Deploying the Quality of Service (QoS)

The Quality of Service (QoS) feature ensures that traffic to and from priority applications gets preference in using network resources. QoS actions are defined by service-policies that are deployed using policy-maps. During the QoS process, packets are encapsulated with QoS information. The encapsulation is monitored and accounted by the QoS accounting function.

Parameterized QoS (PQoS) is another form of QoS in which the traffic priority is based on the characteristic of the data being carried by the traffic.

BNG supports merging of multiple QoS policy-maps applied through multiple dynamic templates and implementing them on a single subscriber.

Table 1: Feature History for Configuring Subscriber Features

Release	**Modification**
Release 4.2.0	Initial release of this document.

This chapter explains deploying QoS, and covers the following topics:

Quality of Service Overview

Quality of Service (QoS) is the technique of prioritizing network traffic for time-sensitive and mission-critical applications such as VoIP services, live streaming of videos, priority accesses to database, and so on. Functions that QoS provides ensure that such applications receive sufficient bandwidth at low latency, with reduced data loss.

QoS functions perform preferential forwarding of high-priority packets. To do so, the packet is identified, classified, and then prioritized on all routers along the data-forwarding path throughout the network. As a result, priority applications get the resources they require, while other applications access the network, simultaneously.

QoS functions provide service providers cost benefits by enabling them to use existing resources efficiently and ensure the required level of service without reactively expanding, or over-provisioning their networks. QoS also improves customer experience when they get reliable services from a variety of network applications.

It is ideal to deploy QoS on BNG because BNG is present at the edge router, and subscriber directly connects to it. One of the unique features of BNG is QoS accounting. This feature enables BNG to collect and report QoS encapsulation information to the RADIUS server. For details, see QoS Accounting, on page 26.

The deployment of QoS involves three components:

- Class-map — Classifies different forms of traffic, like video, data, VOIP and so on, based on matching rules.
- Policy-map — Defines the QoS actions to be applied to the classified traffic. It references the classes previously defined in the class-map. These policy-maps are also called QoS maps. The actions defined in the policy-map perform traffic prioritization and bandwidth allocation.
- Service policy — Associates a previously defined policy-map with a attachment point and direction, on BNG. The attachment points are listed in the section QoS Attachment Points, on page 40.The two directions possible for a policy is input and output. The policy direction is relative to the attachment point.

BNG supports two-level hierarchical policy (parent policy and child policy) for deploying QoS. Based on the preference of service provider, the QoS policies are defined and applied on BNG in these ways:

- Define and apply the QoS policy from CLI. See, Configuring Service-policy and Applying Subscriber Settings Through Dynamic Template, on page 4.
- Define the QoS policy in CLI, but apply it from RADIUS. See, Configuring Service-policy and Applying Subscriber Settings Through RADIUS, on page 2.
- Define and apply the QoS policy from RADIUS. It is also called Parameterized QoS, on page 6.

Restriction

- If the subscriber ingress or egress QoS includes policing, shaping, bandwidth, or WRED actions, it is recommended that only active:standby bundle interfaces be used. Load-sharing should be avoided.
- Users can configure only 7 class-maps (both ingress and egress, excluding the class-default map) to achieve a higher scale configuration (say, 32,000 sessions) per node processor.

Configuring Service-policy and Applying Subscriber Settings Through RADIUS

Perform this task to deploy the QoS policy using CLI commands. In this task, subscriber settings are applied from the RADIUS server.

SUMMARY STEPS

1. **configure**
2. **policy-map type qos** *q_in*
3. **class class-default**

4. **service-policy** *q_child_in*
5. **policy-map type qos** *q_out*
6. **class class-default**
7. **service-policy** *q_child_out*
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map type qos** *q_in* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map type qos q_in`	Configures the policy-map for the type qos.

Step 3	**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 4	**service-policy** *q_child_in* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy q_child_in`	Applies a bottom-level policy to the top-level class-default class.
Step 5	**policy-map type qos** *q_out* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map type qos q_out`	Configures the policy-map for the type qos.

Step 6	**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 7	**service-policy** *q_child_out* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy q_child_out`	Applies a bottom-level policy to the top-level class-default class.
Step 8	**commit**	

Configuring Subscriber Policy through CLI and Applying through RADIUS: Examples

```
configure
policy-map type qos q_in
class class-default
end

\\the following procedure is ran in RADIUS
     Service-Type = Outbound-User
     Cisco-avpair = "ipv4:ipv4-mtu=750",
     Cisco-avpair = "ipv4:ipv4-unnumbered=Loopback0",
     Cisco-avpair = "subscriber:sub-qos-policy-in=q_in",
     Cisco-avpair = "subscriber:sub-qos-policy-out=q_out",
     Idle-Timeout = 1000,
     Session-Timeout = 5000
```

Configuring Service-policy and Applying Subscriber Settings Through Dynamic Template

Perform this task to deploy the QoS policy using CLI commands. In this task, subscriber settings are applied using a dynamic template.

SUMMARY STEPS

1. **configure**
2. **policy-map type qos** *q_in*
3. **class class-default**
4. **service-policy** *q_child_in*
5. **policy-map type qos** *q_out*
6. **class class-default**
7. **service-policy** *q_child_out*
8. **dynamic-template type ppp** *dynamic_config*
9. **service-policy input** *q_in*
10. **service-policy output** *q_out*
11. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**policy-map type qos** *q_in* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map type qos q_in`	Configures the policy-map in the input direction.
Step 3	**class class-default**	Configures or modifies the parent class-default class.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(config-pmap)# class class-default`	**Note** You can configure only the class-default class in a parent policy. Do not configure any other traffic class.
Step 4	**service-policy** *q_child_in* **Example:** `RP/0/RSP0/CPU0:router(config)# service-policy q_child_in`	Configures the service policy for the input direction. **Note** The q_in and q_out policy maps are parent policy maps.
Step 5	**policy-map type qos** *q_out* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map type qos q_out`	Configures the policy-map for the output direction. **Note** The q_in and q_out policy maps are parent policy maps.

Step 6	**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 7	**service-policy** *q_child_out* **Example:** `RP/0/RSP0/CPU0:router(config)# service-policy q_child_out`	Applies a bottom-level policy to the top-level class-default class. **Note** The q_in and q_out policy maps are parent policy maps.
Step 8	**dynamic-template type ppp** *dynamic_config* **Example:** `RP/0/RSP0/CPU0:router(config)# dynamic-template type ppp dynamic_config`	Configures dynamic-template of the type ppp and applies the configuration through dynamic-template.
Step 9	**service-policy input** *q_in* **Example:** `RP/0/RSP0/CPU0:router(config-dynamic-template-type)# service-policy input q_in`	Configures the service-policy in the input direction.
Step 10	**service-policy output** *q_out* **Example:** `RP/0/RSP0/CPU0:router(config-dynamic-template-type)# service-policy input q_out`	Configures the service-policy in the output direction.
Step 11	**commit**	

Configuring Subscriber Policy through CLI and Applying to Subscriber through Dynamic-Template: Examples

```
configure
policy-map type qos q_in  // policy-map input direction
class class-default
end

configure
policy-map type qos q_out  // policy-map output direction
class class-default
end

// applying configuration through dynamic-template
configure
dynamic-template type ppp dynamic_policy
service-policy input q_in
service-policy output q_out
end
```

Parameterized QoS

Parameterized Quality of Service (PQoS) guarantees reliable performance of a network application by reserving for it the required network bandwidth. In this case, the prioritization is based on the type of data being carried by the packet.

In the standard QoS, the importance of a packet is based on the priority level that is defined for it. It is possible that in once case a video packet and an asynchronous data transfer packet have the same priority level defined. In such a case, the router gives equal importance to both packets. As a result, because of bandwidth conflict, there can be video degradation.

On the other hand, in PQoS, packet importance is based on the characteristics or parameters of the data that is carried by the packet. For example, it is possible to have PQoS provide dedicated bandwidth for video packets. Even at times when heavy loads of asynchronous data traffic are introduced into the network, PQoS guarantees that video packets have priority over other data streams that do not require real-time streaming.

Parameterized QoS has the ability to define, modify, or delete QoS policy-map based Vendor Specific Attributes (VSAs). VSAs are downloaded through the RADIUS server. The attributes from the parameterized QoS policies are filtered and passed on to the policy object library; the latter parses and translates them into policy objects. The VSAs define a two-level hierarchical policy to be applied on the subscriber session. The format of the QoS VSAs is:

```
AVPair: qos-policy-in=add-class(sub,<parent-class, child-class>,<action-list>)
AVPair: qos-policy-out=add-class(sub,<parent-class, child-class>,<action-list>)
AVPair: qos-policy-in=remove-class(sub,<parent-class, child-class>)
AVPair: qos-policy-out=remove-class(sub,<parent-class, child-class>)
```

where:

- “sub”, is a constant string, signifies that the current policy on the subscriber is to be modified
- <class-list> gives the hierarchy of the class to be added or removed (i.e. parent-class, child-class)

- <action-list> gives the QoS actions to be applied under the class being added

For more information about QoS parameters and its syntax, see *Parameterized QoS Syntax* in the Configuring Parameterized QoS Policy Through RADIUS, on page 12.

When a parameterized QoS policy for a subscriber is downloaded from the RADIUS server for the first time, the VSAs are used to build the policy from scratch. After the policy is applied on the subscriber, any new or modified VSAs downloaded for that subscriber from the RADIUS server automatically modifies the already applied policy.

For deploying a Parameterized QoS policy from the RADIUS server, see Configuring Parameterized QoS Policy Through RADIUS, on page 12.

Using Change of Authorization (CoA), it is possible to update the service-policy by modifying the class-maps that were previously configured by the parameterized QoS. Modifying can involve removing existing classes, or adding new classes. To make updates to the service-policy, see Modifying Service Policy through CoA , on page 14.

Parameterized QoS Syntax

Parameterized QoS Syntax

QoS Action Parameter	Qualifiers	Commands
Shape	QoS Action	shape(<rate-in-kbps>)
	CLI Equivalent	shape average <shape-rate> <kbps>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default),shape(14700))
Shape in percentage	QoS Action	Shape-rpct(<rate-in-pct>)
	CLI Equivalent	shape average percent < rate-in-pct >
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default),shape-pct(25))
Police (Variant 1)	QoS Action	police(<conform-rate-in-kbps>, <conform-burst-in-kBytes>, <exceed-rate-in-kbps>, <exceed-burst-in-kbytes>, <conform-action>, <exceed-action>, <violate-action>)

QoS Action Parameter	Qualifiers	Commands
	CLI Equivalent	police rate <conform-rate> <kbps> burst <conform-burst> <kbps> peak-rate <exceed-rate> exceed-burst <exceed-burst> conform-action <action> exceed-action <action> violate-action <action>
	RADIUS Equivalent - Example	qos-policy-in:add-class(sub,(class-default, voip),police(2000,2000, 4000, 4000,transmit, set-ipprec(< precedence>), drop))
Police (Variant 2)	QoS Action	Police (<conform-rate-in-kbps>)
	CLI Equivalent	police rate <kbps>
	RADIUS Equivalent - Example	qos-policy-in:add-class(sub,(class-default, voip), police(200000))
Police in percentage (Variant 1)	QoS Action	police-rpct(<conform-rate-in-pct>, <conform-burst-in-us>, <exceed-rate-in-pct>, <exceed-burst-in-us>, <conform-action>, <exceed-action>, <violate-action>)
	CLI Equivalent	police rate percentage <pct> burst <conform-burst> < us> peak-rate percentage<pct> exceedburst <exceed-burst> conform-action <action> exceed-action <action> violate-action <action>
	RADIUS Equivalent - Example	qos-policy-in:add-class(sub,(class-default, voip),police-rpct(20,20, 40, 40,transmit, set-ipprec(< precedence>), drop))
Police in percentage (Variant 2)	QoS Action	Police-rpct(<conform-rate-in-pct>

QoS Action Parameter	Qualifiers	Commands
	CLI Equivalent	police rate percentage <pct>
	RADIUS Equivalent - Example	qos-policy-in:add-class(sub,(class-default, voip), police-rpct(20))
Set IP Precedence	QoS Action	set-ip-prec(<precedence>)
	CLI Equivalent	set precedence <precedence>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-ip-prec(5))
Set CoS	QoS Action	set-cos(<cos-val>)
	CLI Equivalent	set cos <cos-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-cos(5))
Minimum Bandwidth	QoS Action	bw-abs(<bw-in-kbps>)
	CLI Equivalent	bandwidth <bw-in-kbps>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,video),bw-abs(2000))
Minimum bandwidth percentage	QoS Action	bw-pct(<bw-in-pct>)
	CLI Equivalent	bandwidth percent <pct>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,video),bw-abs(2000))
Bandwidth Remaining Percentage	QoS Action	bw-rpct(<pct>)
	CLI Equivalent	bandwidth remaining percent <pct>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip),bw-rpct(33))
Set IP DSCP	QoS Action	set-ip-dscp(<dscp-val>)

QoS Action Parameter	**Qualifiers**	**Commands**
	CLI Equivalent	Set dscp <dscp-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-ip-dscp(46))
Queue Limit in packets	QoS Action	queue-limit(<qlimit-in-packets>)
	CLI Equivalent	queue-limit <val> < packets>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip),queue-limit(64))
Queue Limit in us	QoS Action	queue-limit-us(<qlimit-in-us>)
	CLI Equivalent	queue-limit <val> <us>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip),queue-limit-us(240))
DSCP based WRED	QoS Action	random-detect-dscp(<dscp>, <min-threshold>, <max-threshold>, <probability>)
	CLI Equivalent	random-detect dscp <dscp-val> < Min-thresh> <Kbytes> <max-thresh> <Kbytes> probability < probability-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), random-detect-dscp (24, 25000, 35000))
Precedence based WRED	QoS Action	random-detect-prec (<precedence>, <min-threshold>, <max-threshold>, <probability>)
	CLI Equivalent	random-detect precedence <prec-val> < Min-thresh> <Kbytes> <max-thresh> <Kbytes> probability < probability-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), random-detect- (24, 25000, 35000))
Set qos group	QoS Action	set-qos-grp(<group-val>)
	CLI Equivalent	set qos-group <qos-group-val>

QoS Action Parameter	Qualifiers	Commands
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-qos-grp (24))
Priority Level	QoS Action	pri-level(<priority-level>)
	CLI Equivalent	priority level <priority-level>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default, voip), pri_level(1))
Set discard class	QoS Action	set-dclass(<discard-class-val>)
	CLI Equivalent	set discard-class <discard-class-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-dclass (4))
Set MPLS exp topmost bit	QoS Action	set-mpls-exp-topmost (<mpls-exp- topmost-val>)
	CLI Equivalent	set mpls experimental topmost <mpls-exp- topmost-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-mpls-exp-topmost (4))
Set MPLS exp imposition bit	QoS Action	set-mpls-exp- imposition (<mpls-exp-imposition-val>)
	CLI Equivalent	set mpls experimental imposition <mpls-exp- imposition-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-mpls-exp-imposition (4))
Set Tunnel precedence	QoS Action	set-tunnel-prec(<prec-val>)
	CLI Equivalent	set precedence tunnel <precedence-val>
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-tunnel-prec(4))
Set Tunnel DSCP	QoS Action	set-tunnel-dscp (<dscp-val>)
	CLI Equivalent	set dscp tunnel <dscp-val>

QoS Action Parameter	Qualifiers	Commands
	RADIUS Equivalent - Example	qos-policy-out:add-class(sub,(class-default,voip), set-tunnel-dscp(4))

Configuring Parameterized QoS Policy Through RADIUS

Perform this task to deploy parameterized QoS policy and apply subscriber settings through the RADIUS server. These steps are performed on the RADIUS server for each subscriber.

Note

- Parameterized QoS configuration through the RADIUS server is applicable only for user-profiles; not for service-profiles.
- In parameterized QoS configuration, the policy-map is not defined on the CLI. It is dynamically created based on the configuration passed through RADIUS. This procedure applies to the RADIUS server as part of RADIUS user configurations. The policy-map results are applied to the subscriber when that user profile is downloaded after executing a control policy authentication or authorization action. The class-map must be configured through CLI. For this task, the *classes voice_in*, *video_in*, *data_in*, *video_out*, *voice_out*, and *data_out* are configured separately.

SUMMARY STEPS

1. **Cisco-AVPair** = *"ip:qos-policy-in=add-class(sub, (class-default),police(2000))"*
2. **Cisco-AVPair**+= *"ip:qos-policy-in=add-class(sub, (class-default,voice_in), pri-level(1), police(256))"*
3. **Cisco-AVPair**+= *ip:qos-policy-in=add-class(sub, (class-default,video_in), pri-level(2), police(1000))"*
4. **Cisco-AVPair** += *"ip:qos-policy-in=add-class(sub, (class-default,data_in), set-qos-grp(4))"*
5. **Cisco-AVPair** += *"ip:qos-policy-in=add-class(sub, (class-default,class-default), set-qos-grp(7))"*
6. **Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default), shape(4000))"*
7. **Cisco-AVPair** += *"ip:qos-policy-out=add-class(sub, (class-default,voice_out), pri-level(1),queue-limit-us(10000))"*
8. **Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default,video_out),queue-limit-us(30000), shape(2000))"*
9. **Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default,data_out), bw-rpct(20))"*
10. **Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default,class-default))"*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**Cisco-AVPair** = *"ip:qos-policy-in=add-class(sub, (class-default),police(2000))"* **Example:** `Cisco-AVPair = "ip:qos-policy-in=add-class(sub, (class-default),police(2000))"`	Configures the cisco-avpair class-map in input direction for police action parameter.

	Command or Action	Purpose
Step 2	**Cisco-AVPair**+= *"ip:qos-policy-in=add-class(sub, (class-default,voice_in), pri-level(1), police(256))"* **Example:** `Cisco-AVPair =  "ip:qos-policy-in=add-class(sub, (class-default,voice_in), pri-level(1), police(256))"`	Configures the cisco-avpair class-map in input direction for the police action parameter.
Step 3	**Cisco-AVPair**+= *ip:qos-policy-in=add-class(sub, (class-default,video_in), pri-level(2), police(1000))"* **Example:** `Cisco-AVPair = ip:qos-policy-in=add-class(sub, (class-default,video_in), pri-level(2), police(1000))"`	Configures the cisco-avpair class-map in input direction for the police action parameter.
Step 4	**Cisco-AVPair** += *"ip:qos-policy-in=add-class(sub, (class-default,data_in), set-qos-grp(4))"* **Example:** `Cisco-AVPair = "ip:qos-policy-in=add-class(sub, (class-default,data_in), set-qos-grp(4))"`	Configures the cisco-avpair class-map in input direction for the police action parameter.
Step 5	**Cisco-AVPair** += *"ip:qos-policy-in=add-class(sub, (class-default,class-default), set-qos-grp(7))"* **Example:** `Cisco-AVPair = "ip:qos-policy-in=add-class(sub, (class-default,class-default), set-qos-grp(7))"`	Configures the cisco-avpair class-map in input direction for the set qos action parameter.
Step 6	**Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default), shape(4000))"* **Example:** `Cisco-AVPair = "ip:qos-policy-out=add-class(sub, (class-default), shape(4000))"`	Configures the cisco-avpair class-map in output direction for the shape action parameter.
Step 7	**Cisco-AVPair** += *"ip:qos-policy-out=add-class(sub, (class-default,voice_out), pri-level(1),queue-limit-us(10000))"* **Example:** `Cisco-AVPair = "ip:qos-policy-out=add-class(sub, (class-default,voice_out), pri-level(1),queue-limit-us(10000))"`	Configures the cisco-avpair class-map in output direction for the queue-limit-us action parameter.
Step 8	**Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default,video_out),queue-limit-us(30000), shape(2000))"*	Configures the cisco-avpair class-map in output direction for the queue-limit-us and the shape action parameters.

	Command or Action	Purpose
	Example: `Cisco-AVPair = "ip:qos-policy-out=add-class(sub, (class-default,video_out),queue-limit-us(30000), shape(2000))"`	
Step 9	**Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default,data_out), bw-rpct(20))"* **Example:** `Cisco-AVPair = "ip:qos-policy-out=add-class(sub, (class-default,data_out), bw-rpct(20))"`	Configures the cisco-avpair class-map in output direction for the bandwidth action parameter.
Step 10	**Cisco-AVPair**+= *"ip:qos-policy-out=add-class(sub, (class-default,class-default))"* **Example:** `Cisco-AVPair = "ip:qos-policy-out=add-class(sub, (class-default,class-default))"`	Configures the cisco-avpair class-map in output direction for the class action parameter. **Note** For the complete list of QoS action parameters that can be configured and applied through RADIUS, see *Parameterized QoS Syntax* section in Parameterized QoS Syntax, on page 7.

Configuring Parameterized Subscriber Policy Defined and Applied through RADIUS: An example

```
  Cisco-AVPair = "ip:qos-policy-in=add-class(sub, (class-default),police(2000))"
  Cisco-AVPair += "ip:qos-policy-in=add-class(sub, (class-default,voice_in), pri-level(1),
 police(256))"
  Cisco-AVPair += "ip:qos-policy-in=add-class(sub, (class-default,video_in), pri-level(2),
 police(1000))"
  Cisco-AVPair += "ip:qos-policy-in=add-class(sub, (class-default,data_in), set-qos-grp(4))"

  Cisco-AVPair += "ip:qos-policy-in=add-class(sub, (class-default,class-default),
set-qos-grp(7))"
  Cisco-AVPair += "ip:qos-policy-out=add-class(sub, (class-default), shape(4000))"
  Cisco-AVPair += "ip:qos-policy-out=add-class(sub, (class-default,voice_out),
pri-level(1),queue-limit-us(10000))"
  Cisco-AVPair += "ip:qos-policy-out=add-class(sub,
(class-default,video_out),queue-limit-us(30000), shape(2000))"
  Cisco-AVPair += "ip:qos-policy-out=add-class(sub, (class-default,data_out), bw-rpct(20))"

  Cisco-AVPair += "ip:qos-policy-out=add-class(sub, (class-default,class-default))"
```

Modifying Service Policy through CoA

Perform this task to modify service-policy through CoA.

Note The Web Portal or Radius server that supports CoA should be configured to generate a CoA request with Cisco VSA corresponding to the steps in this task.

SUMMARY STEPS

1. **qos-policy-out** *remove-class(sub, (class-default, voip))*
2. **qos-policy-out** *add-class(sub, (class-default, video), bw-rpct(50), pri-level(2))*
3. **qos-policy-out** *add-class(sub, (class-default, data), shape(400),set-ip-prec(1))*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**qos-policy-out** *remove-class(sub, (class-default, voip))* **Example:** `qos-policy-out=remove-class(sub, (class-default, voip))`	Removes the class map, where voip is the class to be removed from a previously configured parameterized QoS for a subscriber.
Step 2	**qos-policy-out** *add-class(sub, (class-default, video), bw-rpct(50), pri-level(2))* **Example:** `qos-policy-out=add-class(sub, (class-default, video), bw-rpct(50), pri-level(2))`	Adds a class map, where video is the class to be added to a previously configured parameterized QoS for a subscriber.
Step 3	**qos-policy-out** *add-class(sub, (class-default, data), shape(400),set-ip-prec(1))* **Example:** `qos-policy-out=add-class(sub, (class-default, data), shape(400),set-ip-prec(1))`	Configures the qos-policy-out for shape, set ip precedence parameters.

Modifying Service Policy through CoA : Examples

```
//Policy-map configuration before CoA
policy-map __sub_5e311c4f_child1
 class voip
  priority level 1
  police rate 10000 kbps burst 8 kbytes
  !
!
class video
  priority level 1
  police rate 10000 kbps burst 16 kbytes
  !
!
class data
  shape average 80000 kbps
!
class class-default
!
end-policy-map
!
policy-map __sub_5e311c4f
 class class-default
  service-policy __sub_5e311c4f_child1
  shape average 100000 kbps
!
end-policy-map
!

//Modifying Service Policy through CoA
qos-policy-out=remove-class(sub, (class-default, voip))
qos-policy-out=add-class(sub, (class-default, video), bw-rpct(50), pri-level(2))
qos-policy-out=add-class(sub, (class-default, data), shape(400),set-ip-prec(1))

//Policy-map configuration after CoA looks like:
 policy-map __sub_ffffffec1a37f_child1
class video
priority level 2
  bandwidth percent 50
  police rate 10000 kbps burst 16 kbytes
  !
!
class data
  shape average 400 kbps
  set precedence 1
!
class class-default
!
end-policy-map
!
policy-map __sub_ffffffec1a37f
class class-default
  service-policy __sub_ffffffec1a37f_child1
  shape average 100000 kbps
!
end-policy-map
!
```

Parameterized QoS for Line Card Subscribers

From Cisco IOS XR Release 5.3.2 and later, parameterized QoS (PQoS) as auto-service is supported for LC subscribers, along with RP subscribers. For PQoS as auto-service, all the PQoS attributes are defined as VSAs in the service profile, and activated as auto-service from the user profile. The regular mode of PQoS, where the attributes are defined in user profile and activated by a service logon CoA request, is not supported for LC subscribers. Whereas, RP subscribers support both modes of PQoS.

In user profile-based PQoS, the entire set of Cisco-AVPairs needs to be downloaded every time a new session comes up. Whereas, for PQoS as auto-service, the attributes need to be downloaded only for the first session. If the same service is to be activated for the next session, the attributes that were downloaded earlier for the previous session can be used from the BNG router itself. This reduces the processing time considerably and provides more flexibility in activating and deactivating a service.

To deactivate a PQoS service, use the service-logoff request irrespective of the way it was activated. To modify the PQoS feature per subscriber session, send a multi-action CoA request with a deactivation command for the active service (**cisco-avpair += "subscriber:sd=<old-service>"**) and an activation command for the new service (**cisco-avpair += "subscriber:sa=<new-service>"**). To modify the service definitions which are currently used by the session, send the service update CoA request with new parameters.

Configuring Parameterized QoS as Auto-service

Configuration Guidelines

- For each service in the user profile, there must be a corresponding **Method-List** specified. Else, the BNG router considers that the service profile is defined locally.
- Once you download pqos as auto-service from the RADIUS server, the only way to change the service definition in the router, is through a CoA service-update request.
- The CoA account status query might not reply the **echo-strings** for the service.
- While a session starts, the user might want to apply default QoS service apart from the PQoS service. In such cases, ensure that the default QoS profile is applied as service template and activated after the authentication action. This avoids multiple instances of apply and undo apply during session bring-up, thereby providing good bring-up calls-per-second (CPS). It also avoids unnecessary feature installation for access-rejected users as well.

Configuration of PQoS as Auto-service: Example

This example shows a sample user profile and service profile, to activate the services 1_Mbps_IN and 1_Mbps_OUT as auto-service:

User Profile:

```
BNGuser1@bngtm.com Cleartext-Password := cisco
  service-Type=Framed-User,
  Cisco-AVPair += "echo-string-1=1_Mbps_IN",
  Cisco-AVPair += "echo-string-2=1_Mbps_OUT",
  Framed-Filter-Id = ACL_VOZ_CONTROL_IN.in,
  Cisco-avpair += "subscriber:sa=1_Mbps_IN”,
  Cisco-AVPair += "Method-List=default"
  Cisco-avpair += "subscriber:sa=1_Mbps_OUT",
  Cisco-AVPair += "Method-List=default
```

Service Profile:

```
1_Mbps_IN  Cleartext-Password := "cisco"
       Cisco-AVPair += "ip:qos-policy-in=add-class(sub,(class-default),police(1085))",
       Cisco-AVPair +=
"ip:qos-policy-in=add-class(sub,(class-default,BROADBAND_VOZ),police(512,transmit,drop),set-mpls-exp-imposition(5))",

       Cisco-AVPair +=
"ip:qos-policy-in=add-class(sub,(class-default,BROADBAND_CRITICOS),set-mpls-exp-imposition(1))",

       Cisco-AVPair +=
"ip:qos-policy-in=add-class(sub,(class-default,BROADBAND_BUSINESS),set-mpls-exp-imposition(1))",

       Cisco-AVPair +=
"ip:qos-policy-in=add-class(sub,(class-default,class-default),set-mpls-exp-imposition(0),set-ip-dscp(0))”

1_Mbps_OUT  Cleartext-Password := "cisco"
       Cisco-AVPair += "ip:qos-policy-out=add-class(sub,(class-default),shape(1064))",
       Cisco-AVPair +=
"ip:qos-policy-out=add-class(sub,(class-default,BROADBAND_VOZ),police(512,transmit,drop),pri-level(1),set-cos(5))",

       Cisco-AVPair +=
"ip:qos-policy-out=add-class(sub,(class-default,BROADBAND_CRITICOS),bw-rpct(50),set-cos(1))",
       Cisco-AVPair +=
"ip:qos-policy-out=add-class(sub,(class-default,BROADBAND_BUSINESS),bw-rpct(35),set-cos(1))",

       Cisco-AVPair +=
"ip:qos-policy-out=add-class(sub,(class-default,class-default),bw-rpct(15),set-cos(0))"
```

Single CoA Request: Example

This example shows a single CoA request to activate a PQoS service:

```
echo "Acct-Session-Id=08000001,Cisco-avpair+='subscriber:sa=pQOS_SVC_1MIN',
Cisco- AVPair+='Method-List=default'" | /usr/local/bin/radclient -x 6.6.6.18:1500 coa cisco
 -r 1
Sending CoA-Request of id 134 to 6.6.6.18 port 1500 Acct-Session-Id = "08000001"
Cisco-AVPair += "subscriber:sa=pQOS_SVC_1MIN"
Cisco-AVPair += "Method-List=default"
rad_recv: CoA-ACK packet from host 6.6.6.18 port 1500, id=134, length=50
Cisco-AVPair = "sa=pQOS_SVC_1MIN”
```

This example shows a single CoA request to deactivate a PQoS service:

```
echo "Acct-Session-Id=08000001,Cisco-avpair+='subscriber:sd=pQOS_SVC_1MIN',
Cisco- AVPair+='Method-List=default'" | /usr/local/bin/radclient -x 6.6.6.18:1500 coa cisco
 -r 1
Sending CoA-Request of id 21 to 6.6.6.18 port 1500 Acct-Session-Id = "08000001"
Cisco-AVPair += "subscriber:sd=pQOS_SVC_1MIN"
Cisco-AVPair += "Method-List=default"
rad_recv: CoA-ACK packet from host 6.6.6.18 port 1500, id=21, length=50
Cisco-AVPair = "sd=pQOS_SVC_1MIN"
```

Multi-action CoA Request: Example

This example shows a sample multi-action CoA request used to deactivate the service, pQOS_SVC_1MOUT and to activate the service, pQOS_SVC_2MOUT. It also updates the corresponding echo-string in single CoA request:

```
echo "Acct-Session-Id=080043e5,Cisco-Avpair+='subscriber:sd=pQOS_SVC_1MOUT',cisco-
avpair+='Method-List=default',Cisco-AVPair+='echo-string-2=2_Mbps_OUT',
Cisco- avpair+='subscriber:sa=pQOS_SVC_2MOUT',cisco-avpair+='Method-List=default'" |
/usr/local/bin/radclient -x 6.6.6.18:1500 coa cisco

Sending CoA-Request of id 77 to 6.6.6.18 port 1500 Acct-Session-Id = "080043e5"

Cisco-AVPair += "subscriber:sd=pQOS_SVC_1MOUT" Cisco-AVPair += "Method-List=default"
Cisco-AVPair += "echo-string-1=1_Mbps_OUT"
Cisco-AVPair += "echo-string-2=2_Mbps_OUT" Cisco-AVPair += "subscriber:sa=pQOS_SVC_2MOUT"
Cisco-AVPair += "Method-List=default"
rad_recv: CoA-ACK packet from host 6.6.6.18 port 1500, id=77, length=80 Cisco-AVPair =
"sd=pQOS_SVC_1MOUT"  Cisco-AVPair = "sa=pQOS_SVC_2MOUT"
```

Service-update CoA Request: Example

This example shows a sample service-update CoA request to modify the parameters of a policy-map that is active on the BNG router:

```
echo "cisco-avpair+='subscriber:command=service-update',Cisco-
avpair+='subscriber:service-name=pQOS_SVC_1MIN',Cisco-AVPair+='Method- List=default',
Cisco-AVPair+='ip:qos-policy-in=add-class(sub,(class-
default),police(1085))',Cisco-AVPair+='ip:qos-policy-in=add-class(sub,
(class- default,BROADBAND_VOZ),police(512,transmit,drop),set-mpls-exp-imposition(4))',
Cisco- AVPair+='ip:qos-policy-in=add-class(sub,(class-default,BROADBAND_CRITICOS),set-mpls-
 exp-imposition(2))',
Cisco-AVPair+='ip:qos-policy-in=add-class(sub,(class-
default,BROADBAND_BUSINESS),set-mpls-exp-imposition(2))',
Cisco-AVPair+='ip:qos-policy-
in=add-class(sub,(class-default,class-default),set-mpls-exp-imposition(0),set-ip- dscp(0))'"
 | /usr/local/bin/radclient -x 6.6.6.18:1500 coa cisco

Sending CoA-Request of id 173 to 6.6.6.18 port 1500 Cisco-AVPair +=
"subscriber:command=service-update"

Cisco-AVPair += "subscriber:service-name=pQOS_SVC_1MIN" Cisco-AVPair += "Method-List=default"
Cisco-AVPair += "ip:qos-policy-in=add-class(sub,(class-default),police(1085))"
Cisco-AVPair += "ip:qos-policy-in=add-class(sub,(class-
default,BROADBAND_VOZ),police(512,transmit,drop),set-mpls-exp-imposition(5))"
Cisco-AVPair += "ip:qos-policy-in=add-class(sub,(class-
default,BROADBAND_CRITICOS),set-mpls-exp-imposition(1))"
Cisco-AVPair += "ip:qos-policy-in=add-class(sub,(class-
default,BROADBAND_BUSINESS),set-mpls-exp-imposition(1))"
Cisco-AVPair += "ip:qos-policy-in=add-class(sub,(class-default,class-
default),set-mpls-exp-imposition(0),set-ip-dscp(1))”

rad_recv: CoA-ACK packet from host 6.6.6.18 port 1500, id=173, length=20
```

Verifying PQoS Configuration

You can use these show commands to verify the PQoS configuration:

- Verify if all the policy parameters, like policer and shaper values received from the RADIUS server, are applied on the interface:

```
router# show policy-map applied interface GigabitEthernet0/0/0/0.1.pppoe4

Input policy-map applied to GigabitEthernet0/0/0/0.1.pppoe4:

  policy-map __sub_655b501d
   class class-default
    service-policy __sub_655b501d_child1
    police rate 2085 kbps
    !
   !

Child policy-map(s) of policy-map __sub_655b501d:

  policy-map __sub_655b501d_child1
   class BROADBAND_VOZ
    police rate 512 kbps
     conform-action transmit
     exceed-action drop
    !
    set mpls experimental imposition 5
   !
   class BROADBAND_CRITICOS
    set mpls experimental imposition 1
   !
   class BROADBAND_BUSINESS
```

```
    set mpls experimental imposition 1
   !
   class class-default
    set mpls experimental imposition 0
    set dscp 0
   !
   end-policy-map
  !
```

- Verify the applied service(s) for the session:

```
router# show subscriber session all detail internal

Interface:                GigabitEthernet0/0/0/0.1.pppoe4
- - -
- - -
Policy Executed:

  event Session-Start match-first [at Wed May 13 10:40:43 2015]
    class type control subscriber PPP_CM do-until-success [Succeeded]
      10 activate dynamic-template PTA_TEMPLATE_1 [cerr: No error][aaa: Success]
  event Session-Activate match-first [at Wed May 13 10:40:43 2015]
    class type control subscriber PPP_CM do-all [Succeeded]
      10 authenticate aaa list default [cerr: No error][aaa: Success]
      20 activate dynamic-template DEF_SEVICE [cerr: No error][aaa: Success]
Session Accounting:
 Acct-Session-Id:          10000003
 Method-list:              default
 - - -
 - - -
Last COA request received: unavailable
User Profile received from AAA:
 Attribute List: 0x1000f524
1:  service-type    len=  4  value= Framed
2:  inacl           len= 18  value= ACL_VOZ_CONTROL_IN
Services:
  Name        : PTA_TEMPLATE_1
  Service-ID  : 0x4000002
  Type        : Template
  Status      : Applied
---------------------------
  Name        : 2_Mbps_IN
  Service-ID  : 0x400001d
  Type        : Profile
  Status      : Applied
```

RADIUS Based Policing - QoS Shaper Parameterization

Radius Based Policing (RaBaPol) allows customized parameters, instead of the default parameters, to be used to activate BNG subscriber services. BNG supports parameterization of QoS **shape-rate**. The shaper parameters can either be sent to BNG by the RADIUS server during connection establishment, as CISCO VSAs in an Access Accept message, or they can be sent to BNG as part of the CoA messages.

To configure QoS Shaper Parameterization, use the shape average $*var_name* = *value* command in policy-map class configuration mode.

According to RaBaPol, the dynamic template associated with the subscriber contains individual feature configuration. The syntax and semantics of parameterization is feature dependent. For QoS, a dollar sign ($) is added as a prefix to the **shape-rate** variable, and the default value, along with the variables, is configured in the policy-map definition.

If the service that is to be activated is already associated to the subscriber, the incoming variable-list is compared with the exiting one. If the variable-list is the same, then this is a duplicate request and the request gets dropped. Otherwise, the old variable-list is cached and the new variable-list is associated to the subscriber. After the service is successfully activated, the iEdge echoes the VSA that trigged the service-activate, as an acknowledgment back to the AAA server.

If any feature returns an error during its activation, the iEdge component rollbacks all features to their previous states. If the feature or service has a variable-list associated with it, then that variable-list is also rolled back to the previous cached variable-list.

RaBaPol also supports policy merge, where QoS policies from multiple dynamic templates (configured through CLI or downloaded from AAA server) are merged for the subscriber.

High Availability - In the case of process restart, the session is re-established using the variable-list that is already associated with the service.

Sample Configuration and Use Cases for QoS Shaper Parameterization

Sample Configuration for QoS Shaper Parameterization

This is a sample configuration for QoS Shaper Parameterization:

```
dynamic-template type service SERVICE-POLICY-OUT
    service-policy output out-policy merge 10

policy-map out-policy
    class class-default
       shape average $shape-rate= 100000 Kbps
service-policy output-child
policy-map output-child
      class class-default
```

In this example, the service named SERVICE-POLICY-OUT has QoS features enabled. This dynamic template has outgoing QoS policies configured, with a default value of **shape-rate** being 100 Mbps.

Use Cases for QoS Shaper Parameterization

These are some use cases for QoS Shaper Parameterization:

- User initiates a subscriber session with this user profile:

```
user-cpe-xyz1@abc.com         Password="abc"
        Framed-Protocol=PPP,
        Service-Type=Framed-User
        …..
        Cisco-avpair = "subscriber:sa=SERVICE-POLICY-OUT(shape-rate=1203000)”
```

  The AAA server sends to BNG an Access-accept message that contains the service name that is to be activated (SERVICE-POLICY-OUT, in this example), action type (subscriber:sa) , and the variable list, along with its values. Now, the service name maps with the dynamic-template defined on BNG. The

VSA contains QoS **shape-rate** value (For example, shape-rate=1203000) to override the default values locally configured on BNG. In BNG, the policy gets merged with default and customized values. For the variables that were not specified in the AAA message, default values are retained.

Alternatively, the new service activation can be performed using CoA. In this case, the old policy is removed and the new, merged policy gets configured in the hardware.

- User wants to change the QoS shaper value of the subscriber. This can be ideally be done in two ways:
    - Service-modify of same service - This is currently not supported.
    - Service-activate of the new service followed by Service-deactivate of the old service - At first, a new service is activated using the new shaper value sent through the Access-accept message. After that, a CoA message is sent from the AAA server to the BNG, to deactivate the old service.

Verification of QoS Shaper Parameterization Configurations

These show commands can be used to verify the QoS Shaper Parameterization configurations in BNG:

SUMMARY STEPS

1. **show policy-map interface all**
2. **show policy-map applied interface** *interface-type interface-name*
3. **show running-configuration policy-map**
4. **show qos-ea interface** *interface-type interface-name*

DETAILED STEPS

Step 1 **show policy-map interface all**

Displays the QoS shaper rate configured on the subscriber interface by the AAA server, either through an Access-Accept message or through a CoA message. The statistics rate field, transmitted, displays the shaper rate.

Example:

```
RP/0/RSP0/CPU0:router#
show policy-map interface all
node0_1_CPU0: Service Policy not installed
node0_0_CPU0: Service Policy not installed
node0_RSP1_CPU0: node0_RSP0_CPU0:
Bundle-Ether1.1.pppoe62151: policy-parent
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                             : 458
    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
```

Step 2 **show policy-map applied interface** *interface-type interface-name*

Displays the actual policy-map applied on the subscriber interface.

Example:

```
RP/0/RSP0/CPU0:router#
show policy-map applied interface Bundle-Ether1.1.pppoe62151
Output policy-map applied to Bundle-Ether1.1.pppoe62151:
   policy-map policy-parent
   class class-default
   service-policy policy-child
    shape average $shaperP = 500 mbps
   !
 Child policy-map(s) of policy-map policy-parent:

  policy-map policy-child
   class prec2
    shape average $shaperC1 = 600 kbps
   !
   class prec3
    shape average $shaperC2 = 700 kbps
   !
   class class-default
    shape average $shaperC3 = 200 kbps
   !
   end-policy-map
```

Step 3 **show running-configuration policy-map**

Displays the details of the policy-map configured on BNG.

Example:

```
RP/0/RSP0/CPU0:router#
show running-configuration policy-map
policy-map policy-parent
class class-default
  service-policy policy-child
  shape average $shaperP = 500 mbps
!
end-policy-map
!

policy-map policy-child
class prec2
  shape average $shaperC1 = 600 kbps
!
class prec3
  shape average $shaperC2 = 700 kbps
!
class class-default
  shape average $shaperC3 = 200 kbps
!
end-policy-map
!
```

Step 4 **show qos-ea interface** *interface-type interface-name*

Displays the QoS programmed in the hardware.

Example:

```
RP/0/RSP0/CPU0:router#
show qos-ea interface bundle-Ether 1.4 output member gigabitEthernet 0/1/0/0

Interface: GigabitEthernet0_1_0_0 output policy: vlan_policy_egress
Total number of classes:   1
Total number of UBRL classes:  0
Total number of CAC classes:   0
-------------------------------------------------------
Policy name: vlan_policy_egress
Hierarchical depth 1
Interface type VLAN Subif
Interface rate 1000000 kbps
Port Shaper rate 0 kbps
Interface handle 0x00096060
ul_ifh 0x060000C0, ul_id  0x00000000
uidb index 0x001B
qos_ifh 0x810800000001b
Local port 0, NP 0
Policy map id 0x2014, format 16, uidb index 0x001B
-------------------------------------------------------
 Index 0 Level 0 Class name class-default service_id 0x0 Policy name vlan_policy_egress
 Node flags: LEAF Q_LEAF DEFAULT DEFAULT-ALL
 Stats flags: Queuing enabled
 Node Config:
 Shape: CIR/CBS/PIR/PBS: 0kbps/11250000B/900000kbps/11250000B
 WFQ: BW/Sum of BW/Excess ratio: 0kbps/0kbps/1
 Queue limit 11250000 Guarantee 0
 Node Result: Class-based stats:Stat ID 0x005114F3
 Queue: Q-ID 0x00030062 Stat ID(Commit/Excess/Drop): 0x006E01EA/0x00000000/0x006E01EB
-------------------------------------------------------
```

Supported Scenarios of QoS Shaper Parameterization

These scenarios are supported for QoS Shaper Parameterization:

- Merging of QoS policies through AAA server is supported - A subscriber session does not come up if only one of the policies (applied either through dynamic template control policy or through RADIUS server) has the **merge** keyword enabled. This is irrespective of whether the shaper parameterization is enabled, or not.

  In the case of re-configuring through a CoA message:

  - If only one of the policies has the **merge** keyword enabled, the policy is rejected irrespective of whether the shaper parameterization is enabled, or not.
  - If none of the policies have the **merge** keyword enabled, the policy applied through the RADIUS server replaces the one applied through the control subscriber policy.
  - If both the policies have the **merge** keyword enabled, the policy is accepted irrespective of whether the shaper parameterization is enabled, or not.

- These service policy replacement scenarios are supported:

  - A subscriber, with service policy A (applied through dynamic template) having default shaper parameterized values, which is replaced by service policy B having shaper parameterized values.
  - A subscriber, with service policy A (applied through dynamic template) having default shaper parameterized values, which is replaced by service policy B with no shaper parameterized values.

Restrictions of QoS Shaper Parameterization

The QoS Shaper Parameterization is subjected to these restrictions:

- Parameterization of only the QoS *shape-rate* feature associated with the subscriber service is supported; other features are not supported. For the *shape-rate* variable, the parameterization of only the value attribute is supported; the parameterization of rate units (such as kbps, mbps, and so on) including excess burst size, is not supported.
- Linecard (LC) subscribers are not supported.
- The service profile downloaded from the RADIUS server is not supported.
- The addition or modification of the *shape-rate* variable field is rejected for any policy-map that is applied on an interface.
- The modification of the default *shape-rate* variable value is rejected for a policy-map that is applied on an interface.
- The variable names in the policy-map definition must be different across the system.
- A service which is defined on BNG, without parameterization enabled, does not accept parameters through CoA.
- Service modification of variable-list is not supported.
- Parameterized shapers are not supported with multi-action CoA.
- The maximum number of different variable-lists supported is 2000. This limit includes the already-active sessions wherein the previous variable-list is also stored.
- Only absolute shaper values is supported in the highest level of the policy. At the child level of the policy, the absolute and the percent-based shaper values can be configured.
- Shared Policy Instance (SPI) is not supported on parameterized shaper policies.
- Scenarios with Parameterized (PQoS) policy-map and CLI policy-map applied through the RADIUS server, are not supported.
- Service Accounting - For a subscriber with service policy A (applied through dynamic template), a service policy-replacement with service policy B, is not supported (irrespective of whether shaper parameterized variables are present in both A or B) for all these scenarios:
  - If service accounting is enabled either in service policy A or in service policy B.
  - If service accounting is enabled in both A and B.
- If more than one service with service accounting is configured for each subscriber session, one of the services must show aggregate traffic of all the services. To have proper per-service accounting, it is recommended to define a parent service in such scenarios, with all the other services defined as children.

QoS Accounting

The QoS overhead accounting feature enables BNG to account for various encapsulation types when applying QoS to packets. The ATM overhead accounting enables the BNG to account for the ATM encapsulation on the subscriber line. It also accounts for the overhead added by cell segmentation. This accounting enables the service provider to prevent overruns on the subscriber line and ensures that the BNG executes QoS features on the actual bandwidth allocated to the subscriber traffic. The ATM overhead encapsulation details are listed in this table.

Table 2: ATM Overhead Encapsulation Details

DSLAM to CPE Encapsulation		**ALE Tags (RFC 4679)**		
CLI Option	**Overhead (in bytes)**	**Data Link**	**Encapsulation1**	**Encapsulation2**
snap-pppoa	12	AAL5	N/A	PPPoA LLC (1)
mux-pppoa	10	AAL5	N/A	PPPoA Null (2)
snap-1483routed	18	AAL5	Untagged Ethernet	IPoA LLC (3)
mux-1483routed	8	AAL5	Untagged Ethernet	IPoA NULL (4)
snap-rbe	28	AAL5	Untagged Ethernet	Ethernet over AAL5 LLC without FCS (6)
snap-dot1q-rbe	32	AAL5	Single-Tagged Ethernet	Ethernet over AAL5 LLC without FCS (6)
mux-rbe	24	AAL5	Untagged Ethernet	Ethernet over AAL5 Null without FCS (8)
mux-dot1q-rbe	28	AAL5	Single-Tagged Ethernet	Ethernet over AAL5 Null without FCS (8)

To enable QoS overhead accounting, see Configuring QoS Accounting, on page 26.

Configuring QoS Accounting

Perform this task to enable QoS Layer2 overhead accounting.

SUMMARY STEPS

1. **configure**
2. **dynamic-template**
3. **type** [**ppp** | **ip-subscriber** | **service**] *name*
4. **qos-account** [**AAL5** | **user-defined**] [**mux-1483routed** | **mux-dot1q-rbe** | **mux-pppoa** | **mux-rbe** | **snap-1483routed** | **snap-dot1q-rbe** | **snap-ppoa** | **snap-rbe**]

5. **exit**
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**dynamic-template** **Example:** `RP/0/RSP0/CPU0:router(config)# dynamic-template`	Enters dynamic template configuration mode.
Step 3	**type** [**ppp** | **ip-subscriber** | **service**] *name* **Example:** `RP/0/RSP0/CPU0:router(config-dynamic-template)# type ppp p1`	Specifies the type of dynamic template that needs to be applied. Three type are: • PPP • IP-subscriber • Service
Step 4	**qos-account** [**AAL5** | **user-defined**] [**mux-1483routed** | **mux-dot1q-rbe** | **mux-pppoa** | **mux-rbe** | **snap-1483routed** | **snap-dot1q-rbe** | **snap-ppoa** | **snap-rbe**] **Example:** `RP/0/RSP0/CPU0:router(config-dynamic-template-type)# qos-account AAL5 snap-rbe`	Defines the L2 QoS overhead accounting. Various keywords such as mux-1483routed, snap-rbe define different available encapsulations between the DSLAM and CPE. For details about keywords, see Table 2: ATM Overhead Encapsulation Details, on page 26.
Step 5	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-dynamic-template-type)# exit`	Exits from the current mode.
Step 6	**commit**	

Configuring QoS Accounting: An example

```
configure
dynamic-template type ppp p1
qos account AAL5  mux-1483routed
service-policy input input_1
end
```

Support for Shared Policy Instance

Shared Policy Instance (SPI) allows allocation of a single set of QoS resources among groups of BNG sub-interfaces and bundle sub-interfaces, and shares them across a group of sub-interfaces, multiple Ethernet flow points (EFPs), or bundle interfaces.

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

For bundle interfaces, hardware resources are replicated per bundle member. All sub-interfaces 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 using SPI, 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 or shaper Instance.

BNG configures a complete hierarchical policy-map that includes parent and child policies. Optionally, the SPI name can be defined and attached to the appropriate dynamic template or downloaded from RADIUS, in this manner:

- Policy configured through a CLI and applied through a dynamic-template
- Policy configured through a CLI and applied through RADIUS

Note The SPI has to be used across subscriber interfaces, but all subscriber interfaces have to be under the same access-interface (that is, the parent interface has to be same).

Restrictions

These restrictions apply to the usage of shared policy instance:

- SPI is not supported for subscribers on non-bundle interfaces.
- SPI is not supported for Parameterized QoS (PQoS). In a PQoS configuration, if there exists a SPI name, then it is ignored.
- Prior to Cisco IOS XR Release 5.2.0, SPI modified through CoA is not supported on subscribers.
- The SPI name must be changed if the policy-map associated with it is changed.
    - Once an SPI policy has been applied on a subscriber, a new policy with same policy-map name and different SPI name is rejected.

- Once an SPI policy has been applied on a subscriber, a new policy with different policy-map name and same SPI name is rejected.

Configuring a Policy with SPI in the Input or Output Direction Using Dynamic Template

Perform this task to configure a policy with shared policy instance in the input and output direction using dynamic template.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy_map_name*
3. **class** {*class_name* | **class-default** | } [**type qos**]
4. **service-policy** *service_policy_name*
5. **commit**
6. **policy-map** *policy_map_name*
7. **class** {*class_name* | **class-default** | } [**type qos**]
8. **police rate** *value*
9. **commit**
10. **dynamic-template type ipsubscriber** *dynamic_template_name*
11. **service-policy** {**input** | **output**}*policy_map_name* [**shared-policy-instance** *instance_name*]
12. **service-policy** {**input** | **output**}*policy_map_name* [**shared-policy-instance** *instance_name*]
13. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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 submode.

Step 3	**class** {*class_name* | **class-default** | } [**type qos**]
	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 submode. This example configures a traffic policy for the default class of the traffic policy policy1. The default class is named class-default.

Step 4	**service-policy** *service_policy_name* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy policy1_child`	Attaches a policy map to an input or output interface.

	Command or Action	Purpose
Step 5	**commit**	
Step 6	**policy-map** *policy_map_name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1_child`	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 submode.

Step 7	**class** {*class_name* | **class-default** | } [**type qos**]
	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 submode. This example configures a traffic policy for the default class of the traffic policy policy1. The default class is named class-default.

Step 8	**police rate** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 1024`	Configures traffic policing and enters policy map police configuration mode. The value represents the committed information rate and ranges from 1 to 4294967295.
Step 9	**commit**	
Step 10	**dynamic-template type ipsubscriber** *dynamic_template_name* **Example:** `RP/0/RSP0/CPU0:router(config)# dynamic-template type ppp PTA_TEMPLATE_1`	Creates a dynamic template of type ipsubscriber.
Step 11	**service-policy** {**input** | **output**}*policy_map_name* [**shared-policy-instance** *instance_name*] **Example:** `RP/0/RSP0/CPU0:router(config)# service-policy input policy1 shared-policy-instance spi_1`	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 entering into that interface.
Step 12	**service-policy** {**input** | **output**}*policy_map_name* [**shared-policy-instance** *instance_name*] **Example:** `RP/0/RSP0/CPU0:router(config)# service-policy output policy1 shared-policy-instance spi_2`	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 13	**commit**	

Configuring a Policy with SPI in the Input or Output Direction Using Dynamic Template: Example

```
configure
policy-map policy1
```

```
class class-default
service-policy policy1_child
!!

policy-map policy1_child
class class-default
police rate 1024 kbps
!!

dynamic-template
type ppp PTA_TEMPLATE_1
service-policy input policy1 shared-policy-instance spi_1
service-policy output policy1 shared-policy-instance spi_2
commit
```

Configuring a Policy with SPI in the Input or Output Direction Using RADIUS

Perform this task to configure a policy with shared policy instance in the input or output direction using RADIUS.

SUMMARY STEPS

1. **configure**
2. **policy-map** *policy_map_name*
3. **class** {*class_name* | **class-default**} [**type qos**]
4. **service-policy** *service_policy_name*
5. **commit**
6. **policy-map** *policy_map_name*
7. **class** {*class_name* | **class-default**} [**type qos**]
8. **police rate** *value*
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
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 submode.

Step 3	**class** {*class_name* | **class-default**} [**type qos**]
	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 submode. This example configures a traffic policy for the default class of the traffic policy policy1. The default class is named class-default.

Step 4	**service-policy** *service_policy_name* **Example:**	Attaches a policy map to an input or output interface.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-pmap-c)# service-policy policy1_child`	
Step 5	**commit**	
Step 6	**policy-map** *policy_map_name* **Example:** `RP/0/RSP0/CPU0:router(config)# policy-map policy1_child`	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 submode.

Step 7	**class** {*class_name* | **class-default**} [**type qos**]
	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 submode. This example configures a traffic policy for the default class of the traffic policy policy1. The default class is named class-default.

Step 8	**police rate** *value* **Example:** `RP/0/RSP0/CPU0:router(config-pmap-c)# police rate 1024`	Configures traffic policing and enters policy map police configuration mode. The value represents the committed information rate and ranges from 1 to 4294967295.
Step 9	**commit**	

Configuring a Policy with SPI in the Input or Output Direction Using RADIUS: Example

```
configure
policy-map policy1
class class-default
service-policy policy1_child
!!

policy-map policy1_child
class class-default
police rate 1024 kbps
commit
!!

//In the USER file in RADIUS
RoadRunner_P1@Chasing1 Cleartext-Password := "LooneyTunes_P1"
cisco-avpair += "sub-qos-policy-in=policy1 shared-policy-instance spi_1",
cisco-avpair += "sub-qos-policy-out=policy1 shared-policy-instance spi_2",
Framed-Protocol += PPP,
Service-Type += Framed-User,
Fall-Through = no
```

What to do next

Run these steps in the USER file in RADIUS:

```
RoadRunner_P1@Chasing1 Cleartext-Password := "LooneyTunes_P1"
```

```
cisco-avpair += "sub-qos-policy-in=policy1 shared-policy-instance spi_1",
cisco-avpair += "sub-qos-policy-out=policy1 shared-policy-instance spi_2",
Framed-Protocol += PPP,
Service-Type += Framed-User,
Fall-Through = no
```

Merging QoS Policy-maps

Multiple QoS policies, applied through multiple dynamic templates, can be merged and implemented on a single subscriber. The order in which the policies are merged is important, and is determined by the value of the sequence number configured in the dynamic template. A policy is deployed using a policy-map. A new optional **merge** keyword is provided with the **service-policy** command under dynamic template sub mode to allow the merging of policy-maps applied through multiple dynamic templates.

When more than two policy-maps are to be merged, two policy-maps are first merged together based on their sequence number (using the rules listed below) to create a merged policy-map. Similarly, a third policy-map is merged with the first merged policy-map. This continues till all policy-maps that are to be merged are merged together. The sequence numbers of the policy-maps are significant only while merging the policy-maps and are not related to the priority of the classes.

For example, let's say that policy-maps *p1, p2, p3, p4* each with a specific sequence number, are to be merged in that order; *p1* and *p2* are merged first based on their sequence numbers (see the rules listed below). Next, *p3* is merged with the <p1-p2> merged policy-map. Finally, *p4* is merged with the <p1-p2-p3> merged policy-map, giving the final merged policy-map.

The rules for merging two policy-maps are:

- The policy-map with the lowest sequence number takes precedence over the other.

  For example, if policy-maps *p1* with a sequence number 100 and *p2* with a sequence number 20 are to be merged, then *p2* overrides *p1*, as it has the lowest sequence number. Similarly, if policy-map *p3* with a sequence number 10 has to be merged with the policy-map *p2* having sequence number 20, then the policy-map *p3* takes precedence over the other, as it has the lowest sequence number.

- If the same class (except for the default class) is configured under both the policies, the instance of that class (including all actions configured under it) in the second policy is ignored.

- If the default class under the first policy contains any actions other than any child policy actions, then that default class is added to the end of the merged policy. If it contains any child policy actions, then the default class from the second policy is added at the end of the merged policy.

- If a child policy is configured under the default class of both policies, the two child policies are merged using the rules above. The merged child policy is then applied as the child policy under the default class of the merged parent policy.

- If a child policy is configured under the default class of either the first or second policy (but not both), then it is applied (as it is) as the child policy under the default class of the merged policy. Child policies under classes other than the default class are never merged together.

Note If the sequence numbers of two policies to be merged are configured to be the same, the order in which they are merged with respect to each other is random, and may change after the process restarts. Such configurations must be avoided.

Enabling Policy-maps Merge

Perform this task to enable merging of multiple QoS policy-maps applied through multiple dynamic templates.

SUMMARY STEPS

1. **configure**
2. **dynamic-template**
3. **type service** *dynamic-template-name*
4. **service-policy** {**input** | **output** | **type**} *service-policy_name* [**acct-stats**] [**merge** *seq_num*]
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**dynamic-template** **Example:** `RP/0/RSP0/CPU0:router(config)# dynamic-template`	Enters the dynamic-template configuration mode.
Step 3	**type service** *dynamic-template-name* **Example:** `RP/0/RSP0/CPU0:router(config-dynamic-template)# type service s1`	Creates a dynamic-template with a user-defined name for a service.

Step 4	**service-policy** {**input** | **output** | **type**} *service-policy_name* [**acct-stats**] [**merge** *seq_num*]
	Example:
	`RP/0/RSP0/CPU0:router(config-dynamic-template-type)# service-policy input QoS1 merge 10`
	`RP/0/RSP0/CPU0:router(config-dynamic-template-type)# service-policy output QoS2 merge 20`
		Associates a service-policy to the dynamic template, and enables merging of multiple QoS policies.

Step 5	**commit**	

Enabling Policy-maps Merge: Examples

```
dynamic-template type service default-service
   service-policy input default-policy-in merge 100
   service-policy output default-policy-out merge 100
!
dynamic-template type service voip-service
   service-policy input voip-policy-in merge 20
   service-policy output voip-policy-out merge 30
!
dynamic-template type service vod-service
   service-policy input vod-policy-in merge 30
   service-policy output vod-policy-out merge 50
!
dynamic-template type service turbo-button-service
   service-policy input turbo-button-policy-in merge 10
   service-policy output turbo-button-policy-out merge 10
!
end

\\the following configuration explains the merging behavior of egress qos policies
policy-map type qos default-policy-out
  class class-default
      shape average 2 mbps
      bandwidth 512 kbps
    service-policy default-policy-child-out
  !
  end-policy-map

policy-map type qos default-policy-child-out
   class critical-data
     bandwidth percent 90
     set cos 3
     queue-limit 500 ms
   !
   class best-effort-data
     shape average percent 50
     random-detect 100 ms 200 ms
      set cos 5
    !
    class class-default
      shape average percent 20
      set cos 7
    !
    end-policy-map

policy-map type qos voip-policy-out
   class class-default
     service-policy voip-policy-child-out
   !
  end-policy-map

 policy-map type qos voip-policy-child-out
   class voip-control
     priority level 1
     set cos 2
   !
   class voip-data
     priority level 2
     set cos 2
     random-detect 100 ms 200 ms
  !
   class class-default
   !
   end-policy-map

policy-map type qos vod-policy-out
  class class-default
    service-policy vod-policy-child-out
  !
  end-policy-map


policy-map type qos vod-policy-child-out
  class vod-control
     priority level 1
     set cos 1
  !
  class vod-data
   priority level 2
   queue-limit 100 ms
  !
  class class-default
  !
   end-policy-map

policy-map type qos turbo-button-policy-out
  class class-default
     shape average 10 mbps
     bandwidth 2 mpbs
   !
   end-policy-map


\\after the default and voip services are enabled on a subscriber session

policy-map type qos  <merged-policy-1>   !! Name is generated internally. This is just an
example.
  class class-default
      shape average 2 mbps
      bandwidth 512 kbps
      service-policy <merged-child-policy-1>
    !
   end-policy-map

policy-map type qos <merged-child-policy-1>
   class voip-control
     priority level 1
     set cos 2
   !
   class voip-data
     priority level 2
     set cos 2
     random-detect 100 ms 200 ms
  !
   class critical-data
     bandwidth percent 90
     set cos 3
     queue-limit 500 ms
   !
   class best-effort-data
     shape average percent 50
     random-detect 100 ms 200 ms
      set cos 5
    !
    class class-default
      shape average percent 20
      set cos 7
    !
    end-policy-map

\\after the turbo-button service is enabled

policy-map type qos <merged-policy-2>
   class class-default
     shape average 10 mbps
     bandwidth 2 mpbs
    service-policy <merged-child-policy-1> !! <merged-child-policy-1> is the same as before
 since the
                                            !! the turbo-button-policy-out does not have
```

```
any child policy
                                            !! to be merged.
   !

\\after the vod service is enabled

policy-map type qos  <merged-policy-3>
  class class-default
      shape average 10 mbps
      bandwidth 2 mbps
      service-policy <merged-child-policy-2>
    !
   end-policy-map

policy-map type qos <merged-child-policy-1>
   class voip-control
     priority level 1
     set cos 2
   !
   class voip-data
     priority level 2
     set cos 2
     random-detect 100 ms 200 ms
  !
  class vod-control
     priority level 1
     set cos 1
  !
  class vod-data
   priority level 2
   queue-limit 100 ms
  !
   class critical-data
     bandwidth percent 90
     set cos 3
     queue-limit 500 ms
   !
   class best-effort-data
     shape average percent 50
     random-detect 100 ms 200 ms
      set cos 5
    !
    class class-default
      shape average percent 20
      set cos 7
    !
    end-policy-map
```

QoS Features Supported on BNG

BNG supports these QoS features:

Policing and Queuing Support

BNG provides ingress and egress traffic policers. BNG also supports pre-existing traffic policing mechanisms per subscriber session. 1R2C and 2R3C policers with marking actions is supported at parent-level in subscriber policies. Only absolute police rates are supported at the parent-level of subscriber policies. 1R2C and 2R3C policers with marking actions are supported at the child-level in subscriber policies. Both absolute and percentage based police rates are supported at child-level of subscriber policies.

BNG supports traffic shaping at the physical port level, at the subscriber session level, at the class level, and at the VLAN level only in egress direction. The system supports all pre-existing queuing actions for subscriber sessions. The configuration of minimum-bandwidth at the parent-level in subscriber policies is blocked. If subscriber policies do not have a queuing action, the traffic on those subscribers is still subjected to S-VLAN shaping and the traffic goes out through S-VLAN policy queues if those are present; if not, the traffic goes through the interface default-queue. The shaping or bandwidth-remaining queuing action is mandatory in flat S-VLAN policies. Only absolute shape rates is supported in S-VLAN flat policies and the parent-level of subscriber policies. However, only shaping and bandwidth-remaining queuing actions are supported in the parent-level of subscriber policies and all queuing actions are supported in the child-level of subscriber policies.

These additional queuing features are supported in egress policies applied on subscribers:

- A policy can have 1 P1, 1 P2, 1 P3 and 5 normal priority queues.
- A policy can have 1 P1, 2 P2 and 5 normal priority queues. P1 and P3 queues can be shared by multiple classes whereas P2 queues are never shared.

Default Marking

BNG supports all pre-existing classification and marking options supported for L3 interfaces for use with subscriber sessions. BNG also supports L3 marking to L2 marking mapping. BNG also supports ToS to CoS mapping at LAC for downstream PPPoE frames and provides mechanisms to mark 802.1p and IP TOS fields. The system allows flexible IP TOS marking for L2TP packets based on ingress subscriber qos policy. Marking is supported at the parent-level in subscriber policies and at the child-level in subscriber policies.

QoS Policy Modification

BNG supports in-service QoS policy-modification. Modification of subscriber-policy (through Radius), S-VLAN policy (through CLI) and port sub-rate policy (through CLI) are also supported.

L2 Encapsulation

For PPPoE subscribers, the L2 encapsulation size used in QoS rate calculations must be adjustable based on the last mile encapsulation (DSLAM to subscriber home) signaled in the PPPoE tags.

Classification

The BNG supports all pre-existing classification and marking options supported for L3 interfaces for use with subscriber sessions. BNG also supports ingress classification based on 802.1P values for single and double tagged COS, classification based on DSCP in either direction, classification based on L3/L4 ACLs in either direction, and classification of L2TPv2 traffic based on the outer DSCP marking.

The classification of an incoming L2TP packet on the ingress core side interface is always based on the outer IP fields even if the packet arrives with an MPLS tag stack.

Policy Inheritance

This table is relevant for egress direction only, as in ingress direction sub-rate policy and S-VLAN policy is not supported:

Port	S-VLAN	Subscriber
Sub-rate policy	No policy is configured. Inheritance limited to traffic getting shaped by port sub-rate policy. This is done irrespective of whether a policy is configured on the S-VLAN, or not.	Subscriber policy , if present, is executed first; then, traffic is subjected to port-shaper.
Sub-rate policy	Policy is configured. Inheritance limited to traffic that gets shaped by port sub-rate policy. This is done irrespective of whether a policy is configured on the S-VLAN or not.	Subscriber policy is executed first, if present, and then, S-VLAN policy is executed. Finally traffic is subjected to port-shaper.
HQoS or policy with more than class-default	Policy configuration is blocked and port policy is inherited.	Policy configuration is blocked and port policy inherited through the S-VLAN.
No policy configured	Policy is configured.	Subscriber policy is executed first, if present, and then S-VLAN policy is executed.

Subscriber with No QoS

When QoS is not configured on a subscriber, the parent S-VLAN, or on the port, subscriber traffic goes out using the default-queue of its parent's physical port.

- The subscriber is subjected to the S-VLAN policy and goes out using S-VLAN policy queues, if those are present. If the S-VLAN policy does not have its own queues, then all the S-VLAN traffic, including the subscriber's, goes out through the default queue of the physical interface.
- The subscriber is subject to a port policy, but no S-VLAN policy. Similar to the S-VLAN case, the subscriber traffic is subject to it and uses its queues.
- If a non-port-shaper policy is applied on the port, the application of policy on S-VLAN and subscriber is blocked. In such a scenario, subscriber traffic is subjected to the policy applied on the port.

Control Packet Handling

BNG provides priority treatment in handling PPP Link Control Protocol (LCP) packets. The control packets are handled in high priority without the need of user configuration, and these packets are not subjected to QoS policies that are applied on both ingress and egress of the interface. In the case of LAC upstream direction, if user wants a trusted COS value, then a PPP command is provided to impose the core-side header based on the set trusted-COS. Thus, this ensures the priority treatment of these control packets in the network.

S-VLAN Shaping and Statistics

In the egress direction, the BNG supports the ability to have policies at three different levels: the subscriber interface level, the stacked virtual local area network (S-VLAN), and at the port level. The egress S-VLAN and port-level policies are applied through CLI directly at the interface level. For applying a QoS policy on S-VLAN, see Configuring Policy on S-VLAN, on page 42

The subscriber policy can only be applied through a dynamic template or via RADIUS. The egress subscriber policy can be a two-level policy. The S-VLAN and port-level policies can only be flat policies, with only the class default, with the only action being a shaped rate. Essentially it provides a means to constrain the S-VLAN or port to a maximum rate via shaping.

In the ingress direction, the traffic is only subject to the subscriber input policy where the subscriber policies are applied through RADIUS or dynamic-template.

The traffic through the S-VLAN includes traffic to many subscribers that may have already been shaped by the subscriber policies. Providing statistics on that S-VLAN shaper is important in order to monitor whether it is reaching the maximum capacity. Unlike the subscriber QoS policies, the HW does not have the ability to directly track the usage or transmitted packets/bytes through this S-VLAN shaper. So unlike other statistics, the BNG provides the S-VLAN QoS policy-related statistics by aggregating the statistics of the underlying subscriber policies. The statistics are displayed via show commands (and MIBs as appropriate) consistent with all other interface types.

S-VLAN supports these conditions:

- Modification of QoS rates.
- Modification of S-VLAN policy to change number of levels in the policy is rejected.
- Modification of two-level S-VLAN policy to add or remove child-level classes is rejected.
- Modification of classification criteria in child-level classes, in two-level policy, is rejected.
- Addition or removal of actions, in both two-level and flat policy, is rejected.

QoS Attachment Points

This table lists the QoS attachment points, and modes for definition and application.

QoS Attachment Point	Definition	Application	Type of Policy
Port (sub-rate policy)	CLI/XML	CLI/XML	Flat – class-default only
S-VLAN	CLI/XML	CLI/XML	Flat – class-default only. 2 level, with parent class-default only and child any classification.
Subscriber	CLI/XML	Dynamic-Template	2 level, with parent class-default only and child any classification.
Subscriber	CLI/XML	RADIUS	2 level, with parent class-default only and child any classification.
Subscriber	RADIUS (parameterized QoS)	RADIUS	2 level, with parent class-default only and child any classification.

Un-supported configurations will not be blocked. In S-VLAN policies and subscriber policies, any configuration other than the ones listed in these tables will be blocked:

Table 3: Supported Configuration in Ingress Direction

	Classification	Action	Rates
Subscriber Parent Level Policy	Class-default only	police, marking	Absolute only
Subscriber Child Level Policy	Any, with baseline restrictions	police, marking	Absolute and percent

Table 4: Supported Configurations in Egress Direction

	Classification	Action	Rates
S-VLAN Flat Policy	Class-default only	Any, with mandatory shape action	Absolute only
S-VLAN Parent Level Policy	Class-default only	Any, with mandatory shape action	Absolute only
S-VLAN Child Level Policy	Any, with baseline restrictions	Any	Absolute and percent
Subscriber Parent Level Policy	Class-default only	shape, bandwidth remaining, police, marking	Absolute only
Subscriber Child Level Policy	Any, with baseline restrictions	Any	Absolute and percent

VLAN Policy on Access Interface

BNG supports ingress and egress VLAN policies on an access-interface. Unlike as in the case of S-VLAN (subscriber-parent) policy, the access-interface VLAN policy is not inherited by the session policy. The VLAN policy does not provide reference bandwidth to session policies. The VLAN policy statistics does not include session policy statistics. Only the access-interface traffic is subjected to the VLAN policy.

For details, see Configuring VLAN Policy on an Access Interface, on page 43.

This table summarizes the support for VLAN and S-VLAN policies in ingress and egress directions:

Policy Direction	V-LAN policy (without subscriber-parent keyword)	S-VLAN policy(with subscriber-parent keyword)
Ingress	Supported	Not supported
Egress	Supported	Supported

Restrictions

These restrictions apply to the VLAN policy on the access-interface, when used without the **subscriber-parent** keyword:

- The VLAN policy needs to be attached to the access-interfaces, before bringing up the sessions with QoS policies.
- The restrictions specified for the in-place modification of S-VLAN policy, are applicable to VLAN policy as well. For instance, the in-place modification for the VLAN policy supports only rate-changes. This restriction also applies in adding a policer or shaper and in changing the policy-map to include more classes.

Configuring Policy on S-VLAN

Perform this task to apply a QoS policy on a S-VLAN.

Note

- S-VLAN policy has to be provisioned before any policies are installed on subscribers.
- Application of S-VLAN policy is rejected, if policies are already installed on subscribers.
- Removal of S-VLAN policy is rejected, if subscriber policies are present under that S-VLAN.

SUMMARY STEPS

1. **configure**
2. **interface** *type*
3. **service-policy output** *name* **subscriber-parent**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type* **Example:** `RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1.1`	Configures the subscribers on the Bundle-Ether access interface.
Step 3	**service-policy output** *name* **subscriber-parent** **Example:** `RP/0/RSP0/CPU0:router(config-if)# service-policy output svlan subscriber-parent`	Configures the s-vlan policy with the subscriber-parent keyword.
Step 4	**commit**	

Configuring Policy on S-VLAN: An example

```
configure
interface Bundle-Ether1.1
service-policy output svlan_pmap subscriber-parent
end
!
```

Configuring VLAN Policy on an Access Interface

Perform this task to apply an ingress and egress QoS VLAN policy on an access interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type*
3. **service-policy input** *service-policy-name*
4. **service-policy output** *service-policy-name*
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type* **Example:** `RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether18.203`	Configures subscribers on the Bundle-Ether access interface.
Step 3	**service-policy input** *service-policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-subif)# service-policy input mark`	Configures the ingress VLAN QoS policy on the access-interface.
Step 4	**service-policy output** *service-policy-name* **Example:** `RP/0/RSP0/CPU0:router(config-subif)# service-policy output metering`	Configures the egress VLAN QoS policy on the access-interface.
Step 5	**commit**	

Configuring Ingress and Egress VLAN Policies on an Access Interface: Example

```
//Attaching Ingress and Egress VLAN Policies on an Access Interface

configure
interface Bundle-Ether1.1
service-policy input INGRESS_MARKING_POLICING_POLICY
service-policy output VLAN_POLICY
end
!


//Attaching Ingress VLAN Policy and Egress S-VLAN Policies on an Access Interface

configure
interface Bundle-Ether1.2
service-policy input INGRESS_MARKING_POLICING_POLICY
service-policy output S_VLAN_POLICY subscriber-parent
end
!
```

Additional References

These sections provide references related to implementing QoS.

MIBs

MIB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html