

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, on page 1
- How to Configure Flow Aware QoS, on page 11
- Configuration Examples for Configuring Flow Aware QoS, on page 24
- Additional References, on page 26

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.

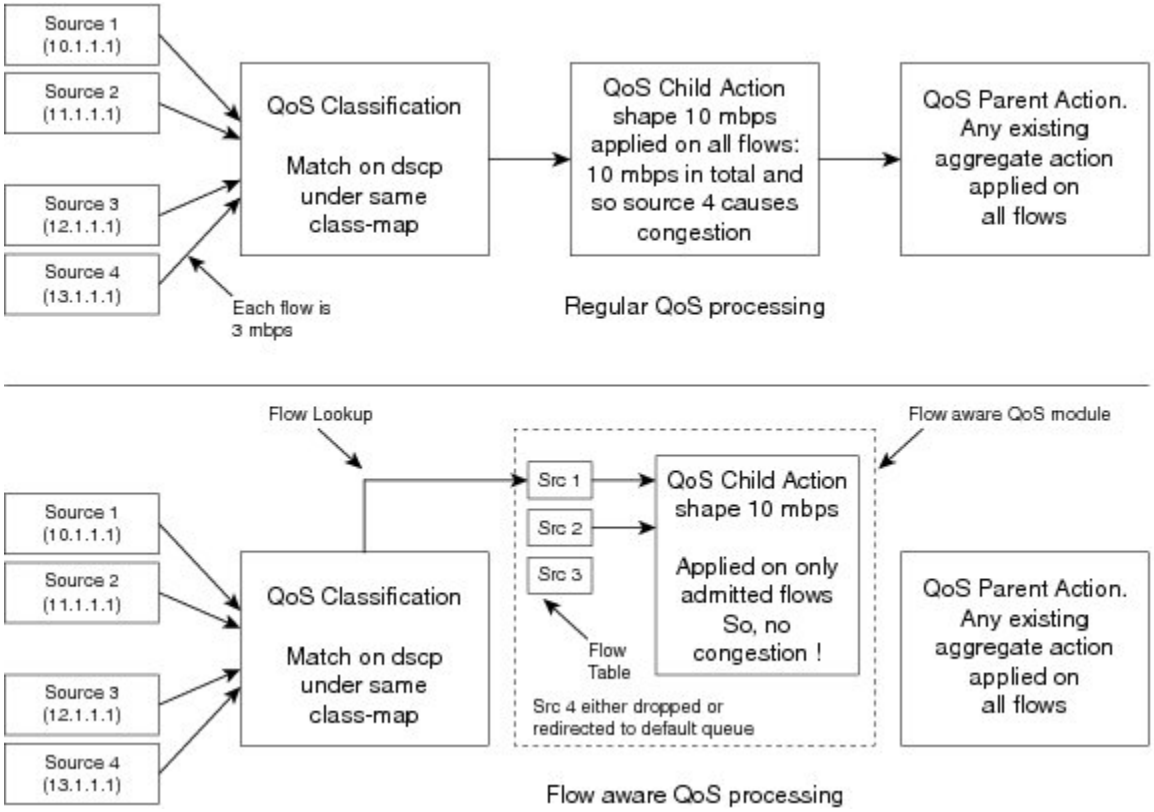
- User-based Rate Limiting (UBRL)

  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 1: Regular QoS vs Flow Aware CAC



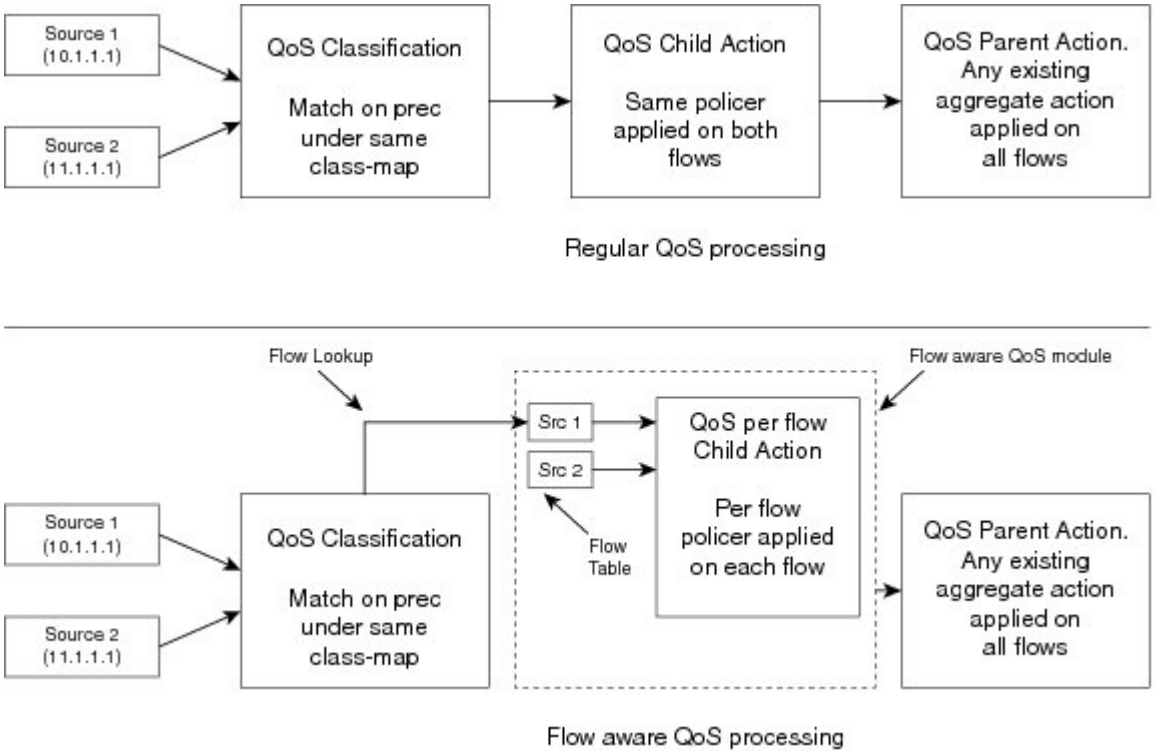
Let us assume there are 4 sources—Source 1, 2, 3, 4—with a QoS child Shape action at 10 mbps applied on all flows. If each source sends out a flow at 3 mbps, then, in the regular QoS processing, the source 4 causes congestion leading to random drop in the flow quality. However, in the Flow Aware CAC 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 2: 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.

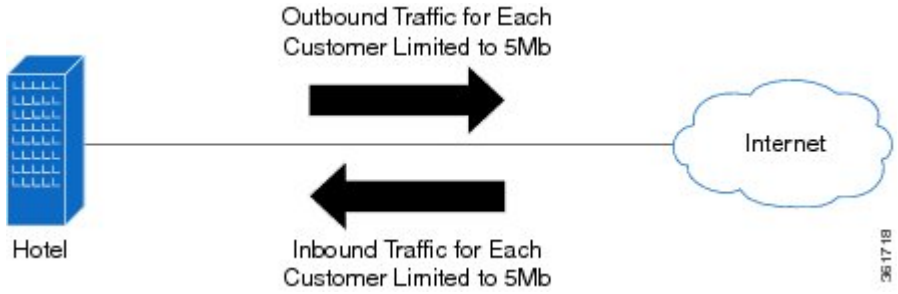
Bidirectional UBRL

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 3: 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 1: 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