attach hierarchical policies to interfaces, subinterfaces, virtual circuits, and virtual LANs, use the **service-policy** {**input** | **output**} *policy-map-name* command.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **service-policy {input | output}** *policy-map-name*
4. **end**
5. or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface pos 0/2/0/0`	Specifies the interface to attach the hierarchical policy.
Step 3	**service-policy {input | output}** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy input All_Traffic`	Attaches the policy map you specify. • **input**—Apply the QoS policy to inbound packets. • **output**—Apply the QoS policy to outbound packets. • *policy-map-name*—Name of a previously configured top-level policy map
Step 4	**end**	

	Command or Action	Purpose

Step 5	or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Enhanced Hierarchical Ingress Policing

The difference between configuring enhanced hierarchical ingress policing and configuring hierarchical ingress policing is the addition of the child-conform-aware command.

When used in the parent policer, the child-conform-aware command prevents the parent policer from dropping any ingress traffic that conforms to the maximum rate specified in the child policer.

Restrictions

Enhanced Hierarchical Ingress Policing has the following limitations:

- Sum of all child policer rates cannot be greater than the parent policer rate.
- Single-rate two-color policer (color blind) only.
- Configurations that specify burst size in the **police rate** command are supported; configurations that specify peak burst become single-rate three-color policers and are therefore rejected.
- Configure the **child-conform-aware** command only in the parent policer.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy-name*
3. **class** *class-name*
4. **service-policy** *policy-map-name*
5. **police rate** {*value* [*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-rate** *value* [*units*]] [**peak-burst** *peak-burst* [*burst-units*]]

6. **child-conform-aware**
7. **conform-action** [**drop** | **set** *options* | **transmit]**
8. **exceed-action** [**drop** | **set** *options* | **transmit**]
9. **end** or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map parent`	Enters policy map configuration mode. Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Enters policy map class configuration mode. Specifies the name of the class whose policy you want to create or change.
Step 4	**service-policy** *policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy child`	Applies the bottom-level policy map to the parent class-default class. **Note** • Do not specify an input or output keyword.
Step 5	**police rate** {*value* [*units*] | **percent** *percentage*} [**burst** *burst-size* [*burst-units*]] [**peak-rate** *value* [*units*]] [**peak-burst** *peak-burst* [*burst-units*]] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate percent 50`	Configures traffic policing and enters policy map police configuration mode.
Step 6	**child-conform-aware** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# child-conform-aware`	Prevents the parent policer from dropping any ingress traffic that conforms to the maximum rate specified in a child policer.
Step 7	**conform-action** [**drop** | **set** *options* | **transmit]** **Example:**	Configures the action to take on packets that conform to the rate limit. The allowed action is:

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# conform-action transmit`	**transmit**—Transmits the packets.
Step 8	**exceed-action** [**drop** | **set** *options* | **transmit**] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exceed-action drop`	Configures the action to take on packets that exceed the rate limit. The allowed action is: **drop**—Drops the packet.

Step 9	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:
		Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Two-Level Hierarchical Queueing Policy: Example

The following example shows a two-level policy applied at the Multilink Frame Relay main interface. The same policy can be applied at Multilink PPP main interface.

```
class-map match-any video
 match precedence 1
 end-class-map
!
class-map match-any premium
 match precedence 2 3
 end-class-map
!
class-map match-any voice-ip
 match precedence 0
 end-class-map
!
class-map match-any best-effort
 match precedence 4
 end-class-map
```

```
policy-map parent_shape
 class class-default
  service-policy child_policy
  shape average percent 90
 !
 end-policy-map
!

policy-map child_policy
 class voice-ip
  priority level 1
  police rate percent 20
  !
 !
 class video
  bandwidth percent 40
 !
 class premium
  bandwidth percent 10
  random-detect precedence 2 10 ms 100 ms
  random-detect precedence 3 20 ms 200 ms
  queue-limit 200 ms
 !
 class best-effort
  bandwidth percent 20
  queue-limit 200 ms
 !
 class class-default
 !
 end-policy-map
!

interface Multilink0/2/1/0/1
 service-policy output parent_shape
 encapsulation frame-relay
 frame-relay intf-type dce
```

Three-Level Hierarchical Queueing Policy: Examples

Three-Level Hierarchical Queueing Policy: Examples

In this example, policy grand-parent is applied to the main Ethernet interface. The grand-parent policy limits all outbound traffic of the interface up to 500 Mbps. The parent policy has class vlan1 and vlan2, and traffic in vlan1 or vlan2 is limited to 40 percent of 500 Mbps. The policy child_policy classifies traffic based on different services and allocates bandwidth for each class accordingly.

```
class-map match-any video
 match precedence 1
 end-class-map
!
class-map match-any premium
 match precedence 2 3
 end-class-map
!
class-map match-any voice-ip
 match precedence 0
 end-class-map
!
class-map match-any best-effort
 match precedence 4
```

```
 end-class-map

class-map match-any vlan1
 match vlan 1
 end-class-map

class-map match-any vlan2
 match vlan 2
 end-class-map

policy-map grand-parent
class class-default
 shape average 500 Mbps
 service-policy parent
 !
 end-policy-map

policy-map parent
 class vlan1
  service-policy child_policy
  shape average percent 40
 !
 class vlan2
  service-policy child_policy
  shape average percent 40
 !
 end-policy-map
!

policy-map child_policy
 class voice-ip
  priority level 1
  police rate percent 20
  !
 !
 class video
  bandwidth percent 40
 !
 class premium
  bandwidth percent 10
  random-detect precedence 2 10 ms 100 ms
  random-detect precedence 3 20 ms 200 ms
  queue-limit 200 ms
 !
 class best-effort
  bandwidth percent 20
  queue-limit 200 ms
 !
 class class-default
 !
 end-policy-map

 interface GigabitEthernet0/0/0/9
 service-policy output grand-parent
```

SIP 700 for the ASR 9000

In this example, the policy parent_policy is applied to the Multilink Frame Relay main interface. The policy parent_policy has two classes, which match on Frame Relay DLCIs. The Multilink Frame Relay main interface has two Frame Relay PVCs configured (DLCI 16, DLCI 17).

```
interface Multilink0/2/1/0/1
```

```
 mtu 1504
 service-policy output parent_policy
 encapsulation frame-relay
 frame-relay intf-type dce
!

policy-map parent_policy
 class parentQ_1
  service-policy child_queuing_policy
  shape average 64 kbps
 !
 class parentQ_2
  service-policy child_queuing_policy
  shape average 1 mbps
 !
 class class-default
 !
 end-policy-map
!

class-map match-any parentQ_1 <----- class map parent class dlci=16
 match frame-relay dlci 16
 end-class-map
!

class-map match-any parentQ_2 <----- class map parent class dlci=17
 match frame-relay dlci 17
 end-class-map
!

interface Multilink0/2/1/0/1.16 point-to-point <------ dlci 16 pvc config
 ipv4 address 192.1.1.1 255.255.255.0
 pvc 16
  encap cisco
 !
!
interface Multilink0/2/1/0/1.17 point-to-point <------ dlci 17 pvc config
 ipv4 address 192.1.2.1 255.255.255.0
 pvc 17
  encap cisco
 !
!
policy-map child_queuing_policy <--------- child policy map
 class voice-ip
  priority level 1
  police rate percent 20
  !
 !
 class video
  bandwidth percent 40
 !
 class premium
  service-policy gchild_policy
  bandwidth percent 10
  random-detect discard-class 2 10 ms 100 ms
  random-detect discard-class 3 20 ms 200 ms
  queue-limit 200 ms
 !
 class best-effort
  bandwidth percent 20
  queue-limit 200 ms
 !
 class class-default
 !
```

```
 end-policy-map
!

policy-map gchild_policy <-------- grandchild policy map
 class premium_g1
  police rate percent 10
  !
  set discard-class 2
 !
 class premium_g2
  police rate percent 50
  !
  set discard-class 3
 !
 class class-default
 !
 end-policy-map
!

show run class-map  <----------- shows all class-map configs
Mon Aug  2 11:35:19.479 UTC
class-map match-any video
 match precedence 1
 end-class-map
!
class-map match-any premium
 match precedence 2 3
 end-class-map
!
class-map match-any voice-ip
 match precedence 0
 end-class-map
!
class-map match-any parentQ_1
 match frame-relay dlci 16
 end-class-map
!
class-map match-any parentQ_2
 match frame-relay dlci 17
 end-class-map
!
class-map match-any premium_g1
 match precedence 2
 end-class-map
!
class-map match-any premium_g2
 match precedence 3
 end-class-map
!
class-map match-any best-effort
 match precedence 4
 end-class-map
```

Three-Parameter Scheduler: Examples

Three-Parameter Scheduler: Examples

This example shows how to configure a three-parameter scheduler in a two-level hierarchical policy.

```
policy-map Bottom-ChildA
class A1
```

```
      shape average 400 kbps
class A2
     shape average 400 kbps

policy-map Bottom-ChildB
class B1
      shape average 250 kbps
class B2
     shape average 450 kbps

policy-map Top-Parent
 class parentA
  shape average 500 kbps
  bandwidth percent 30
  bandwidth remaining percent 80
 service-policy Bottom-ChildA
class parentB
  shape average 500 kbps
  bandwidth percent 60
  bandwidth remaining percent 10
 service-policy Bottom-ChildB
```

SIP 700 for the ASR 9000

This example shows how to configure a three-parameter scheduler in a two-level hierarchical policy.

```
policy-map Bottom-Child
 class A
   bandwidth percent 30
   bandwidth remaining percent 80
   shape average percent 50
 class B
   bandwidth percent 60
   bandwidth remaining percent 10
class class-default
exit

policy-map Top-Parent
 class-default
   shape average 1 mbps
 service-policy Bottom-Child
```

Hierarchical Policing: Examples

Hierarchical Policing: Examples

This example shows a two-level policy with police actions at each level. There are two classes in the top level, one for each customer. Aggregated traffic from each customer is subject to a rate limit as specified by the **police rate** command in the top level. Traffic in different classes in the bottom level is limited by an additional set of police actions to control different types of traffic for each customer.

```
class-map match-any customera
 match vlan 10-14
class-map match-any customerb
 match vlan 15-19
class-map match-any prec1
 match precedence 1
class-map match-any prec3
 match precedence 3
```

```
policy-map parent
 class customera
  service-policy childa
  bandwidth remaining ratio 10
  police rate percent 50
    conform-action transmit
    exceed-action drop
 class customerb
  service-policy childb
  bandwidth remaining ratio 100
  police rate percent 70
    conform-action transmit
    exceed-action drop

policy-map childa
 class prec1
  police rate percent 25
  conform-action transmit
  exceed-action drop
 class prec3
  police rate percent 25
   conform-action transmit
   exceed-action drop

policy-map childb
 class prec1
  police rate percent 30
  conform-action transmit
  exceed-action drop
 class prec3
  police rate percent 30
   conform-action transmit
   exceed-action drop
```

SIP 700 for the ASR 9000

In this example, policers are specified in the policy child in class Prec1 and class Prec3, and also in the class-default in the policy parent. The policers in the child policy, police traffic in class Prec1 at 30 percent (of 50 percent), police traffic in class Prec3 at 60 percent (of 50 percent) and police any other traffic at 10 percent (of 50 percent). Cumulatively, all traffic on the interface is policed at 50 percent of the interface rate by the policer in the parent policy.

```
class-map match-any prec1
    match precedence 1

class-map match-any prec3
    match precedence 3

policy-map parent
    class class-default
       service-policy child
           police rate percent 50
        conform-action transmit
        exceed-action drop
policy-map child
    class prec1
       police rate percent 30
           conform-action transmit
           exceed-action drop
    class prec3
       police rate percent 60
```

```
        conform-action transmit
        exceed-action drop
    class class-default
       police rate percent 10
        conform-action transmit
        exceed-action drop
```

Attaching Service Policies to Physical and Virtual Links: Examples

Physical Link: Example

In this example, the p1 policy is applied to a Gigabit Ethernet interface:

```
interface gigabitethernet 0/2/0/0
service-policy input p1
```

Virtual Link: Example

In this example, the p2 policy is applied to the private virtual circuit (PVC) under a multilink Frame Relay subinterface. A QoS policy can be applied only to a PVC under a Frame Relay subinterface; it cannot be applied directly to a Frame Relay subinterface.

```
interface Multilink0/2/1/0/1.16 point-to-point
 encapsulation frame-relay
 ipv4 address 192.1.1.1 255.255.255.0
 pvc 16
  service-policy output p2
  encap cisco
```

Service Fragment on LACP: Examples

The following example displays the service-fragment premium being created on LACP.

```
policy-map tsqos-port-policy
    class class-default
        shape 500 mbps
    class dscp1
        shape 1 Gbps
        service-fragment premium
    class dscp0
        shape average 100 mbps
        service-fragment sga
```

This example shows the service-fragment premium being referred (at the sub-interface):

```
policy-map tsqos-subif-policy-premium
class class-default
fragment premium
shape 20 mbps
bandwidth remaining ratio 20
service-policy subif-child
end-policy
exit
```

Service Fragment Configurations: Example

This example shows the service-fragment premium being created.

```
policy-map tsqos-port-policy
    class class-default
        shape 500 mbps
    class dscp1
        shape 1 Gbps
        service-fragment premium
        end-policy
      exit
```

This example shows the service-fragment premium being referred (at the sub-interface):

```
policy-map tsqos-subif-policy-premium
    class class-default
        fragment premium
        shape 20 mbps
        bandwidth remaining ratio 20
        service-policy subif-child
        end-policy
       exit
```

Enhanced Hierarchical Ingress Policing: Example

This example shows parent and child policies in which two classes are defined in the child policy. In class AF1, the exceed action is set to an action other than to drop traffic.

If the child-conform-aware command were not configured in the parent policy, the parent policer would drop traffic that matches the conform rate of the child policer but exceeds the conform rate of the parent policer.

When used in the parent policer, the child-conform-aware command prevents the parent policer from dropping any ingress traffic that conforms to the committed rate specified in the child policer.

In this example, class EF in the child policy is configured with a committed rate of 1 Mbps, a conform action and an exceed action. The traffic that is below 1 Mbps is presented to the parent policer with the MPLS EXP bit set to 4, and traffic that exceeds 1 Mbps is dropped.

Class AF1 in the child policy is configured with a committed rate of 1 Mbps, a conform action and an exceed action. The traffic that is below 1 Mbps is presented to the parent policer with the MPLS EXP bit set to 3, and traffic that exceeds 1 Mbps is presented to the parent policer with the MPLS EXP bit set to 2.

With this child policy configuration, the parent policer sees traffic from the child classes as exceeding its committed rate of 2 Mbps. Without the **child-conform-aware** command in the parent policer, the parent polices to 2 Mbps, which can result into dropping some conformed traffic from class EF in the child policy. When the **child-conform-aware** command is configured in the parent policer, the parent policer does not drop any traffic that conforms under the child policy.

```
policy-map parent
  class class-default
    service-policy child
    police rate 2 mbps
      child-conform-aware
      conform-action transmit
      exceed-action drop

policy-map child
  class EF
    police rate 1 mbps
```

```
    conform-action set mpls experimental imposition 4
    exceed-action drop
class AF1
  police rate 1 mbps
    conform-action set mpls experimental imposition 3
    exceed-action set mpls experimental imposition 2
```

Verifying the Configuration of Hierarchical Policies

To verify hierarchical policies, enter any of the following commands in privileged EXEC mode:

show policy-map interface	Displays policy configuration information for all classes configured for all service policies on the specified interface.
show qos interface	Displays QoS information for all classes in the service policy that is attached to the specified interface.
show running-config class-map	Displays the configuration of all class maps configured on the router.
show running-config policy-map	Displays the configuration of all policy maps configured on the router.
show running-config policy-map policy-map-name	Displays the configuration of all classes contained in the policy map you specify.

Additional References

The following sections provide references related to implementing Hierarchical QoS.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Gett Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Mod of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Ro of Cisco Cisco ASR 9000 Series Aggregation Services R Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

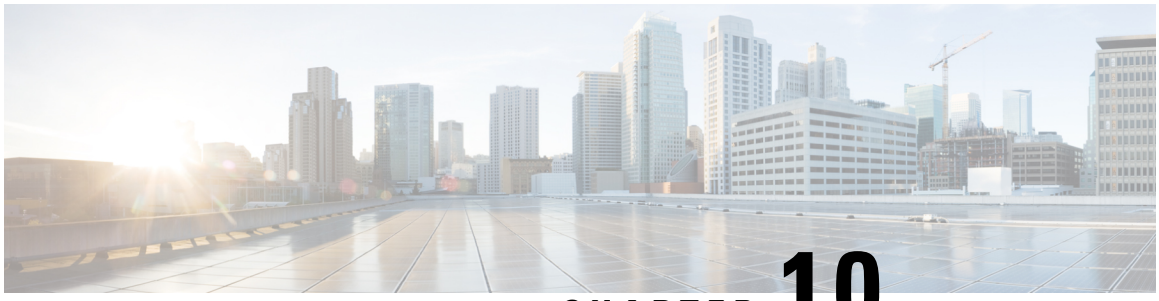
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, Cisco MIB Locator found at the following URL and choose a under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 10

Configuring Modular QoS on Link Bundles

A link bundle is a group of one or more ports that are aggregated together and treated as a single link. This module describes QoS on link bundles.

Line Card, SIP, and SPA Support

Feature	ASR 9000 Ethernet Line Cards	SIP 700 for the ASR 9000
QoS on Link Bundles	Yes	Yes

Feature History for Configuring Modular QoS on Link Bundles on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.9.0	The QoS on Link Bundles feature was introduced on ASR 9000 Ethernet Line Cards.
Release 6.0.1	The aggregate bundle QoS feature was introduced on ASR 9000 Ethernet Line Cards.

Link Bundling Overview

The Link Bundling feature allows you to group multiple point-to-point links together into one logical link and provide higher bidirectional bandwidth, redundancy, and load balancing between two routers. A virtual interface is assigned to the bundled link. The component links can be dynamically added and deleted from the virtual interface.

The virtual interface is treated as a single interface on which one can configure an IP address and other software features used by the link bundle. Packets sent to the link bundle are forwarded to one of the links in the bundle.

A link bundle is simply a group of ports that are bundled together and act as a single link. The advantages of link bundles are as follows:

- Multiple links can span several line cards to form a single interface. Thus, the failure of a single link does not cause a loss of connectivity.
- Bundled interfaces increase bandwidth availability, because traffic is forwarded over all available members of the bundle. Therefore, traffic can flow on the available links if one of the links within a bundle fails. Bandwidth can be added without interrupting packet flow..

All the individual links within a single bundle must be of the same type and the same speed.

Cisco IOS XR software supports these methods of forming bundles of Ethernet interfaces:

- IEEE 802.3ad—Standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. Links that are incompatible or have failed are automatically removed from a bundle.
- EtherChannel —Cisco proprietary technology that allows the user to configure links to join a bundle, but has no mechanisms to check whether the links in a bundle are compatible.

Load Balancing

Load balancing is supported on all links in the bundle. Load balancing function is a forwarding mechanism to distribute traffic over multiple links based on Layer 3 routing information in the router. There are two types of load balancing schemes:

- Per-Destination Load Balancing
- Per-Packet Load Balancing

When a traffic stream arrives at the router, per-packet load balancing allows the traffic to be evenly distributed among multiple equal cost links. Per-packet schemes make routing decision based on round-robin techniques, regardless of the individual source-destination hosts.

Only Per-Destination Load Balancing is supported.

Per-destination load balancing allows the router to distribute packets over one of the links in the bundle to achieve load sharing. The scheme is realized through a hash calculating based on the source-destination address and user sessions.

When the per-destination load balancing is enabled, all packets for a certain source-destination pair will go through the same link, though there are multiple links available. In other words, per-destination load balancing can ensure that packets for a certain source-destination pair could arrive in order.

Layer 3 Load Balancing on Link Bundles

By default, load balancing on Layer 2 link bundles is done based on the MAC SA/DA fields in the packet header. Layer 3 load balancing for link bundles is done on Ethernet Flow Points (EFPs) and is based on the IPv4 source and destination addresses in the packet.When Layer 3 service-specific load balancing is configured, all egressing bundles are load balanced based on the IPv4 source and destination addresses. When packets do not have IPv4 addresses, default load-balancing is used.

QoS and Link Bundling

All Quality of Service (QoS) features, currently supported on physical interfaces and subinterfaces, are also supported on all Link Bundle interfaces and subinterfaces. QoS is configured on Link Bundles in the same way that it is configured on individual interfaces. However, the following points should be noted:

- When a QoS policy is applied on a bundle (ingress or egress directions), the policy is applied at each member interface. Any queues and policers in the policy map (ingress or egress directions) will be replicated on each bundle member.
- If a QoS policy is not applied to a bundle interface or bundle VLAN, both the ingress and egress traffic will use the per link members port default queue.
- Link bundle members may appear across multiple Network Processing Units and linecards. The shape rate specified in the bundle policymap is not an aggregate for all bundle members. The shape rate applied to the bundle will depend on the load balancing of the links. For example, if a policy map with a shape rate of 10 Mbps is applied to a bundle with two member links, and if the traffic is always load-balanced to the same member link, then an overall rate of 10 Mbps will apply to the bundle. However, if the traffic is load-balanced evenly between the two links, the overall shape rate for the bundle will be 20 Mbps.

Example 1 shows how a traffic policy is applied on an Ethernet link bundle, in the ingress direction. The policy is applied to all interfaces that are members of the Ethernet link bundle.

Example 1 Applying a Traffic Policy to an Ethernet Link Bundle

```
interface Bundle-Ether bundle-id
 service-policy input policy-1
end
```

QoS for POS link bundling

For POS link bundles, percentage-based bandwidth is supported for policers and output queues. Time-based queue limit is supported for output queues.

Input QoS Policy setup

For input QoS, queuing is not supported and thus bandwidth is used for policer only. As a member link is added or removed from a bundle with input QoS configured, the aggregate bundle bandwidth for that affected line card will change. One input QoS policy instance is assigned for each SIP 700 line card that is part of the POS link bundle.

Output QoS Policy setup

When a member link is added to a bundle with output QoS configured, the policy-map of the bundle is applied to the member link.

Example 2 shows the output QoS policy supported on POS link bundles.

Example 2 : Output QoS policy supported on POS link bundles

```
policy-map out-sample
  class voice
    priority level 1
    police rate percent 10
  class premium
    bandwidth percent 30
    queue-limit 100 ms
  class class-default
    queue-limit 100 ms
```

Aggregate Bundle QoS Mode

Aggregated Bundle QoS allows the shape, bandwidth, police rates, and burst values to be distributed between the active members of a bundle where a QoS policy-map is applied. For instance, consider that the traffic is load-balanced among the members of the bundle. In aggregate mode, the bundle ethernet traffic is shaped to 10 Mbps to match the configuration of QoS policy.

When the policy is applied on a member of the bundle, a ratio can be calculated based on the total bandwidth of the bundle to that of the bandwidth of a member in the bundle. For example, if the bandwidth of the bundle is 20 Gbps, and the bandwidth of a member in the bundle is 10 Gbps, then the ratio will be 2:1.

A change in the bundle (with a member down, added, removed or activated) or mode results in the automatic recalculation of QoS rate.

The user QoS policy is invalid when applied to the bundle interface under the following scenarios:

- A 10 Gbps interface and 40 Gbps interface are part of a bundle, and the 40 Gbps interface is inactive. Currently, QoS policy is also programmed on non-active members. When programming the 40Gbps bundle member, the bundle bandwidth is 10 Gpbs, but the member bandwidth is 40 Gbps. The ratio of bundle bandwidth to member bandwidth does not work for this member.
- Consider a shape of 15 Gbps. This action is valid on bundles with multiple 10G active members, but invalid when only one member is active and that member is in QoS inconsistent state. To view inconsistency details for the QoS policy, run the **show qos inconsistency** command in EXEC mode. This scenario is also applicable during the reload of a router where only few interfaces in line cards (LC) becomes available before the rest of the interfaces in all LCs.
- A failure in the hardware when programming a rate change during the bundle bandwidth change or when a new member is added to the bundle.
- If an interface has the QoS policy configured to an absolute value, you cannot change the aggregated bundle mode from enabled to disabled. You must modify the policy or remove it before attempting to disable the aggregate bundle mode.
- An invalid policy combination, with the absolute values of port shaper less than the policy shape rate is accepted without an error in console or log file.
- You apply QoS policy on bundle interfaces for BNG subscribers. In this case, there is no aggregation across bundle members, and the policy is applied individually. For the policy to work correctly, modify the rate according to the number of members in the bundle.
- You apply QoS policy on PWHE and BVI interfaces. Both these interfaces don’t support bundle aggregate mode.

Load Balancing in Aggregate Bundle QoS

Load balancing requires a large number of flows in order to distribute the traffic among the members of the bundle. Ensure that load is balanced evenly among the members of the bundle before using the aggregate bundle QoS mode. If the under-lying traffic is only a few tunnels (GRE, TE-TUNNELS), it may be possible that the load balancing is not distributing the traffic evenly and may cause problems.

For example, consider bandwidth of a bundle is 20 Gbps, and the bandwidth of a member in the bundle is 10 Gbps. If the traffic is not load balanced, the aggregate traffic output may not reach 10 Gbps even when more than 10 Gbps is sent to the bundle-ether interface.

QoS Policy in Aggregate bundle mode

The following table shows the behavior of aggregate QoS policy mode to various actions:

Action	Behavior
Policing and Shaping / Bandwidth	**Percentage**: No change. The percentage is calculated bas max-rate / interface bandwidth **PPS / Absolute rate**: Divide the rate based on bandwidth **Burst-size**: If `time-units`, no change. Convert `time-uni` based on `service-rate` If configured in absolute value, divide the absolute value bandwidth ratio
Wred / Queue-Limit Threshold	**Time-Units**: No Change. Use the `service-rate` to conve **Absolute value**: Divide the absolute value based on band

Enabling Aggregate Bundle QoS

To enable the aggregate bundle QoS, perform these steps:

SUMMARY STEPS

1. **configure**
2. **hw-module all qos-modeaggregate-bundle-mode**
3. **class** *class-name*
4. **end** or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**hw-module all qos-modeaggregate-bundle-mode** **Example:** `RP/0/RSP0/CPU0:router(config)# hw-module all qos-mode aggregate-bundle-mode`	Enters policy map configuration mode. When aggregated bundle mode changes, QoS polices on bundle interfaces and sub-interfaces are modified automatically. A reload of the line card is not required.
Step 3	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Enters policy map class configuration mode. Specifies the name of the class whose policy you want to create or change.

Step 4	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# end`
	or
	`RP/0/RSP0/CPU0:router(config-pmap-c-police)# commit`	

Policing using Aggregate Bundle QoS

```
policy-map grand-parent
 class class-default
  service-policy parent
  police rate 200 mbps burst 1 kbytes
 end-policy-map

policy-map parent
 class class-default
  service-policy child
  police rate 300 mbps
 end-policy-map

policy-map child
 class 3play-voip
  police rate 10 mbps burst 10 kbytes peak-rate 20 mbps peak-burst 20 kbytes
   conform-color red-cos
   conform-action set precedence 1
   exceed-action set precedence 2
   violate-action drop

class 3play-video
  police rate 15 mbps burst 10 kbytes peak-rate 30 mbps peak-burst 20 kbytes
   conform-color yellow-cos
   conform-action set precedence 1
   exceed-action set precedence 2
   violate-action drop
  !
 !
```

```
 class 3play-premium
  police rate 25 mbps burst 10 kbytes peak-rate 35 mbps peak-burst 20 kbytes
   conform-color green-cos
   conform-action set precedence 1
   exceed-action set precedence 2
   violate-action drop
  !
 !
 class class-default
  police rate 6 mbps
  !
 !
 end-policy-map
!
```

Bundle Aggregate Policer

If a policy map having policer is applied on a bundle interface, the policy is programmed individually on the bundle members. This behaviour means that each bundle member has its own token bucket which results in inconsistencies. For example, if the policer is 1 Gbps and there are three bundle members, the presence of three token buckets allow for 3 Gbps traffic.

Configuring the **hw-module all qos-mode bundle-agg policer** command shares a policer token bucket between members of the bundle interface in the same network processor. In the preceding example, the three bundle members share 1 Gbps policer in the same network processor.

You must reload the line card after configuring this command. Because this operation is in the global configuration mode, all policers on the line card are configured in the aggregate mode.

In bundle aggregate policer mode, a policy replicates across every network processor. In the normal mode, the policy replicates across every member.

Restrictions

Ensure that you read these points before you configure the bundle aggregate policer.

- For policer configured with percent option, the policer percentage is calculated on the bundle member's bandwidth which gets added to the bundle first, and the policer is shared between all the bundle members in the same network processor. This scenario is applicable for mixed-speed bundles as well.

Note From Cisco IOS XR Release 7.5.1 onwards, configuring a percent policer value on a class allocates separate token buckets for each bundle member, which was the default behavior when **hw-module all qos-mode bundle-agg policer** wasn't enabled. See Separate Token Buckets for Percentage Policers , on page 257 for details.

  Policer percentage isn’t calculated based on the overall aggregate bandwidth of bundle members in the same network processor.

- To remove a bundle member, ensure that you first remove all members from that network processor and then add back the required members to the processor. If you remove only one member or don't remove all members from the network processor, the policer may not work properly.

Enabling Bundle Aggregate Policer

To enable the bundle aggregate policer, perform the following steps:

SUMMARY STEPS

1. **configure**
2. **hw-module all qos-mode bundle-agg-policer**
3. **end** or **commit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**hw-module all qos-mode bundle-agg-policer** **Example:** `RP/0/RSP0/CPU0:router(config)# hw-module all qos-mode bundle-agg-policer`	You must reload the line card.

Step 3	**end** or **commit**
	Example:
	`RP/0/RSP0/CPU0:router(config)# end`
	Or
	`RP/0/RSP0/CPU0:router(config)# commit`	

Separate Token Buckets for Percentage Policers

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Separate Token Buckets for Percentage Policers	Release 7.5.1	From this release, configuring a percent policer value on a class allocates separate token buckets for each bundle member. In other words, the policer percentage value is calculated on each single bundle member's bandwidth after configuring **hw-module all qos-mode bundle-agg policer**. This functionality allows you to use both aggregated absolute policers and nonaggregated percent policers. Plus, the percent policers span the entire bundle bandwidth (from 0 through 100%), allowing for better and efficient control over traffic flow. In earlier releases, the policer percentage for the entire bundle was calculated based on the bundle member's bandwidth that you added first. Such a calculation dragged down the traffic rate for the entire bundle.

Overview and Benefits

In earlier releases, configuring the **hw-module all qos-mode bundle-agg policer** command shared a policer token bucket between members of the bundle interface in the same network processor. However, all policers on the line card were configured in the aggregate mode. For percent policers, the percentage was calculated based on the bandwidth of the first bundle member and not on the aggregated bandwidth. Such a calculation dragged down the traffic rate for the entire bundle.

You could have scenarios where you require configuring policers with absolute aggregate values and nonaggregated percent values. For example:

- Configuring ingress policers on your customer subinterfaces, where the policers must have absolute values and aggregated on bundles, because you don't control ingress load-balancing.
- Configuring egress policers on main interfaces, where they can be percent values and not be aggregated because you control egress load-balancing.

From Cisco IOS XR Release 7.5.1, running the **hw-module all qos-mode bundle-agg policer** command allocates separate token buckets for each bundle member for percent policers, enabling their calculation across the entire bandwidth of the bundle.

When you configure **hw-module all qos-mode bundle-agg policer** and reload the line card:

- For classes that have absolute policer values configured: token bucket is shared between bundle members on the same network processor.
- For classes that have percent policer values configured: separate token bucket is allocated to each bundle member irrespective of the network processor they're on. This allocation is the default behavior when you don't enable **hw-module all qos-mode bundle-agg policer**.

This enhancement provides the following benefits:

- The ability for percent policers to span the whole bundle bandwidth (from 0% through 100%), thus allowing better and efficient control over traffic flow.
- The flexibility to configure policers with absolute values (aggregated) and percent values (nonaggregated). As an example, the following sample policy has class default with policer percent and other two classes with policer absolute rate values.

```
policy-map p_bundle
 class c_policer_2r3c
  police rate 10 mbps peak-rate 20 mbps
   conform-action transmit
   exceed-action set cos 4
   violate-action drop
  !
 !
 class c_policer_rate1
  police rate 5 gbps
  !
 !
 class class-default
  police rate percent 10
  !
 !
 end-policy-map
!
```

Guidelines

- The feature is supported on ingress and egress service policies.
- It is supported on satellite bundle interfaces.
- This feature supports in-place modification of a service policy.
- User Based Rate-Limiting (UBRL) isn't supported with this functionality.

Additional References

These sections provide references related to implementing QoS on Link Bundles.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Gett Guide*
Link Bundling	"Configuring Link Bundling on the Cisco ASR 9000 Seri module of *Cisco ASR 9000 Series Aggregation Services Ro and Hardware Component Configuration Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Mod of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Ro of Cisco Cisco ASR 9000 Series Aggregation Services Re Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

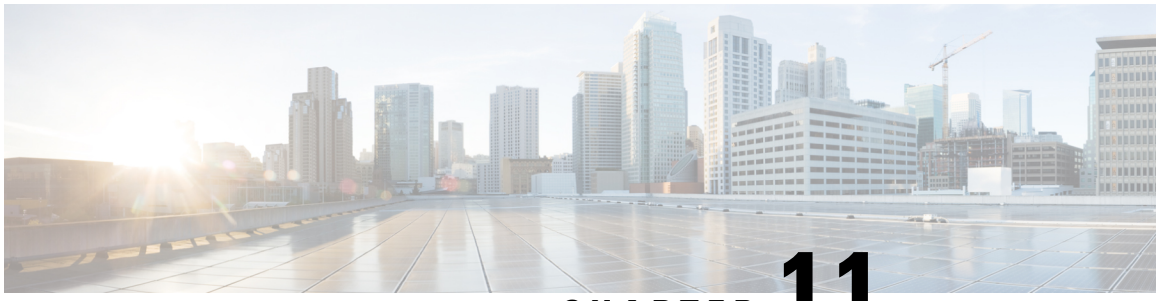
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 11

Configuring Flow Aware QoS

Flow Aware QoS provides packet flow awareness and enhances per-flow action capabilities in the existing QoS functionality. Flow aware QoS suite provides a framework that can support per-flow feature functionality such as admission control and traffic flow based dynamic rate limiting.

This module provides the conceptual and configuration information for Flow Aware QoS.

Feature History for Configuring Flow Aware QoS on Cisco ASR 9000 Series Routers

Release	Modification
Release 5.1.1	Flow Aware QoS feature was introduced.
Release 6.1.2	UBRL Policer Scale Information for ASR 9000 High Density 100GE Ethernet LCs on Cisco IOS XR

Information About Flow Aware QoS

Flow Aware QoS

In Cisco ASR 9000 Series Routers, the granular control of traffic flow is achieved by applying static match criteria and associated QoS action on traffic flow. With real-time on-demand VoIP and video traffic applications, and tailor-made user services, there is an increasing need for the QoS actions to be more flow, application and session aware as opposed to being static, configuration based and stateless. Flow aware QoS feature provides this functionality to QoS and creates a framework to define flow aware QoS solutions such as call admission control or per-user traffic rate limiting.

The Flow aware QoS feature enables QoS actions to be applied at a flow level. The flows are detected or learnt dynamically on a per-class, per-interface, per-direction level and the QoS action or decisions are applied on a per-flow basis guided by a QoS policy applied on the interface. The framework also provides an option to enforce admission control on the incoming traffic to preemptively prevent congestion.

The Flow aware QoS feature suite provides:

- User-defined flow definition—You can define a flow from a flexible choice of flow tuples (srcip, dstip, L4 protocol, sport, dport)
- Configurable flow bandwidth to decide how many video flows to allow—You can configure the flow bandwidth to decide how many video calls/flows to allow pass through a system without causing congestion.
- Redirection of non-admitted flows to default queue—You can redirect all the best-effort delivery traffic flows that exceed a predetermined admissible bandwidth to a default queue thereby providing guaranteed service on a per-flow basis.
- Configurable flow entry idle-timeout to tune as per use case or traffic profile— There are configurable flow age timeouts based on the traffic profile. You can set a timeout and ensure service fairness.

Flow Aware QoS Key Terms

This section lists the key terms of the Flow Aware QoS feature:

- Flow—A specific traffic pattern of the packet identified by unique source IP address (src-ip) or destination IP address (dst-ip) or 5-tuple parameters.
- Flow Tuple—The individual fields that define a flow is known as flow tuple.
- Flow Mask—A list of flow tuples defining a unique flow on a per-class basis is called as a flow mask. The flow tuples that define a flow can be configured at a per-class level.
- Flow Table—A table of flow records that are recorded as per the flow mask is a flow table. It is also referred to the flow table cache.
- Flow Age—The expiry time set in the flow cache to purge out stale flow records so that the new flows are learnt into the cache before the maximum limit is hit is called the flow age. Flow Age is also called as Idle Timeout.
- Flow Action—The QoS action that requires per-flow resource allocation is known as flow action.
- Micro-Flow policer—A QoS policer acting on a single traffic flow is known as micro-flow policer.
- Video CAC—The call admission control (CAC) functionality customized for video streams with capabilities to admit or reject individual traffic flows at a per-user or per-application level is known as Video CAC. Video CAC is also known as Video Q or flow aware CAC.
- CAC Reject—A CAC action variant in which packets from all unadmitted flows are dropped.
- CAC Redirect—A CAC action variant in which packets from all unadmitted flows are directed to a different child class. The QoS action for the redirected packets depends on the configuration of the "redirect" class.
- Aggregate action—Aggregate action could either be a regular QoS action such as mark or set, which is enforced on each flow, but is common to all flows or an aggregate parent policer / queuing action enforced on all flows.
- Catch-all Policer—The police action configured in a micro-flow policer class is to be applied on each of the flows. When the flows are being learnt or when the flow table is exhausted, all the packets are subjected to an aggregate policer called the "catch-all policer". The value of the catch-all policer is 100 Gbps and is not configurable.

- CAC Rate—The user configurable total bandwidth for CAC admitted flows. It should be equal to or less than class service rate.

Variants of Flow Aware QoS

Two major feature variants of Flow Aware QoS supported in Cisco IOS XR Release 5.1.1 on Cisco ASR 9000 Series Routers are:

- Call Admission Control (CAC)

  This variant is also known as Video Q or Flow aware CAC.

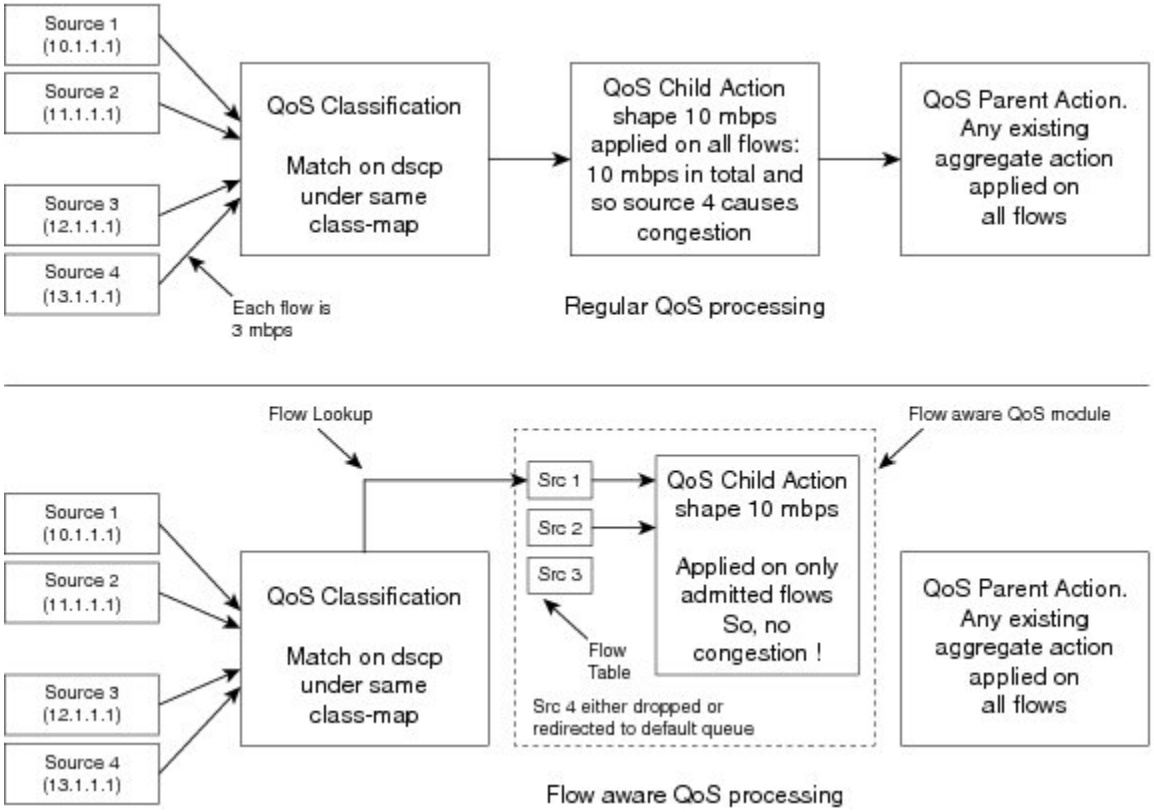
- User-based Rate Limiting (UBRL)

  This variant is also known as Micro flow policer or Flow aware policer.

Difference between Regular QoS and Flow Aware CAC

Figure 1 depicts the difference in the packet path between a regular QoS process and Flow Aware CAC.

Figure 14: Regular QoS vs Flow Aware CAC



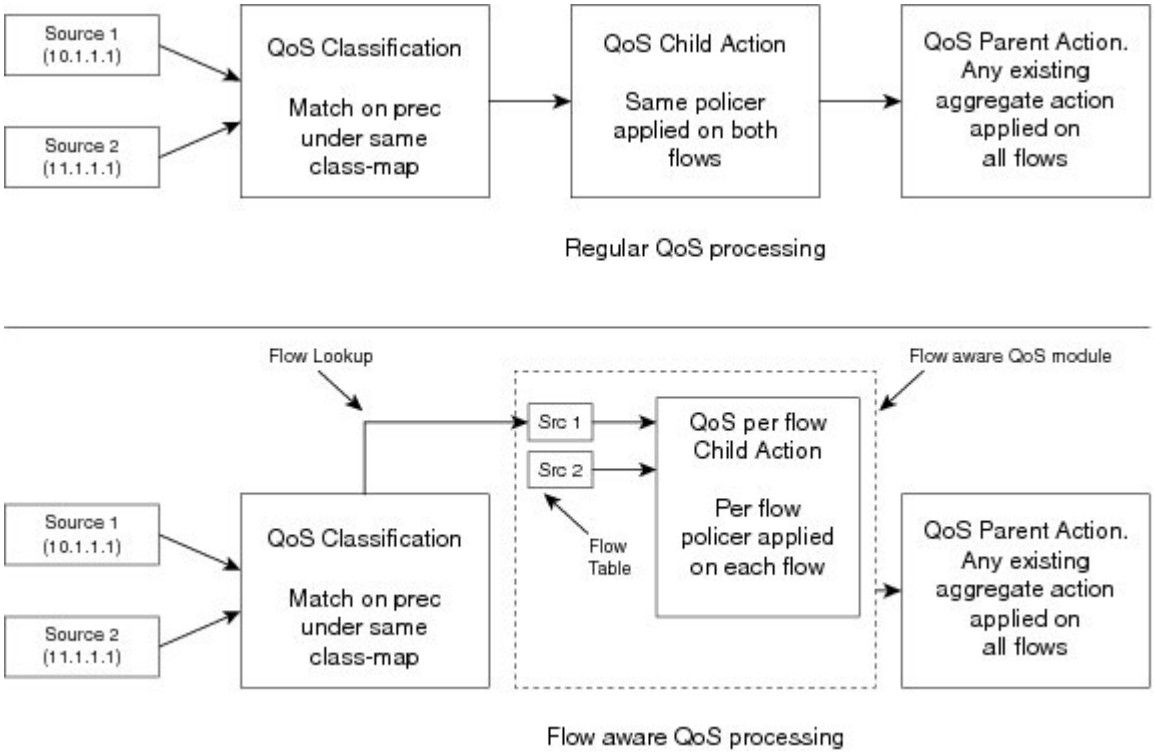
Let us assume there are 4 sources—Source 1, 2, 3, 4—with a QoS child Shape action at 10 mbps applied on all flows. If each source sends out a flow at 3 mbps, then, in the regular QoS processing, the source 4 causes congestion leading to random drop in the flow quality. However, in the Flow Aware CAC processing, where the Shape action is configured as 10 mbps, only three sources are admitted and source 4 is either dropped or

redirected to a default queue. Thus, the QoS Shape action is applied only to the 3 flows that were admitted, and as a result, there is no congestion.

Difference between Regular QoS and Flow Aware Policer or UBRL

Figure 2 depicts the difference in the packet path between a regular QoS process and Flow Aware policer or UBRL.

Figure 15: Regular QoS vs Flow Aware Policer or UBRL



Let us assume there are two sources—Source 1 and Source 2—with a QoS child action policer at 30 mbps. In the regular QoS processing, both the flows are policed at 30 mbps total. In the Flow Aware QoS processing, after the QoS classification, the flow is classified into two different flows based on the source IP. Thus, each flow is policed at 30 mbps.

Flow Aware CAC

When voice and video applications are connected over an interface, which has limited bandwidth, there is a drop in the flow quality. This is because the interface can fit N number of flows without quality degradation. The new N+1 flow affects the quality. There are no well-defined controls to restrict flows over an interface. Therefore, when a new flow is admitted, there is degradation in the flow quality of the flows already admitted.

To limit new flows, in order to protect existing flows, QoS provides Call Admission Control (CAC) feature. CAC dynamically learns traffic flows and admits until a predetermined configured bandwidth is available, thereafter flows are either dropped or redirected. CAC limits the flows in to an interface and ensures that already admitted flows are protected from congestion and random tail drops.

CAC Action Variations

CAC (Call Admission Control) feature controls the number of flows admitted per class. This is based on a count derived using the CAC rate and flow rate programmed in the policy under the "admit cac local" sub-mode. The action performed when the CAC feature is triggered is called the CAC action. There are two types of CAC actions:

CAC Reject

The number of flows that are admitted per class is derived based on the rate or flow-rate configuration. Only the specified number of flows is admitted and the remaining flows are dropped. Thus, in the CAC reject action, the packets from all the unadmitted flows are either dropped.

CAC Redirect

In the CAC redirect action, once the specified number of flows are admitted, the remaining flows are redirected to a different child class. The flows get redirected based on the configuration of the "redirect" or "unadmit" class.

Note The flow is always admitted in the admit class, and then, gets redirected to the other class at the child level.

Scale Information for CAC

The Flow Aware CAC feature is only supported on ASR 9000 Enhanced Ethernet line cards (LCs). Following are the scale information for CAC:

- Up to 64000 unique flow entries are supported for SE (Service Optimized) and 4000 for TR (Transport Optimized) version of the LCs for Cisco ASR 9000 Series Routers.
- Cisco ASR 9001 also supports the same scale as supported by Cisco ASR 9000 Series Routers.
- Each class supports a maximum of 16000 unique flows and up to 4000 such unique class-maps per NP.
- The scale is configurable per LC.

Note Full scale is not achieved for a configured scale size due to hardware resource recycling restrictions. The final scale may vary between M (maximum size) and M - 64, depending on internal hardware resource recycle rate and incoming flow fluctuations.

Restrictions

- CAC does not support 5-tuple flows with IPv6 traffic due to address length constraints.
- CAC is not supported on L2 forwarding interfaces.
- CAC is not supported for Pseudowire Headend (PWHE), Bridge Virtual Interface (BVI), Broadband Network Gateway (BNG) subscriber interfaces, cluster inter rack link (IRL) and satellite interfaces.
- CAC does not support user-specified tuple. It uses a 5-tuple flow mask by default.
- CAC Redirect always requires 2-level policies with only 2 classes at the child-level.

- The policer action is not supported on the leaf CAC class. Note: A leaf class is class that has no sub-classes or child classes.
- CAC actions are supported only at the leaf level.
- The CAC submode for a redirect action can only be at a parent level.
- For CAC Redirect action, the child classes support only CAC admit or unadmit match criteria.
- CAC does not support **flow idle-timeout none**.
- Dynamic enforcement of CAC bandwidth based on incoming flow rate sampling is not supported. Only static values derived from configured CAC bandwidth and per-flow rate will be used to derived an admissible flow count
- CAC supports only IPv4 unicast traffic topology. IPv6 transport and IP multicast traffic is not supported.
- CAC supports only L3 (routed) interfaces. CAC does not support L2 and MPLS interfaces or transport types.
- For bundle interfaces (port channel), flows are learnt and CAC actions are applied per-member and not on aggregate traffic across all the members.
- CAC does not provide information on admitted and rejected flows.
- Applying more than 64 flow aware policy instances to a line card is possible. However, removal of more than 64 flow aware policy instances simultaneously during configuration replacement, reverting to the previous configuration, saving multiple configurations, and so on, can lock the console for long durations and cause unintentional timeouts in various operations.
- CAC and policy based forwarding (PBF) features do not work together on the same interface or direction.
- CAC with redirect action and ACL based forwarding (ABF) do not work together on the same interface or direction.
- CAC allows first few packets from unadmitted flows even after hitting the max flow count due to the time taken for the programming of QoS in hardware.
- No new Management Information Base (MIB) support for CAC statistics and drop counters.
- CAC supports only plain IPv4 unicast traffic type. However, if unsupported traffic types match the CAC admit class, even though they are never learnt as admitted flows, would still get QoS processed and hit the CAC admit queue.
- CAC redirection is improper when a node processor's flow table scale exceeds the scale of the CAC counter, for each class . Some symptoms include random packet drops of the unadmitted flows and incorrect **show policy-map stats** output.
- Flow idle-timeout has a 10s granularity. Hence, the actual purge of a specific flow entry could be off by another 10s.
- For 5-tuple key with unknown (non TCP and UDP) protocol, CAC degrades 5-tuple key to a 3 tuple key usage (src-ip + dst-ip + protocol number).
- Flows are learnt and per-flow resources allocated by the feature even when the packets in the flow are dropped by features that get applied after QoS or by fabric and egress card.
- For 5 tuple flow mask and IPv4 fragment traffic flows, the first fragment would be learnt with the correct L4 details. For the subsequent fragments the flow entry will not have the L4 port details and gets degraded

to 3 tuple. This can cause oversubscription due to two policers allocated (one per flow) or congestion for fragmented flows when many fragmented streams between the same IP peers match the same second flow record.

- Ingress marking does not work on the packets that the router can’t forward such as time to live (TTL) packets. QoS policy is matched and show policy-map counters increment correctly. But the packets post punt and inject on transmission don't have the remarked precedence to differentiated services code point (DSCP).

User Based Rate-Limiting (UBRL)

A microflow policer applies a rate-limiting policy on a per-flow basis. User-Based Rate-Limiting (UBRL) is a microflow policer that dynamically learns traffic flows and rate-limit each unique traffic flow to an individual rate on per-flow basis. Unlike a normal microflow policer, UBRL allows a policer to be applied to all traffic to or from a specific user. The UBRL feature is a microflow policer with a source-mask or a destination mask that defines or classifies a user distinctly.

UBRL ensures that a single flow does not lack bandwidth and every customer gets a rate limited guarantee of flows. UBRL also provides enhanced granularity to provide SLA solutions by grouping different customer flows in different class-based user groups. UBRL helps manage traffic based on the offered SLA for customers in a high density aggregation environment.

UBRL Scenarios

This section describes the various UBRL scenarios.

UBRL for Multiple Sources

In this scenario, there is traffic from many customers on the interface. This is a common scenario in internet service provider (ISP) handoffs, where an ISP has customer traffics from multiple sources and a host provider receives traffic from these multiple sources.

Let us assume that each customer has been assigned a unique IP address and has the network credentials and requirements as shown in this table, and the flow-key is configured based on the source IP (src-ip).

Customer Name	Source IP Address	Requested Bandwidth
Company A	180.1.127.1	20 Mb
Company B	120.12.111.2	7 Mb
Company C	140.3.202.3	2 Mb

This scenario behaves differently depending on the policing requirement. If a same policing is applied, then the maximum rate of traffic sent from each customer is controlled to the same rate. In this case, the flows from each source are rate-limited based on the source-IP flow mask, which limits flows from a given customer to the same rate.

If a different policing is applied, then the maximum rate of traffic sent from each customer is controlled to a different rate. In this case, the flows from each source are rate-limited based on the source only flow mask ensuring that all traffic originating from each customer is treated as a single flow.

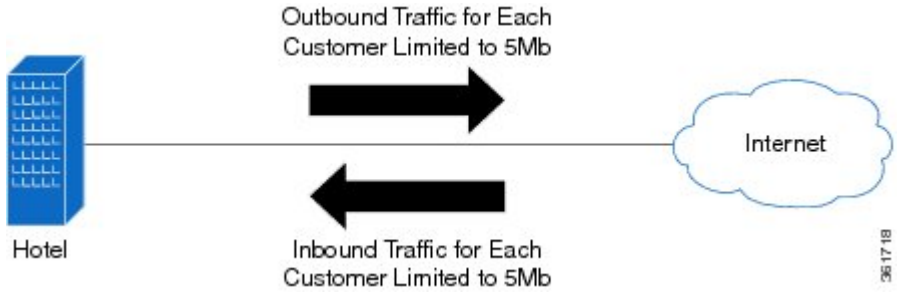
Bidirectional UBRL

Bidirectional UBRL applies the QoS policy in the input as well as in the output direction of the interface. Bidirectional UBRL allows different policies to applied in the input as well as output direction and these are not dependent on each other.

Bidirectional UBRL ensures that the traffic going out of a site is limited on a per user basis and the traffic coming in is also limited on a per user basis. Thus, bidirectional UBRL limits traffic flowing out of a customer site and traffic coming into the customer site, both on a per user basis or per flow basis, which is based on the configured flow-key.

Let us assume an example of Hotel that wants to restrict unwanted or lesser priority traffic coming in from Internet on a per user basis.

Figure 16: Bidirectional UBRL scenario



In this example, two flow masks are combined to limit traffic to and from users in the hotel. Let us assume that each user is limited to upload or download no more than 5Mb of data. To limit traffic to and from the users, two separate policers are configured, one on the inbound and the other on the outbound direction. Each policer uses a different flow mask to match traffic on the inward or outward direction. For outbound traffic, the policer uses a source-only flow mask to match on originating traffic. Every unique user is limited to 5Mb of upstream bandwidth. The return traffic matching on the inbound policer uses the destination only IP flow mask. This matching is applied on the users address, thus, limiting the download bandwidth to also 5Mb.

Egress UBRL

In cases where the traffic sent out of the egress direction of an interface needs to be rate limited on a per user basis, the UBRL feature is deployed at the CPE. This is known as egress UBRL where the customer regulates traffic being sent to the provider. In this scenario, the UBRL is applied at the outward direction of the interface. Egress UBRL is required for aggregate traffic where many input interfaces converge at the service or WAN edge and get routed out of an interface connecting to the provider.

UBRL for Multiple Destination

In this scenario, there is traffic from interface to many customers. The scenario is common for web service providers where traffic from various internet sources access web content in the service providers hosting servers. In this case, the UBRL applied at the ingress direction is called ingress UBRL. The web service provider could use an ingress UBRL to rate limit individual access to the servers and avoid denial of service (DoS) attacks.

Scale Information for UBRL

The UBRL feature is supported on ASR 9000 Enhanced Ethernet line cards (LCs). The scale information for UBRL is:

- Up to 256000 unique flow entries are supported for SE (Service Optimized) and 4000 for TR (Transport Optimized) version of the LCs for Cisco ASR 9000 Series Routers.
- Cisco ASR 9001 also supports the same scale as supported by Cisco ASR 9000 Series Routers.
- The scale is configurable per LC.

Note Full scale is not achieved for a configured scale size due to hardware resource recycling restrictions. The final scale may vary between M (maximum size) and M - 64, depending on internal hardware resource recycle rate and incoming flow fluctuations.

UBRL Policer Scale Information for ASR 9000 High Density 100GE Ethernet LCs on Cisco IOS XR

Cisco ASR 9000 High Density 100GE Ethernet LCs supports a maximum of 368000 unique flows and a minimum of 1000 unique flow entries for each NP. This flow scale is shared by CAC and UBRL. The following are the list of supported line cards:

- A9K-8X100GE-SE
- A9K-8X100GE-TR
- A9K-4X100GE-SE
- A9K-4X100GE-TR
- A99-8X100GE-SE
- A99-8X100GE-TR
- A9K-MOD400-SE
- A9K-MOD400-TR
- A9K-MOD200-SE
- A9K-MOD200-TR
- A9K-400G-DWDM-TR
- A99-12X100GE
- A9K-48X10GE-1G-SE/-TR
- A9K-24X10GE-1G-SE/-TR
- A99-48X10GE-1G-SE/-TR
- A9K-4X100GE

Flow aware policy details on SE and TR are:

- On SE card, a maximum of 64 flow aware policy instances to a line card is possible. The flow table scale is 368000 unique flow entries for each NP.
- On TR card, a maximum of 256 flow aware policy instances to a line card is possible. The flow table scale is 3000 to 4000 unique flow entries for each NP.

Note Applying more than the supported flow aware policy instances to a line card, leads to very delays on bulk policy removal, MPA OIR, commit replace operations and so on. It also causes unintentional timeouts in various operations.

Flow Masks for UBRL

A flow mask defines what fields constitute or differentiate a flow. The Flow Aware QoS feature supports these flow masks listed in the flow table:

Table 9: Flow Masks for UBRL

Flow Mask	Description
5 tuple (srcip, dstip, proto, sport, dport)	Session or Application Policer. The flow mask includes IPv4 source or destination address, L4 protocol number, and source or destination L4 port numbers.
srcip	Specifies the IPv4 or IPv6 source address only flow mask.
dstip	Specifies the IPv4 or IPv6 destination address only flow mask.

Restrictions

- UBRL does not support 5-tuple flows with IPv6 traffic due to address length constraints.
- UBRL supports only L3 (routed) interface. UBRL is not supported on L2 and MPLS interfaces.
- UBRL is not supported for Pseudowire Headend (PWHE), Bridge Virtual Interface (BVI), Broadband Network Gateway (BNG) subscriber interfaces, cluster Inter Rack Link (IRL) and satellite interfaces.
- UBRL actions are not supported in the same class.
- UBRL actions are supported only at the leaf level.
- UBRL does not support percentage policer rates or conform-aware and color-aware policer actions.
- UBRL does not support combination of flow masks such as srcip+dstip.
- UBRL does not support **flow idle-timeout none** and **max flow count per-class**.
- UBRL supports IPv4 and IPv6 unicast traffic topologies. Multicast traffic is not supported.
- UBRL support for IPv6 is restricted to src-ip or dst-ip flow masks.
- UBRL does not support combination feature such as UBRL + shared policy instance (SPI) or UBRL + shared policer feature.
- UBRL and policy based forwarding (PBF) feature will not work together on the same interface or direction.
- Flow idle-timeout has a 10s granularity. Hence, the actual purge of a specific flow entry could be off by another 10s.

- For 5-tuple key with unknown (non TCP and UDP) protocol, UBRL degrades 5-tuple key to a 3 tuple key usage (src-ip + dst-ip + protocol number).
- Flows are learnt and per-flow resources allocated by the feature even when the packets in the flow are dropped by features that get applied after QoS or by fabric and egress card.
- There could be traffic drops during scaled flow learning at Internet mix or lower traffic rates matching UBRL classes. The flow push back drops and flow discard rate increases as load on NP increases.
- Ingress marking does not work on the packets that the router can't forward such as expired time to live (TTL) packets. QoS policy is matched and show policy-map counters increment correctly. But the packets post punt and inject on transmission do not have the remarked precedence to differentiated services code point (DSCP).
- For 5 tuple flow mask and IPv4 fragment traffic flows, the first fragment would be learnt with the correct L4 details. For the subsequent fragments the flow entry will not have the L4 port details and gets degraded to 3 tuple. This can cause oversubscription due to two policers allocated (one per flow) or congestion for fragmented flows when many fragmented streams between the same IP peers match the same second flow record.
- Applying more than 64 flow aware policy instances to a line card is possible. However, removal of more than 64 flow aware policy instances simultaneously during configuration replacement, reverting to the previous configuration, saving multiple configurations, and so on, can lock the console for long durations and cause unintentional timeouts in various operations.

How to Configure Flow Aware QoS

Configuring Flow Aware CAC Reject Action

Perform these tasks to configure flow aware call admission control (CAC) for the CAC reject action.

Before you begin

- Enable flow aware CAC feature on LCs (line cards). Use the **hw-module flow-qos location** *node-id* **max-flow-count** *value* command in Admin configuration mode.
- Reload LCs for the changes to take effect.
- To verify status, use the **show qos flow-aware summary location** command in EXEC mode.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match precedence***precedence-value* [*precedence-value1 ... precedence-value6*]
4. **exit**
5. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
6. **match access-group** [**ipv4** | **ipv6**] *access-group-name*
7. **exit**
8. **policy-map** [**type qos**] *policy-name*
9. **class** *class-name*

10. **police rate** *rate*
11. **exit**
12. **exit**
13. **class** *class-name*
14. **set dscp***tunnel-value*
15. **admit cac local**
16. **flow idle-timeout** *value*
17. **flow rate** *value*
18. **rate** *rate*
19. **exit**
20. **exit**
21. **class** *class-name*
22. **police rate** *rate*
23. Use the **commit** or **end** command.
24. **show running-config class-map**
25. **show running-config policy-map**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-all prec5`	Creates a class map to be used for matching packets to the class specified 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 precedence***precedence-value* [*precedence-value1 ... precedence-value6*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match precedence 5`	Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.

	Command or Action	Purpose
Step 5	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-any video`	Creates a class map to be used for matching packets to the class specified 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 6	**match access-group** [**ipv4** | **ipv6**] *access-group-name* **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match access-group ipv4 102`	(Optional) Configures the match criteria for a class map based on the specified access control list (ACL) name.
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 8	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map premium-services`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 9	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class prec5`	Specifies the name of the class whose policy you want to create or change.
Step 10	**police rate** *rate* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 100 mbps`	Configures the traffic policing rate and enters policy map police configuration mode.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 12	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 13	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class video`	Specifies the name of the class whose policy you want to create or change.
Step 14	**set dscp***tunnel-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set dscp af41`	Sets the IP differentiated services code point (DSCP) in the type of service (ToS) byte to AF41.
Step 15	**admit cac local** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set dscp af41`	Configures the call admission control (CAC) local flow type and enters the policy map class cac configuration sub-mode.
Step 16	**flow idle-timeout** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-cac)# flow idle-timeout 20`	Configures the maximum time of inactivity for the flow as 20 seconds.
Step 17	**flow rate** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-cac)# flow rate 128`	Configures the per flow rate for the flow as 128 kbps.
Step 18	**rate** *rate* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-cac)# rate 896 kbps`	Configures the per flow rate for the flow as 896 kbps.
Step 19	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 20	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 21	**class** *class-name* **Example:**	Specifies the name of the class whose policy you want to create or change.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	
Step 22	**police rate** *rate* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 30 mbps`	Configures the traffic policing rate and enters policy map police configuration mode.
Step 23	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 24	**show running-config class-map** **Example:** `RP/0/RSP0/CPU0:router# show running-config class-map`	Displays the configuration of all class maps configured on the router.
Step 25	**show running-config policy-map** **Example:** `RP/0/RSP0/CPU0:router# show running-config policy-map`	Displays the configuration of all policy maps configured on the router.

Configuring Flow Aware CAC Redirect Action

Before you begin

- Enable flow aware CAC feature on LCs (line cards). Use the **hw-module flow-qos location** *node-id* **max-flow-count** *value* command in Admin configuration mode.
- Reload LCs for the changes to take effect.
- To verify status, use the **show qos flow-aware summary location** command in EXEC mode.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*

3. **match dscp***value*
4. **exit**
5. **class-map** [**type qos**] **match-all** *class-map-name*
6. **match cac admitted local**
7. **exit**
8. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
9. **match dscp** *value*
10. **end-class-map**
11. **policy-map** [**type qos**] *policy-name*
12. **class** *class-name*
13. **set discard-class***value*
14. **exit**
15. **class** *class-name*
16. **set dscp** *value*
17. **exit**
18. **exit**
19. **policy-map** [**type qos**] *policy-name*
20. **class** *class-name*
21. **police rate** *rate*
22. **exit**
23. **exit**
24. **class** *class-name*
25. **service-policy** *policy-map*
26. **admit cac local**
27. **flow idle-timeout** *value*
28. **flow rate** *value*
29. **rate** *rate*
30. **exit**
31. **exit**
32. **class** *class-name*
33. **police rate** *rate*
34. Use the **commit** or **end** command.
35. **show running-config class-map**
36. **show running-config policy-map**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-any dscp_cs5`	Creates a class map to be used for matching packets to the class specified 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 dscp***value* **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match dscp cs5`	Identifies DSCP values as match criteria in a class map.
Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 5	**class-map** [**type qos**] **match-all** *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-all video_admitted`	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode.
Step 6	**match cac admitted local** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match cac admitted local`	Specifies the packets admitted by CAC action as the match criteria in a class map.
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 8	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-all dscp_cs6`	Creates a class map to be used for matching packets to the class specified 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 9	**match dscp** *value* **Example:**	Identifies DSCP values as match criteria in a class map.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-cmap)# match dscp cs6`	
Step 10	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 11	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map video_flows`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 12	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class video_admitted`	Specifies the name of the class whose policy you want to create or change.
Step 13	**set discard-class***value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set discard-class 1`	Sets the discard class on IP Version 4 (IPv4) or Multiprotocol Label Switching (MPLS) packets.
Step 14	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 15	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change.
Step 16	**set dscp** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set dscp cs4`	Marks the packet by setting the DSCP in the ToS byte to cs4.
Step 17	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.

	Command or Action	Purpose
Step 18	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 19	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map premium_services`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 20	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class dscp_cs5`	Specifies the name of the class whose policy you want to create or change.
Step 21	**police rate** *rate* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 100 mbps`	Configures the traffic policing rate and enters policy map police configuration mode.
Step 22	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 23	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 24	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class dscp_cs6`	Specifies the name of the class whose policy you want to create or change.
Step 25	**service-policy** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy video_flows`	Attaches a policy map to an output interface to be used as the service policy for that interface.

	Command or Action	Purpose
Step 26	**admit cac local** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set dscp af41`	Configures the call admission control (CAC) local flow type and enters the policy map class cac configuration sub-mode.
Step 27	**flow idle-timeout** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-cac)# flow idle-timeout 20`	Configures the maximum time of inactivity for the flow as 20 seconds.
Step 28	**flow rate** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-cac)# flow rate 128`	Configures the per flow rate for the flow as 128 kbps.
Step 29	**rate** *rate* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-cac)# rate 896 kbps`	Configures the per flow rate for the flow as 896 kbps.
Step 30	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 31	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 32	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change.
Step 33	**police rate** *rate* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 30 mbps`	Configures the traffic policing rate and enters policy map police configuration mode.
Step 34	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 35	**show running-config class-map** **Example:** `RP/0/RSP0/CPU0:router# show running-config class-map`	Displays the configuration of all class maps configured on the router.
Step 36	**show running-config policy-map** **Example:** `RP/0/RSP0/CPU0:router# show running-config policy-map`	Displays the configuration of all policy maps configured on the router.

Configuring User Based Rate Limiting (UBRL)

Before you begin

- Enable UBRL feature on LCs (line cards). Use the **hw-module flow-qos location** *node-id* **max-flow-count** *value* command in Admin configuration mode.
- Reload LCs for the changes to take effect.
- To verify status, use the **show qos flow-aware summary location** command in EXEC mode.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-all**] *class-map-name*
3. **match precedence** *precedence-value*
4. **match flow-key** [**5-tuple** | **dst-ip** | **flow-cache***idle-timeout* | **src-ip**]
5. **exit**
6. **policy-map** [**type qos**] *policy-name*
7. **class** *class-name*
8. **police rate** *rate*
9. **exit**
10. **exit**
11. **exit**
12. **interface** *type interface-path-id*

13. **service-policy** {**input** | **output**} *policy-map*
14. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-all ubrl-src-class`	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode. If you specify **match-all**, the traffic must match all the match criteria.
Step 3	**match precedence** *precedence-value* **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match precedence 0 1 2 3`	Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 4	**match flow-key** [**5-tuple** | **dst-ip** | **flow-cache***idle-timeout* | **src-ip**] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match flow-key src-ip`	Identifies the specified flow key as the match criteria. • Use **5-tuple** flow key to configure multiple sessions. • Use **dst-ip** flow key to configure outbound traffic. • Use **flow-cache** flow key to configure flow cache parameters. • Use **src-ip** flow key to configure inbound traffic. • Use *idle-timeout* flow key to configure idle timeout period in seconds. The range is from 10 to 2550. The default value is 30.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# exit`	Returns the router to global configuration mode.
Step 6	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map ubrl-src`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

	Command or Action	Purpose
Step 7	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class ubrl-src-class`	Specifies the name of the class whose policy you want to create or change.
Step 8	**police rate** *rate* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 200 kbps`	Configures the traffic policing rate and enters policy map police configuration mode.
Step 9	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c-police)# exit`	Returns the router to policy map class configuration mode.
Step 10	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# exit`	Returns the router to policy map configuration mode.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to global configuration mode.
Step 12	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/1/0/9`	Configures an interface and enters the interface configuration mode.
Step 13	**service-policy** {**input** | **output**} *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy input ubrl-src`	Attaches a policy map to an input or output interface to be used as the service policy for that interface. In this example, the traffic policy evaluates all traffic leaving that interface.
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuration Examples for Configuring Flow Aware QoS

Configuring Flow Aware CAC Reject Action: Example

In this example, two class-maps are created and their match criteria are defined for access-list 102 and match class "video". This flow rate is configured in the admit cac local configuration sub-mode. If any new flow is learnt apart from the already admitted flows, then the new flow is rejected and packets of the flow are dropped. All other packets are classified under class-default and are policed at 30 mbps.

```
class-map match-all prec5
         match precedence 5
   !

class-map match-any video
         match access-group ipv4 102
   !

policy-map premium-services
  class prec5
      police rate 100 mbps
  class video
      set dscp af41
       admit cac local
       flow idle-timeout 20
       flow rate 128 kbps
       rate 896 kbps
 !
!
class class-default
       police rate 30 mbps
end
```

Configuring Flow Aware CAC Redirect Action: Example

In this example, three class-maps are created and their match criteria are defined for match class "dscp_cs5", match class cac, match class "dscp_cs6". This flow rate is configured in the admit cac local configuration sub-mode. If any new flow is learnt apart from the already admitted flows, then the new flow is redirected and the packets for that flow are handled by the redirect class "class-default" in policy "video_flows". All other packets are classified under class-default and are policed at 30 mbps.

```
class-map match-any dscp_cs5
         match dscp cs5
!
class-map match-all video_admitted
        match cac admitted local
!
class-map match-all dscp_cs6
         match dscp cs6
```

```
!
policy-map video_flows
   class video_admitted
     set discard-class 1
   class class-default
     set dscp cs4
!
!
 policy-map premium_services
   class dscp_cs5
      police rate 100 mbps
   class dscp_cs6
      service-policy video_flows
      admit cac local
        flow idle-timeout 20
        flow-rate 128 kbps
        rate 896 kbps
 !
!
class-default
police rate 30 mbps
end
```

Configuring UBRL for Multiple Sources: Example

In this example, a class-map is created and the match criteria is defined for match precedence and match flow-key based on the source IP (src-ip).

```
class-map match-all ubrl-src
    match precedence 0 1 2 3
    match flow-key src-ip
!
policy-map ubrl-mult-src
    class ubrl-src
        police rate 200 kbps
    !
!
interface gigabitethernet 0/0/0/4
    service-policy input ubrl-mult-src
!
end
```

Configuring Bidirectional UBRL: Example

In this example, two class-maps are created, one for inbound and another for outbound traffic, and match criteria are defined. The policy-maps are applied on the input and output direction of the interface.

```
class-map match-all ubrl-src
    match precedence 0 1 2 3
    match flow-key src-ip
!
class-map match-all ubrl-dst
    match precedence 0 1 2 3
    match flow-key dst-ip
!

policy-map ubrl-mult-src
    class ubrl-src
        police rate 200 kbps
```

```
    !
!
policy-map ubrl-mult-dst
    class ubrl-dst
        police rate 200 kbps
    !
!
interface gigabitethernet 0/0/0/4
    service-policy input ubrl-mult-src
    service-policy output ubrl-mult-dst
!
end
```

Configuring UBRL for Multiple Sessions: Example

In this example, a class-map is created and the match criteria is defined for match precedence and match flow-key based on 5-tuple.

```
class-map match-all ubrl-sess
    match precedence 0 1 2 3
    match flow-key 5-tuple
!

policy-map ubrl-mult-sess
    class ubrl-sess
        police rate 200 kbps
    !
!
interface gigabitethernet 0/0/0/4
    service-policy input ubrl-mult-sess
!
end
```

Additional References

The following sections provide references related to implementing QoS.

Related Documents

Related Topic	Document Title
Initial system bootup and configuration	*Cisco ASR 9000 Series Aggregation Services Router Getting Guide*
QoS commands	*Cisco ASR 9000 Series Aggregation Services Router Modular of Service Command Reference*
User groups and task IDs	"Configuring AAA Services on Cisco ASR 9000 Series Route of Cisco Cisco ASR 9000 Series Aggregation Services Route Security Configuration Guide

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

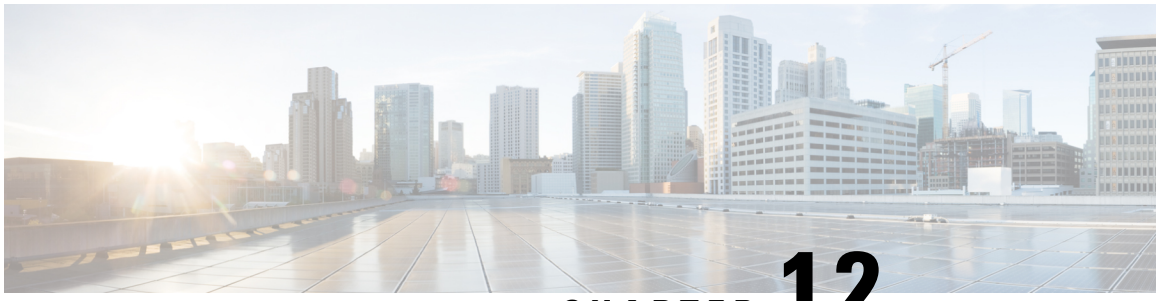
MIBs	MIBs Link
CISCO-CLASS-BASED-QOS-MIB	To locate and download MIBs using Cisco IOS XR softw Cisco MIB Locator found at the following URL and choo under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.sht

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 12

Configuring QoS on the Satellite System

Release	Modification
Release 6.1.2	Included details for QoS offload on NCS 5000 Series Satellite.

QoS on the Satellite System

AutoQoS which automates consistent deployment of QoS features is enabled on the satellite system. All the user-configured Layer2 and Layer3 QoS features are applied on the ASR9000 and no separate Qos configuration required for the satellite system. Auto-Qos handles the over-subscription of the ICL links. All other QoS features, including broadband QoS, on regular ports are supported on satellite ports as well. System congestion handling between the ASR9000 Series Router and satellite ports is setup to maintain priority and protection. AutoQoS Provide sufficient differentiation between different classes of traffic that flow on the satellite ICLs between the ASR9000 Series Router and the Satellite .

The system can support up to 14 unique shape rates for 1G port shapers. 1G ports are represented using a L0 entity in the Traffic Manager (TM) hierarchy. Port shapers are applied at this level. When speed changes on satellite ports, QOS EA would automatically reconfigure any policy-maps based on underlying satellite ports speed. However if there are no policies, then the Policy Manager (PM) needs to setup the speed of the port by calling the port-shaper API (Application Programming Interface). The system shall modify any policies which are percentage-based when the underlying ports speed changes due to AN. There would be a timelag for the Autonegotiated speed to be propagated to the policies on the ASR9000 series router and during that time, packet drops are expected in the satellite device.

Note On satellite access ports, the port and top-level policy shapers calculate traffic rates using the full Layer 1 frame size, including physical and internal overheads. This approach enforces accurate bandwidth limits based on actual transmission usage.

For more information about QoS for the satellite system, refer the *Modular QoS Configuration Guide for Cisco ASR 9000 Series Routers*.

Limitations

- Queueing on an ingress service-policy is not supported on satellite interfaces.
- Only flat and 2-level HQoS policies are supported on satellite interfaces in L2 Fabric and simple ring topologies.
- The burst size can be set to a wide range of sizes up to 2000 ms. However, for satellite ports, the actual burst size when queuing (shaping) is used is always set to 500 usec of 1Gbits or less. This is because of constraints in the hardware.

Auto QoS

Traffic from the Satellite nV system to the Cisco ASR 9000 series router and traffic from the Cisco ASR 9000 series router to the Satellite nV system have been discussed.

Satellite to Cisco ASR 9000 Series Router

- Traffic is handled using the trusted port model.
- Automatic packet classification rules determine whether a packet is control packet (LACP, STP, CDP, CFM, ARP, OSPF etc), high priority data (VLAN COS 5,6,7, IP prec 5, 6, 7) or normal priority data and queued accordingly.

Note Cisco NCS 5000 Series satellite does not classify further into LACP, OAM, BFD and so on as on earlier satellites.

- All user-configured Layer 2 and Layer 3 features(including QoS) are applied on the Cisco ASR 9000 Series Host and not on the satellite.
- Protocol types auto-prioritized by the satellite - all IEEE control protocols (01 80 C2 xx xx xx), LACP, 802.3ah, CFM, STP, CDP, LLDP, ARP, OSPF, BFD, RIP, BGP, IGMP, RSVP, HSRP, VRRP p2 q.

Note Cisco NCS 5000 Series satellite does not auto prioritize the protocols mentioned above.

- User data packets auto-prioritized by the satellite - VLAN COS 5, 6, 7, IP precedence 5, 6, 7 MPLS EXP 5, 6, 7. MPLS EXP is not classified in the case of Cisco NCS 5000 series satellite.

Note Cisco NCS 5000 Series devices used as nV satellite

Figure 17: AutoQoS, Cisco ASR 9000v satellite to host

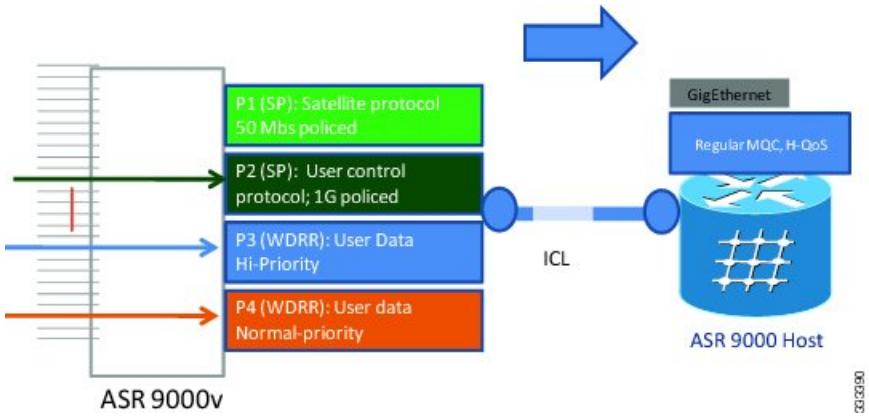
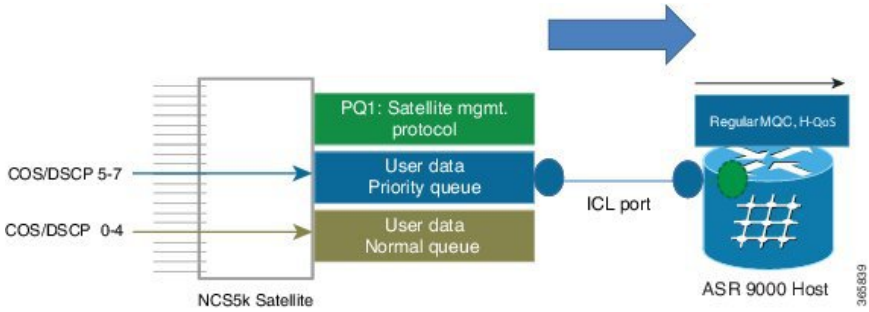


Figure 18: AutoQoS, Cisco NCS 500x series satellite to host



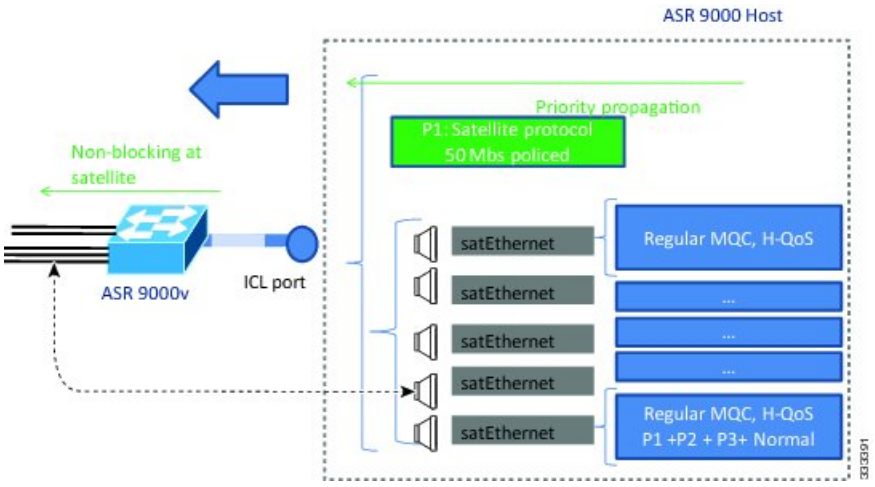
Cisco ASR 9000 Series Router to Satellite

- Traffic targeted to a satellite egress port is shaped on Cisco ASR 9000 to match downstream access port speed.

Note There is no need for further QoS on the satellite itself, since Cisco ASR 9000 QoS is sufficient and provides necessary deep buffering normally not available on Cisco ASR 9000v satellite device with its 4 MB buffers or Cisco NCS 5000 Series standalone device with its 16 MB buffers.

- Traffic is streamed based on the full 3-level egress queuing hierarchy.
- Each remotely managed satellite access GigE port is auto-shaped to match access line speed.
- Satellite protocols going over ICL default queues get highest scheduling priority while full 3 level MQC hierarchy is supported on the egress satellite ports.

Figure 19: AutoQoS, host to satellite



Note The above connections are also applicable to the Cisco NCS 5000 Series devices used as nV satellite.

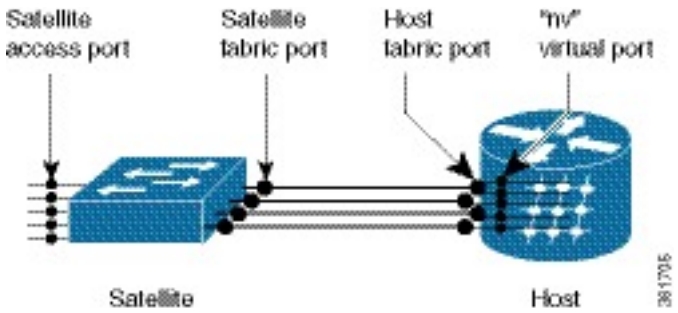
QoS Offload on Satellite

The Cisco ASR 9000 Series Router Satellite System enables you to configure a topology in which one or more satellite switches complement one or more Cisco ASR 9000 Series Router, to collectively deploy a single virtual switching system. In this system, the satellite switches act under the management control of the routers. The connections between the Cisco ASR 9000 Series Router and the satellite switches are called the Inter-chassis link (ICL), which is established using standard Ethernet interfaces.

The ICL link between the Cisco ASR 9000 Series Router and the satellite gets oversubscribed by the access interfaces on the satellite box. This is because the QoS policies applied on the satellite interfaces are programmed on the Cisco ASR 9000 Series Router Line card locally. Therefore, the flow of traffic on the ICL from the satellite switch is not controlled. This leads a loss of high-priority traffic due to congestion on the ICL.

This figure shows the ports where the QoS policies may be applied.

Figure 20: Satellite and Host connection



Benefits of QoS Offload

The QoS offload feature protects the control packets when Satellite fabric links (SFL) is congested. The offloading of QoS policies helps to drop excess traffic at the ingress direction (or access ports) and prioritize the protocol control traffic at the egress direction (or SFL).

Supported Platform-Specific Information for QoS Offload

This section describes the supported capability matrix, various supported classification combinations, and the supported scalability matrix for 9000v and ASR 901 satellites.

Supported Capability Matrix

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
Classification					
Ingress					
COS	Yes	Yes	Yes	0-7	The cos classification is done on the outer vlan tag. **Note** The cos classification based on match-rule is not applicable for untagged packets on the ingress direction.
IP DSCP	Yes	Yes	Yes	0-63	IP DSCP is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side. IP DSCP is supported for IPv4 and IPv6.

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
IP PREC	Yes	Yes	Yes	0-7	IP PREC is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side. IP PR is supported only for IPv4.
MPLS EXPERIMENTAL TOPMOST	Yes	No	Yes	0-7	The mpls experimental topmost feature is supported only for the untagged packets on the ingress direction, from the access-side.
VLAN	Yes	No	No	1-4096	The vlan classification is done on the outer vlan tag based on the policies and the cos value applied on the outer vlan tag. **Note** The vlan classification based on outer vlan tag is not applicable for untagged packets on the ingress direction.
Egress					

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
QOS-GROUP	Yes	Yes	Yes	• 1-5 for 9000v • 1-7 for Cisco NCS 500x	A class-map with multiple "match qos-group" statements is not supported. **Note** • qos-group 0 corresponds to class-default, hence, it cannot be configured. • For 9000v, qos-group 6 and qos-group 7 are reserved, and hence, it cannot be configured.
IP DSCP	No	No	Yes	0-63	—
IP PREC	No	No	Yes	0-7	—
Marking					
Ingress					
COS	Yes	Only outer COS	No	0-7	The cos marking is done on the vlan tag that is added by the satellite on the direction towards host.
DISCARDCLASS	NA	NA	Yes	0-2	The discard-class feature is used along with WRED. But, WRED is not supported in 9000v. Hence, this feature is supported only in 901 satellites.

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
IP DSCP	Yes **Note** IP DSCP marking is supported for IPv4 and IPv6.	Yes	Yes **Note** IP DSCP marking is supported for IPv4.	0-63	IP DSCP is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side.
MPLS EXPERIMENTAL IMPOSITION	No	No	Yes	0-7	—
IP PREC	Yes	No	Yes	0-7	IP PREC is supported for untagged, single-tagged, double-tagged, and mac-in-mac packets on the ingress direction, from the access-side.
QOS-GROUP	Yes	Yes	Yes	• 1-5 for 9000v • 1-7 for Cisco NCS 500x	The qos-group marking feature is only used to redirect packets to a particular queue. The set qos-group 0 on ingress policy is necessary to send the packets to queue 0 on ICL. **Note** If the QoS classification rule at the ICL interface in the egress and ingress direction matches, then the packets are directed to the configured group, else the packets are directed to the class-default group.

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
Police Actions (Ingress Marking)					
QOS-GROUP TRANSMIT	Yes	Yes	Yes	0-5	The set qos-group 6 and 7 is not configurable. On 901 satellites, qos-group 0 is not configurable.
PREC-TRANSMIT	Yes	Yes	Yes	0-7	—
DISCARD-CLASS	No	No	Yes	0-2	—
DSCP-TRANSMIT	Yes	Yes	Yes	0-63	—
COS-TRANSMIT	Yes	Yes	No	0-7	The cos-transmit is done on the vlan tag that is added by the satellite on the host direction.
Egress (Marking)					
IP DSCP	No	No	Yes	0-63	—
IP PREC	No	No	Yes	0-7	—
MPLS EXPERIMENTAL TOPMOST	No	No	Yes	0-7	—
Queuing					
Egress					
Note: For 901 satellite, queuing related actions such as bandwidth, priority, or shape is supported only with **qos-group** classification.					

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
Bandwidth Value	Yes	No	No	8-10000000	For a 9000v satellite, bandwidth value cannot be configured under qos-group 3. A combination of bandwidth types cannot be configured. For example, the bandwidth command can be configured either with kbps, or remaining percent, or remaining ratio, but not with a combination of all.
Bandwidth Percent	Yes	No	Yes	—	
Bandwidth Remaining	Yes	Yes	Yes	1-127	
Bandwidth Remaining Percent	Yes	Yes	Yes	—	
Ratio	Yes	Yes	No	—	
Priority lev 1-3	Yes	Only Priority level 1 is supported	Yes	—	On 9000v satellites, when a priority level is configured at the host, it by default gets configured to priority percent 85 on the satellite. On 9000v satellites, the priority action cannot be combined with other queuing actions. On 9000v satellites, only one class-map with a priority action can be configured. On 9000v satellites, the priority action is only supported under qos-group 3.
Priority Percent	Yes	NA	Yes	—	
Random Detect ~~Discard-class-based~~	No	No	Yes	Discard-class: 0-2 Thresholds: 1-8192000	—

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
Shape Average	Yes	Yes	Yes	8000-10000000000	On 9000v satellites, the shape average command cannot be configured under qos-group 3. On 901 satellites, the shape command cannot be used in the class-default class map unless you use hierarchical policy maps and apply shaping to the parent policy map.
Shape Average Percent	Yes	Yes	No	—	On 9000v satellites, the shape average percent command cannot be configured under qos-group 3.
HQOS	Yes	No	Yes	—	Only **class-default** can be configured in the parent policy map, while configuring H-QoS in the egress direction. Only shape average is supported under the class-default of the parent policy map. For a 9000v satellite, the minimum value that is supported is 40 mbps. For a 901 satellite, the minimum value that is supported is 250 kbps.
Rate Limiting					

Feature	Support on 9000v Platform	Support on Cisco NCS 5000 Series Router (Only from R6.1.2 onwards)	Support on 901 Platform (Not supported from R5.3.3 onwards).	Range	Restrictions
1R2C	Yes	Yes. For more information, please refer *Modular QoS Configuration Guide for Cisco NCS 5000 Series Routers*	Yes	CIR/PIR: 8000-100000000000 Burst bytes: 1000-256000000 Burst ms:1-2000	The bytes can be configured in milliseconds (ms) only if CIR is in percent. **Note** • CIR stands for Committed Information Rate and PIR stands for Peak Information Rate. • Transmit and marking actions are not supported together.
1R3C 2R3C	Yes	NA	Yes		If the exceed-action command is configured, then violate-action is copied from exceed-action, by default. If the exceed-action is not configured, then violate-action and exceed-action are dropped. **Note** • On ASR 9000v platform, 1R3C and 2R3C statistics are supported only for conform & violate actions. • Transmit and marking actions are not supported together. On 901 satellites, only green and red counters are supported.

Supported Classification Combination

These are the allowed classification combination in Cisco ASR 9000 Series Router :

- COS + IP DSCP
- IP DSCP +VLAN
- COS + VLAN
- IP DSCP + IP PREC

Note The IP DSCP + IP PREC combination is not supported for 9000v.

The table lists the allowed classification combinations in 9000v:

Match-all class map	DSCP + PREC + COS
	PREC + DSCP + VLAN
Match-any class map	VLAN + COS + PREC + DSCP
	DSCP + VLAN + COS
	DSCP + PREC + COS
	VLAN + COS + PREC

Note For NCS 5000 Series Satellite, COS+DSCP match is the only supported classification combination on ingress. For Egress, policies can only match on qos-group (1 per class-map). For Egress offload policies on NCS 5000 Series Satellite, it is mandatory to configure eight class-maps including class-default for eight queues, even if all the class maps are not in use.

Supported Scalability Matrix for 9000v

Class-map with options	Number of Field Programmable (FP) entries needed per policy-map(max 8 classes)	Max policy-maps supported
cos (0-7)	7 + 1 (class default)	2304/8 = 288
ip dscp (0-63)	7 + 1	2304/8 = 288
ip precedence (0-7)	7 + 1	2304/8 = 288
vlan (1-4094)	7 + 1	2304/8 = 288
match-any or match-all with single argument		

Class-map with options	Number of Field Programmable (FP) entries needed per policy-map(max 8 classes)	Max policy-maps supported
cos + dscp cos+ prec cos + vlan dscp + vlan prec + vlan	2 *7 + 1 (class-default) = 15	2304/15 = 153.6
match-any with maximum arguments to the match parameters		
cos (max 4)+ ip precedence (max 4)	8 * 7 + 1 (class-default) = 57	2304/57 = 40.4
cos (4) + ip dscp (8)	12 * 7 + 1 (class-default)= 85	2304/85 = 27.1
cos (4) + vlan (30)	34 * 7 + 1 = 239	2304/239 = 9.6
vlan (30) + ip prec (4)	34 *7 + 1 = 239	2304/239 = 9.6
vlan (30)+ip dscp (8)	38*7 + 1 =267	2304/267 = 8.6
match-all with maximum arguments		
cos (4) + ip dscp (8)	32 *7 + 1=225	2304/225 = 10.2
cos (4) + vlan (30)	120 *7+ 1=841	2304/841 = 2.7
vlan (30) + ip prec (4)	120*7+1=841	2304/841 = 2.7
cos (4) + ip prec (4)	16 *7 +1= 113	2304/113 = 20.3
vlan (30) + ip dscp (8)	240 *7 + 1 = 1681	2304/1681 =1.3

Supported Scalability Matrix for 901

ASR 901 satellites are not supported from R5.3.3 onwards.

Note Any number of class-maps can be configured per policy-map. However, a maximum of only 32 policy-maps can be configured.

Class-map with options	Maximum Number of Field Programmable (FP) Entries
Class-map with options	300
cos (0-7)	
ip dscp (0-63)	
ip precedence (0-7)	
mpls exp topmost (0-7)	

QoS Offload Configuration Overview

Three steps to configure QoS Offload are:

1. Create a class-map of the type ‘qos’.
2. Create a policy-map of the type ‘qos’ using the above configured class map.
3. Bind QoS policy to Satellite interfaces such as physical access, bundle access, physical ICL, and bundle ICL.

To modify a QoS Offload configuration:

1. Modify the class-map or policy-map without unbinding the policy-map from the applied interface.

Note QoS Offload configuration with **police rate** in **pps** unit is not supported.

Sample QoS Offload Configuration

```
class-map match-any my_class
 match dscp 10
 end-class-map
!
policy-map my_policy
 class my_class
 police rate percent 30
!
end-policy-map
!
interface GigabitEthernet100/0/0/9
 ipv4 address 10.1.1.1 255.255.255.0
  nv
   service-policy input my_policy
!
!
```

Prerequisites for QoS Offload Configuration

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is

preventing you from using a command, contact your AAA administrator for assistance. Before configuring the QoS offload feature, you must have these hardware and software installed in your chassis.

- Hardware—Cisco ASR 9000 Series Aggregation Services Routers with Cisco ASR 9000 Enhanced Ethernet line cards as the location of Inter Chassis Links and Cisco ASR9000v or Cisco ASR9000v-V2 or Cisco NCS 500x Series as Satellite box.
- Software—Cisco IOS XR Software Release 5.1.1 or higher for ASR9000v and ASR 901 satellites. Cisco IOS XR Software Release 6.1.2 or higher for QoS offload and QoS offload on bundle ICL features, on Cisco NCS 5000 Series satellites.

Offloading Service-policy on Physical Access Port

Perform these tasks to offload the service-policy on the physical access port. This procedure offloads the service-policy in the ingress direction of the Satellite Ethernet interface.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match precedence***precedence-value* [*precedence-value1 ... precedence-value6*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **set qos-group** *qos-group-value*
8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*
11. (Optional) **l2transport**
12. **nv**
13. **service-policy input** *policy-map*
14. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:**	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# class-map match-any class1`	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 precedence***precedence-value* [*precedence-value1 ... precedence-value6*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match precedence 5`	Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 4	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class1`	Specifies the name of the class whose policy you want to create or change.
Step 7	**set qos-group** *qos-group-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set qos-group 5`	Sets the QoS group identifiers on IPv4 or MPLS packets.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:**	Configures an interface and enters the interface configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 100/0/0/0`	
Step 11	(Optional) **l2transport** **Example:** `RP/0/RSP0/CPU0:router(config-if)# l2transport`	Configures the L2 transport offload for satellite.
Step 12	**nv** **Example:** `RP/0/RSP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 13	**service-policy input** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if-nV)# service-policy input policy1`	Attaches a policy map to an input interface to be used as the service policy for that interface.
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Offloading Service-policy on Bundle Access Port

Perform these tasks to offload the service-policy on the bundle access port. This procedure offloads the service-policy in the ingress direction of the Satellite Ethernet interface.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match precedence***precedence-value*
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **set qos-group** *qos-group-value*
8. **exit**

9. **end-policy-map**
10. **interface** *type interface-path-id*
11. **bundle id** *bundle-id*
12. (Optional) **l2transport**
13. **nv**
14. **service-policy input** *policy-map*
15. Use the **commit** or **end** command.
16. **exit**
17. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-any class2`	Creates a class map to be used for matching packets to the class specified 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 precedence***precedence-value* **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match precedence 6`	Identifies IP precedence values as match criteria. • Value range is from 0 to 7. • Reserved keywords can be specified instead of numeric values.
Step 4	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy2`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

	Command or Action	Purpose
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class2`	Specifies the name of the class whose policy you want to create or change.
Step 7	**set qos-group** *qos-group-value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# set qos-group 5`	Sets the QoS group identifiers on IPv4 or MPLS packets.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface bundle-ether 1`	Configures an interface and enters the interface configuration mode.
Step 11	**bundle id** *bundle-id* **Example:** `RP/0/RSP0/CPU0:router(config-if)# bundle id 1`	Creates a multilink interface bundle with the specified bundle ID.
Step 12	(Optional) **l2transport** **Example:** `RP/0/RSP0/CPU0:router(config-if)# l2transport`	Configures the L2 transport offload for satellite.
Step 13	**nv** **Example:** `RP/0/RSP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 14	**service-policy input** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-if-nV)# service-policy input policy2`	Attaches a policy map to an input interface to be used as the service policy for that interface.

	Command or Action	Purpose
Step 15	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 16	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-if)# exit`	Returns the router to global configuration mode.
Step 17	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Offloading Service-policy on Physical Satellite Fabric Link

Perform these tasks to offload the service-policy on the physical Satellite Fabric Link (SFL). This procedure offloads the service-policy in the egress direction of SFL.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match qos-group** [*qos-group-value*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*

11. **nv**
12. **satellite-fabric-link satellite** *satellite_id*
13. **remote-ports** *interface_type remote_subslot*
14. **service-policy output** *policy-map*
15. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-any class3`	Creates a class map to be used for matching packets to the class specified 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 qos-group** [*qos-group-value*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match qos-group 5`	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 63. • Up to eight values (separated by spaces) can be entered in one match statement. • **match qos-group**command is supported only for an egress policy.
Step 4	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy3`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.

	Command or Action	Purpose
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class3`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 13`	Specifies the bandwidth allocated for a class belonging to a policy map.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/1/0/0`	Configures an interface and enters the interface configuration mode.
Step 11	**nv** **Example:** `RP/0/RSP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 12	**satellite-fabric-link satellite** *satellite_id* **Example:** `RP/0/RSP0/CPU0:router(config-if-nV)# satellite-fabric-link satellite 100`	Specifies an interface as an Interface Control Plane Extender(ICPE) inter-chassis link (ICL). **Note** The Interface Control Plane Extender(ICPE) infrastructure has a mechanism to provide the Control Plane of an interface physically located on the Satellite device in the local Cisco IOS XR software.
Step 13	**remote-ports** *interface_type remote_subslot* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# remote-ports Satellite-Ether 0/0/0-9`	Configures the remote satellite ports 0 to 9.

	Command or Action	Purpose
Step 14	**service-policy output** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# service-policy output policy3`	Attaches a policy map to an output interface to be used as the service policy for that interface.
Step 15	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Offloading Service-policy on Bundle SFL

Perform these tasks to offload the service-policy on the bundle Satellite Fabric Link (SFL). This procedure offloads the service-policy in the egress direction of SFL.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match qos-group** [*qos-group-value*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}
8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*
11. **bundle id** *bundle-id*
12. **nv**
13. **satellite-fabric-link satellite** *satellite_id*
14. **remote-ports***interface_type remote_subslot*
15. **service-policy output** *policy-map*
16. Use the **commit** or **end** command.
17. **exit**
18. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-any class4`	Creates a class map to be used for matching packets to the class specified 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 qos-group** [*qos-group-value*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match qos-group 5`	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 63. • Up to eight values (separated by spaces) can be entered in one match statement. • **match qos-group**command is supported only for an egress policy.
Step 4	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy4`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class4`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:**	Specifies the bandwidth allocated for a class belonging to a policy map.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 13`	
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 2`	Configures an interface and enters the interface configuration mode.
Step 11	**bundle id** *bundle-id* **Example:** `RP/0/RSP0/CPU0:router(config-if)# bundle id 2`	Creates a multilink interface bundle with the specified bundle ID.
Step 12	**nv** **Example:** `RP/0/RSP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 13	**satellite-fabric-link satellite** *satellite_id* **Example:** `RP/0/RSP0/CPU0:router(config-if)# satellite-fabric-link satellite 100`	Specifies an interface as an Interface Control Plane Extender(ICPE) inter-chassis link (ICL). **Note** The Interface Control Plane Extender(ICPE) infrastructure has a mechanism to provide the Control Plane of an interface physically located on the Satellite device in the local Cisco IOS XR software.
Step 14	**remote-ports***interface_type remote_subslot* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# remote-ports GigabitEthernet 0/0/0-5`	Configures the remote satellite ports 0 to 5.
Step 15	**service-policy output** *policy-map* **Example:**	Attaches a policy map to an output interface to be used as the service policy for that interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# service-policy output policy4`	
Step 16	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 17	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-if)# exit`	Returns the router to global configuration mode.
Step 18	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Offloading Service-policy on L2 Fabric Physical SFL

Perform these tasks to offload the service-policy on L2 Fabric physical Satellite Fabric Link (SFL). This procedure offloads the service-policy in the egress direction of SFL.

SUMMARY STEPS

1. **configure**
2. **class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name*
3. **match qos-group** [*qos-group-value1*]
4. **end-class-map**
5. **policy-map** [**type qos**] *policy-name*
6. **class** *class-name*
7. **bandwidth** {*bandwidth* [*units*] | **percent** *value*}

8. **exit**
9. **end-policy-map**
10. **interface** *type interface-path-id*
11. **encapsulation dot1q***vlan-identifier*
12. **nv**
13. **satellite-fabric-link satellite** *satellite_id*
14. **remote-ports***interface_type remote_subslot*
15. **service-policy output** *policy-map*
16. Use the **commit** or **end** command.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map** [**type qos**] [**match-any**] [**match-all**] *class-map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# class-map match-any class5`	Creates a class map to be used for matching packets to the class specified 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 qos-group** [*qos-group-value1*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match qos-group 5`	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 63. • Up to eight values (separated by spaces) can be entered in one match statement. • **match qos-group**command is supported only for an egress policy.
Step 4	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration.

	Command or Action	Purpose
Step 5	**policy-map** [**type qos**] *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy5`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy and enters the policy map configuration mode.
Step 6	**class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class5`	Specifies the name of the class whose policy you want to create or change.
Step 7	**bandwidth** {*bandwidth* [*units*] | **percent** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 13`	Specifies the bandwidth allocated for a class belonging to a policy map.
Step 8	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# exit`	Returns the router to policy map configuration mode.
Step 9	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# end-policy-map`	Ends the policy map configuration.
Step 10	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface TenGigabitEthernet 0/1/0/0.1`	Configures an interface and enters the interface configuration mode.
Step 11	**encapsulation dot1q***vlan-identifier* **Example:** `RP/0/RSP0/CPU0:router(config-if)# encapsulation dot1q 20`	Defines the encapsulation format as IEEE 802.1Q (**dot1q**), and specifies the VLAN identifier.
Step 12	**nv** **Example:** `RP/0/RSP0/CPU0:router(config-subif)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 13	**satellite-fabric-link satellite** *satellite_id* **Example:**	Specifies an interface as an Interface Control Plane Extender(ICPE) inter-chassis link (ICL). **Note**

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-if-nV)# satellite-fabric-link satellite 100`	The Interface Control Plane Extender(ICPE) infrastructure has a mechanism to provide the Control Plane of an interface physically located on the Satellite device in the local Cisco IOS XR software.
Step 14	**remote-ports***interface_type remote_subslot* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# remote-ports GigabitEthernet 0/0/0-5`	Configures the remote satellite ports 0 to 5.
Step 15	**service-policy output** *policy-map* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# service-policy output policy5`	Attaches a policy map to an output interface to be used as the service policy for that interface.
Step 16	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

How to Configure HQoS on a Satellite

Hierarchical QoS allows you to specify QoS behavior at multiple policy levels, which provides a high degree of granularity in traffic management. A hierarchical policy is a QoS model that enables you to specify QoS behavior at multiple levels of hierarchy.

Note HQoS is not supported on Cisco NCS 5000 Series satellites to Cisco ASR 9000 Series Hosts that have the Cisco ASR 9000 4th Generation QSFP28 based dense 100GE line cards. However, HQoS is supported on Cisco NCS 5000 Series satellites to Cisco ASR 9000 Series Hosts that have the Cisco ASR 9000 High-Density 100GE Ethernet line cards.

Configure the Traffic Class

Perform these tasks to create class-maps.

SUMMARY STEPS

1. **configure**
2. **class-map match-any** *class-map-name*
3. **match qos-group** [*qos-group-value*]
4. **end-class-map**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**class-map match-any** *class-map-name* **Example:**	Creates a class map to be used for matching packets to the class specified and enters the class map configuration mode. The **match-any** keyword indicates that atleast one of the match criteria must be met for traffic entering the traffic class to be classified as part of the traffic class.
Step 3	**match qos-group** [*qos-group-value*] **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# match qos-group 5`	Specifies service (QoS) group values in a class map to match packets. **Note** The **match qos-group** [*qos-group-value*] is just an example of one of the match commands that can be used. For a list of other match commands, see the Supported Capability Matrix table.
Step 4	**end-class-map** **Example:** `RP/0/RSP0/CPU0:router(config-cmap)# end-class-map`	Ends the class map configuration. **Note** Repeat Steps 1 through 4 to configure additional class-maps.

Configure the Traffic Policy

This procedure creates both the child policy and the parent policy and applies the child policy to the parent policy.

SUMMARY STEPS

1. **configure**
2. **policy-map** *child-policy-map-name*
3. **class** {*class-name* | **class-default**}

4. **bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*]
5. **end-policy-map**
6. **configure**
7. **policy-map** *parent-policy-map-name*
8. **class class-default**
9. **shape average** *rate* [*units*]
10. **service-policy** *child-policy-map-name*
11. **end-policy-map**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**policy-map** *child-policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# policy-map child-policy`	Creates a child policy map and enters the policy map configuration mode.
Step 3	**class** {*class-name* | **class-default**} **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class4`	Assigns the traffic class that you specify to the policy map. Enters policy map class configuration mode.

Step 4	**bandwidth** {*rate* [*units*] | **percent** *percentage-value*} **or bandwidth remaining** [**percent** *percentage-value* | **ratio** *ratio-value*]
	Example:
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth percent 30`
	`or`
	`RP/0/RSP0/CPU0:router(config-pmap-c)# bandwidth remaining percent 80`
		Specifies the minimum bandwidth allocated to a class as a percentage of link bandwidth.
		Specifies how to allocate excess bandwidth to a class.
		Note Repeat Steps 3 and 4 to include additional class-maps to the child-policy
		If you use "bandwidth remaining percent", minimum bandwidth is allocated for each queues based on the configured bandwidth, and the weights are equal for all the queues.
		If you use "bandwidth remaining ratio", the bandwidth is allocated for each queues based on weights and the minimum bandwidth requirement is zero.

	Command or Action	Purpose
Step 5	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# end-policy-map`	Ends the policy-map configuration.
Step 6	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 7	**policy-map** *parent-policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# policy-map parent-policy`	Creates a parent policy map and enters the policy map configuration mode.
Step 8	**class class-default** **Example:** `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	Configures the parent class-default class. **Note** • You can configure only the class-default class in a parent policy. Do not configure any other traffic class.
Step 9	**shape average** *rate* [*units*] **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# shape average 1 mbps`	Shapes traffic to the indicated bit rate. **Note** In the parent policy, only the **shape average** action is supported. For a 9000v satellite, the supported minimum value is 40 mbps. For a 901 satellite, the minimum value that is supported is 250 kbps.
Step 10	**service-policy** *child-policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy child-policy`	Applies a child-level policy to the top-level class-default class. **Note** The service-policy command applies the child-policy-map to the parent-policy-map.
Step 11	**end-policy-map** **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# end-policy-map`	Ends the policy-map configuration.

Attach Hierarchical Policies to the Interface

This procedure attached the hierarchical policies to the interface.

SUMMARY STEPS

1. **interface** *type interface-path-id*
2. **ipv4 point-to-point**
3. **ipv4 unnumbered** *interface-type interface-instance*
4. **nv**
5. **satellite-fabric-link network**
6. **redundancy iccp-group** *group-number*
7. **satellite** *satellite-id*
8. **remote-ports***interface_type remote_subslot*
9. **service-policy output** *parent-policy-map-name*
10. Use the **commit** or **end** command.
11. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet0/2/0/1`	Configures an interface and enters the interface configuration mode.
Step 2	**ipv4 point-to-point** **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 point-to-point`	Configures the IPv4 point to point address.
Step 3	**ipv4 unnumbered** *interface-type interface-instance* **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered Loopback10`	Enables IPv4 processing on a point-to-point interface without assigning an explicit IPv4 address to that interface.
Step 4	**nv** **Example:** `RP/0/RSP0/CPU0:router(config-if)# nv`	Enters the satellite network virtualization (nV) configuration submode.
Step 5	**satellite-fabric-link network** **Example:** `RP/0/RSP0/CPU0:router(config-if-nV)# satellite-fabric-link network`	Specifies the network type of Interface Control Plane Extender(ICPE) inter-chassis link (ICL).

	Command or Action	Purpose
Step 6	**redundancy iccp-group** *group-number* **Example:** `RP/0/RSP0/CPU0:router(config-sfl-network)# redundancy iccp-group 2`	Configures the ICCP redundancy group.
Step 7	**satellite** *satellite-id* **Example:** `RP/0/RSP0/CPU0:router(config-sfl-network) # satellite 500`	Specifies the satellite ID.
Step 8	**remote-ports***interface_type remote_subslot* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# remote-ports GigabitEthernet 0/0/0-9`	Configures the remote satellite ports 0 to 5.
Step 9	**service-policy output** *parent-policy-map-name* **Example:** `RP/0/RSP0/CPU0:router(config-satellite-fabric-link)# service-policy output parent-policy`	Attaches a policy map to an output interface to be used as the service policy for that interface. **Note** Repeat Steps 7 through 9 to attach the policy map to the satellite interfaces.
Step 10	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 11	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-if)# exit`	Returns the router to global configuration mode.

Configuration Examples for QoS Offload

Note While the examples use 1G access ports and 10G fabric ports, the same can be applied to Cisco NCS 5000 series 10G access and 10G/100G fabric ports for supported scenarios.

Offloading Service-policy on Physical Access Port: Example

In this example, a service-policy called policy1 is created. This service policy is associated to a class map called class1 through the use of the class command, and then the service policy is attached in the input direction on a GigabitEthernet interface 100/0/0/0. This service-policy is configured under the nv mode and thus the QoS policy is offloaded to the satellite.

```
config
class-map match-any class1
 match precedence 6
 end-class-map
!
policy-map policy1
 class class1
  set qos-group 5
 !
interface gigabitEthernet 100/0/0/0
nv
service-policy input policy1
end or commit
```

Offloading Service-policy on Bundle Access Port: Example

In this example, a service-policy called policy2 is created. This service policy is associated to a class map called class2 through the use of the class command. The service policy is then attached in the input direction on a bundle-ether interface with bundle id as 1 that has two bundle member links—GigabitEthernet interface 100/0/0/1 and GigabitEthernet interface 100/0/0/2. This service-policy is configured under the nv mode and thus the QoS policy is offloaded to the satellite bundle-ether interface.

```
config
class-map match-any class2
 match precedence 6
 end-class-map
!
policy-map policy2
 class class2
  set qos-group 5
  end-policy-map
 !
interface bundle-ether 1
bundle-id 1
nv
service-policy input policy2
end or commit
!
end or commit
```

Offloading Service-policy on Physical SFL: Example

In this example, a service-policy called policy3 is created, which is associated to a class map called class3 through the use of the class command. The service policy is applied to the host-facing satellite fabric link (SFL) on the satellite 100 and attached in the output direction on a TenGigE interface 0/1/0/0. This is configured under the nv mode and thus the QoS policy is offloaded to the satellite.

```
config
class-map match-any class3
 match qos-group 5
 end-class-map
!
policy-map policy3
 class class3
  bandwidth percent 13
 !
interface TenGigE 0/1/0/0
nv satellite-fabric-link satellite 100
remote-ports GigabitEthernet 0/0/0-9
service-policy output policy3
end or commit
```

Offloading Service-policy on Bundle SFL: Example

In this example, a service-policy called policy4 is created, which is associated to a class map called class4 through the use of the class command. The service policy is applied to the host-facing bundle satellite fabric link (SFL) on the satellite 100 and attached in the output direction on the bundle-ether interface with bundle id 2 that has two bundle member links—TengGig interface 0/1/0/0 and TengGig interface 0/1/0/1. This is configured under the nv mode and thus the QoS policy is offloaded to the satellite.

```
config
class-map match-any class4
 match qos-group 5
 end-class-map
!
policy-map policy4
 class class4
  bandwidth percent 13
 !
interface Bundle-ether 2
nv satellite-fabric-link satellite 100
remote-ports GigabitEthernet 0/0/0-5
service-policy output policy4
exit/commit
interface TengGig 0/1/0/0
bundle-id 2
!
interface TengGig 0/1/0/1
bundle-id 2
!
end or commit
```

Offloading Service-policy on L2 Fabric physical SFL: Example

In this example, a service-policy called policy5 is created, which is associated to a class map called class5 through the use of the class command. The service policy is applied to the host-facing bundle SFL under the

nv mode and attached in the output direction on the TenGigabitEthernet 0/1/0/0.1 sub-interface. The QoS policy is offloaded to the satellite 100 in the L2 Fabric network.

```
config
class-map match-any class5
 match qos-group 5
 end-class-map
!
policy-map policy5
 class class5
  bandwidth percent 13
 !
interface TenGigabitEthernet 0/1/0/0.1
encapsulation dot1q 20
nv satellite-fabric-link satellite 100
remote-ports GigabitEthernet 0/0/0-5
service-policy output policy5
end or commit
```

INDEX

F

H

I

L

M

N

P

Q

R

S

T

U

V