RBFS HQoS Configuration Guide

Version 20.7.1, 15 July 2020

Registered Address	Support	Sales
26, Kingston Terrace, Princeton, New Jersey 08540, United States		
		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

Table of Contents

1. Introduction to Hierarchical Quality of Service (HQoS)

Hierarchical Quality of Service (HQoS) is a technology that allows you to specify QoS (Quality of Service) behavior at multiple policy levels. It provides a high degree of granularity in traffic management. It can ensure that each network service gets the network resources it needs. This is achieved by classifying, policing, shaping, and scheduling the traffic based on service types. For example, in a simple QoS, you can differentiate between services (such as voice and video), but using H-QoS, you can apply QoS policies to different users, VLANs, maybe logical interfaces and so on.

The RtBrick Full Stack (RBFS) uses the following HQoS mechanisms:

- **Classifier:** Classifies each incoming packet as belonging to a specific class, based on the packet contents. Packets are classified according to the information such as ToS/TC in IPv4/v6 header, EXP field in MPLS header, and 802.1p priority of the L2 VLAN tag.
- **Queuing:** Drop unqualified packets in advance using the Weighted Random Early Detection (WRED) technology in the case of congestion to ensure bandwidth for qualified services. This is performed at the egress.
- **Scheduler:** Manage traffic on a device using different algorithms for queue scheduling. Such algorithms include Fair Queuing (FQ), Weighted Round Robin (WRR), and Strict Priority (SP).
- **Policer:** Policer is implemented in the ingress to drop the unwanted traffic. Policer supports Committed Information Rate (CIR), the Committed Burst Size (CBS), Peak Information Rate (PIR), and Peak Burst Size (PBS). Drop behavior is to either mark traffic as green, yellow, or drop.
- **Shaper:** Shaper is implemented in egress to rate-limit the traffic.
- **Remarking:** Remarking allows you to rewrite the outgoing packet's codepoint. Remarking can be performed in the ingress or the egress side of the hardware pipeline.

1.1. Uniform MPLS Model

RBFS supports only Uniform-MPLS model. This model provides the following functionality:

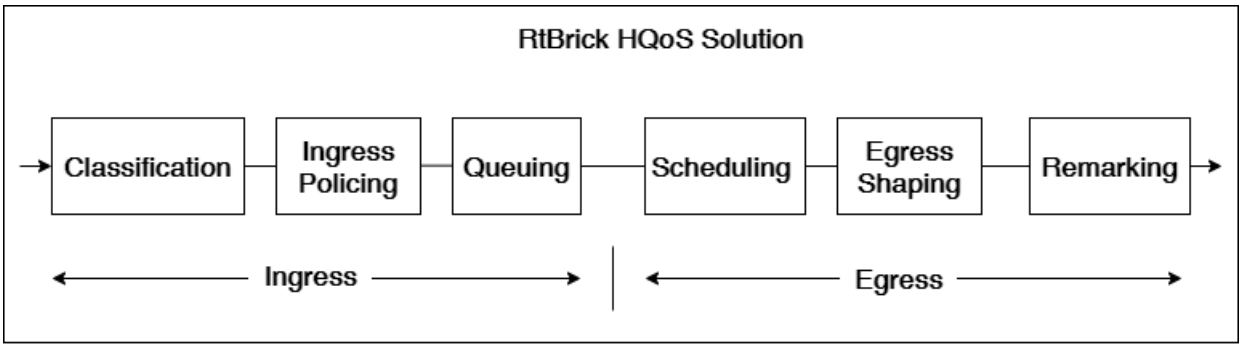
- **During MPLS encapsulation**: MSB 3-bits from 8-bits IPv4-ToS or IPv6-TC are copied to the EXP bits of the newly added MPLS header(s).
- **During MPLS decapsulation**: IPv4-ToS or IPv6-TC bits are derived from the popped MPLS label. ToS/TC bits are derived by copying the 3-bits of the EXP field to the MSB 3-bits of the ToS/TC field. The remaining bits in the ToS/TC are set to all 0's.

MPLS to IPv4-ToS or IPv6-TC and vise-versa mapping for uniform MPLS mode is shown in the table below:

IPv4-TOS / IPv6-TC	EXP	DSCP
0-31	0	0-7
32-63	1	8-15
64-95	2	16-23
96-127	3	24-31
128-159	4	32-39
160-191	5	40-47
192-223	6	48-55
224-255	7	56-63

1.1.1. Components of the RBFS QoS Solution

The figure below shows the primary configuration components of RBFS QoS solution.



By using HQoS, you can:

- Allow a shaper to shape the traffic at the egress
- Allow a Policer to police the traffic at ingress
- Apply QoS Profile on Subscriber Interfaces and /or L3 interfaces

1.2. Supported Hardware

- Edgecore AS5916-XKS

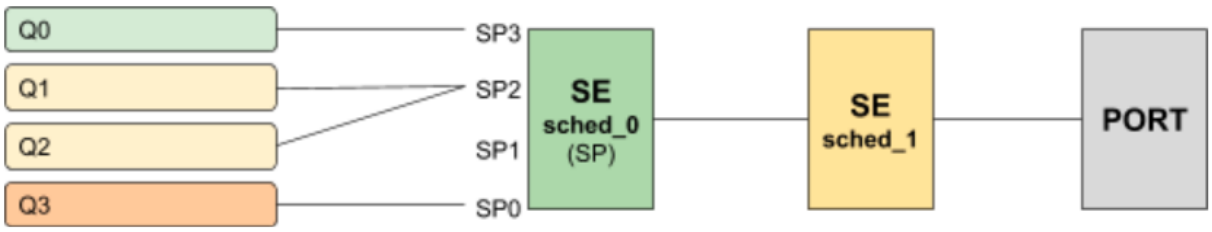
2. HQoS Features

This chapter explains the following topics:

2.1. Priority Propagation

Hierarchical QOS (HQoS) on RBFS is implemented by connecting or chaining queues to scheduler elements (Q —> SE), scheduler elements to each other (SE —> SE) and scheduler elements to ports (SE —> PORT). Each scheduler element can have different child connection points based on types described in section Scheduler.

This means that sched_0 in the example below is not scheduling between the attached queues, but between the different child connection points SP0 to SP3. The scheduler element sched_0 cannot differentiate between Q1 and Q2 in this example because both are connected to SP2.



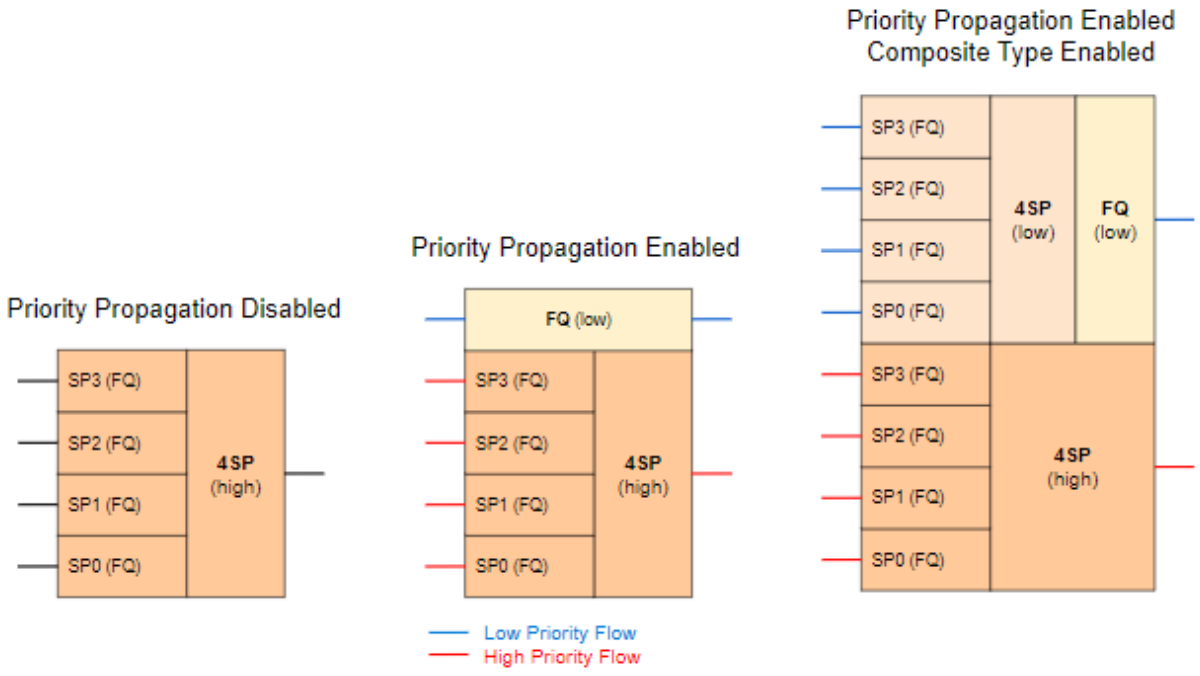
Without priority propagation each scheduler element can have multiple child connection points but just one parent connection point. Therefore traffic leaving a scheduler element can't be differentiated by the parent scheduling element. The

parent scheduler element sched_1 receives the traffic from sched_0 on the selected child connection point. As already mentioned scheduling within a scheduler element happens between child connection points. Second, a scheduler element has only one parent connection point which can be connected to a child connection point of another scheduler element (output of sched_0 → input of sched_1). This results into the situation that all traffic from this SE is handled equally regardless of the queue. This may lead into the dropped priority traffic like voice or control traffic in case of congestion in parent elements. For example, if sched_1 has a shaping rate lower than the one of sched_0, it will drop traffic unaware of its original priority.

This problem is addressed with priority propagation which is enabled per default.

With priority propagation the scheduler elements operate in a dual-flow mode with high and low priority flows. The credits generated from the physical interface will be consumed by all attached high priority flows first and only remaining credits will be available for low priority flows. In this mode an implicit FQ element is created for each scheduler element. All queues assigned to low priority flow will be attached to his element.

An additional composite option of the scheduler element allows also the differentiation between multiple low priority queues if required. This composite type is created implicitly and does not need to be configured.

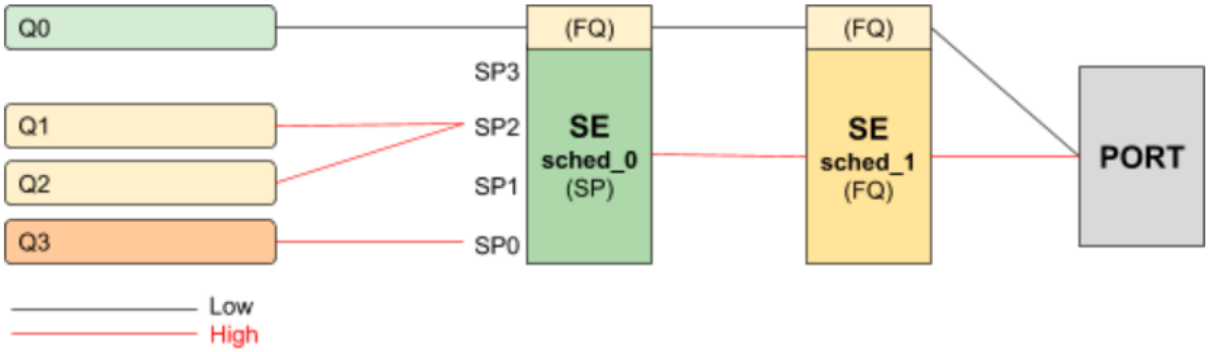


Without priority propagation enabled, each scheduler element consumes only one scheduler resource compared to two elements if enabled. The composite type consumes three scheduler elements.

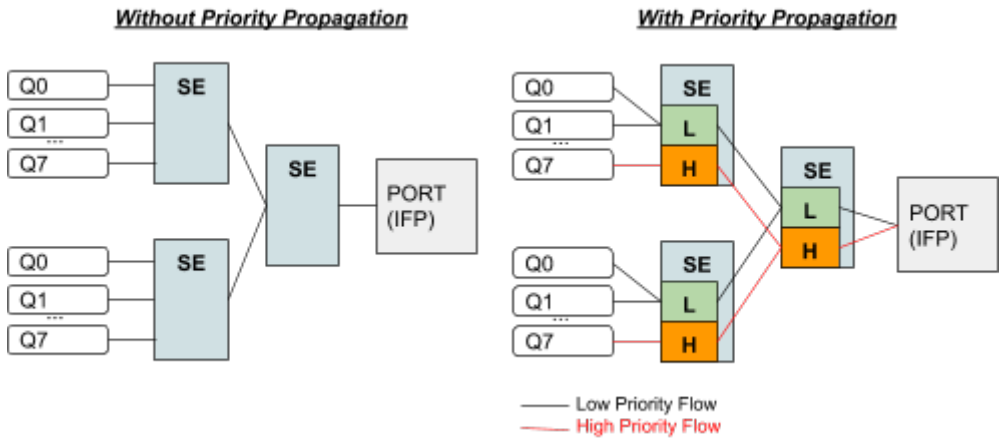
With priority propagation disabled, all traffic is considered as high priority flow.

Now for each queue we can select if connected to high priority or low priority flow where high priority flow is selected per default if not explicitly mentioned.

Assuming the example as before but with priority propagation and Q0 assigned to low priority flow and Q1 - Q3 assigned to high priority flow.



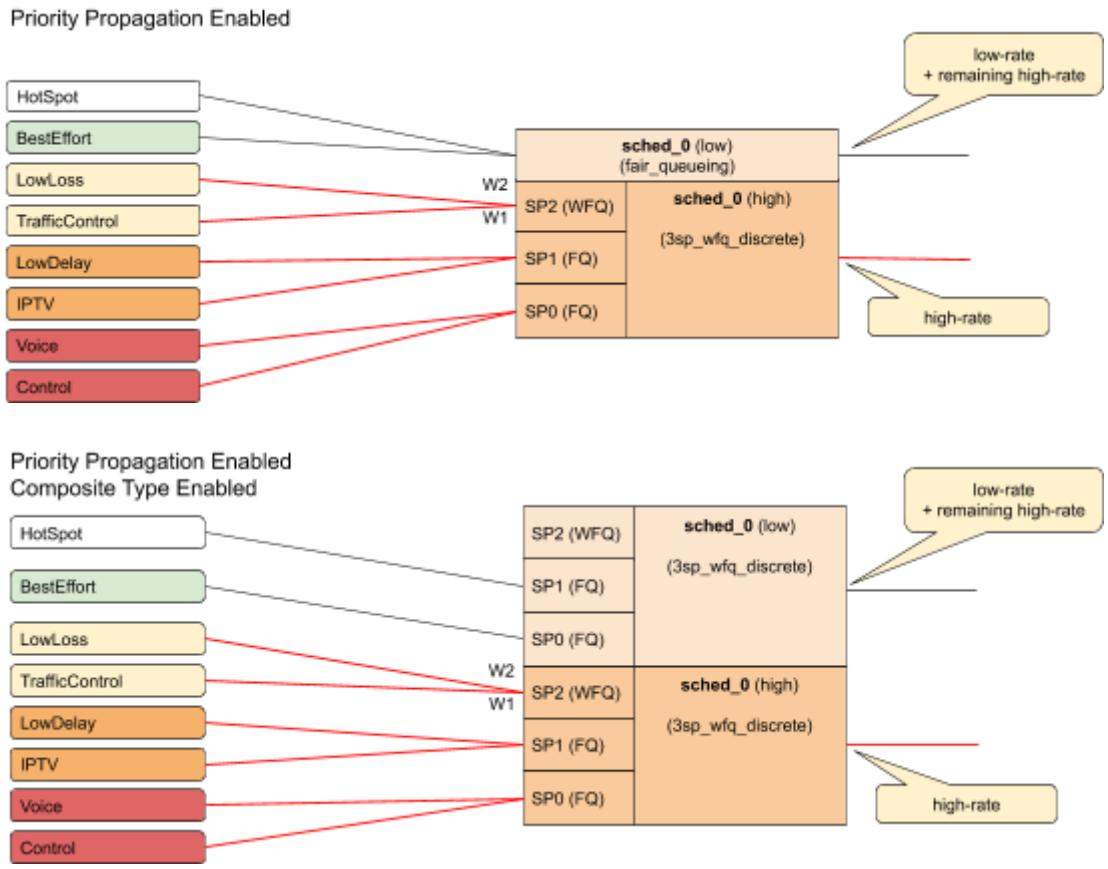
The figure below shows a typical multi level QoS configuration without priority propagation on the left and with priority propagation on the right side.



The credits generated from the physical interface will be consumed by high priority flow first and remaining credits will be available for low priority flow. The high flow traffic at any one element is scheduled based on type and connection point. Between schedulers it depends on how they are connected to the parent scheduling element. Per default all levels there is FQ for low and FQ for high priority flows. The port scheduler is also FQ.

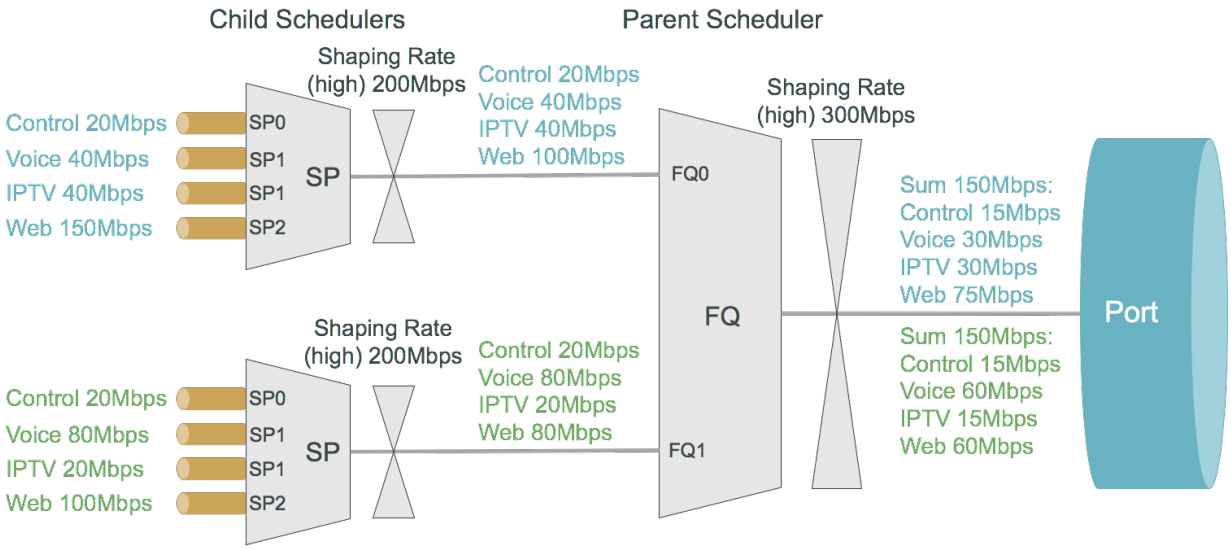
In this mode each shaper supports two different rates for low and high priority where the actual shaper rate is the sum of low and high priority rate. If low priority rate is zero, this flow is only served if high priority flow is not consuming all credits. An example might be a high rate of 9m and low rate of 1m which results in max 10m for low priority flow if high priority flow is not consuming any packets but at least 1m is ensured.

The following example shows a typical access service provider configuration with priority propagation enabled with and without composite type.

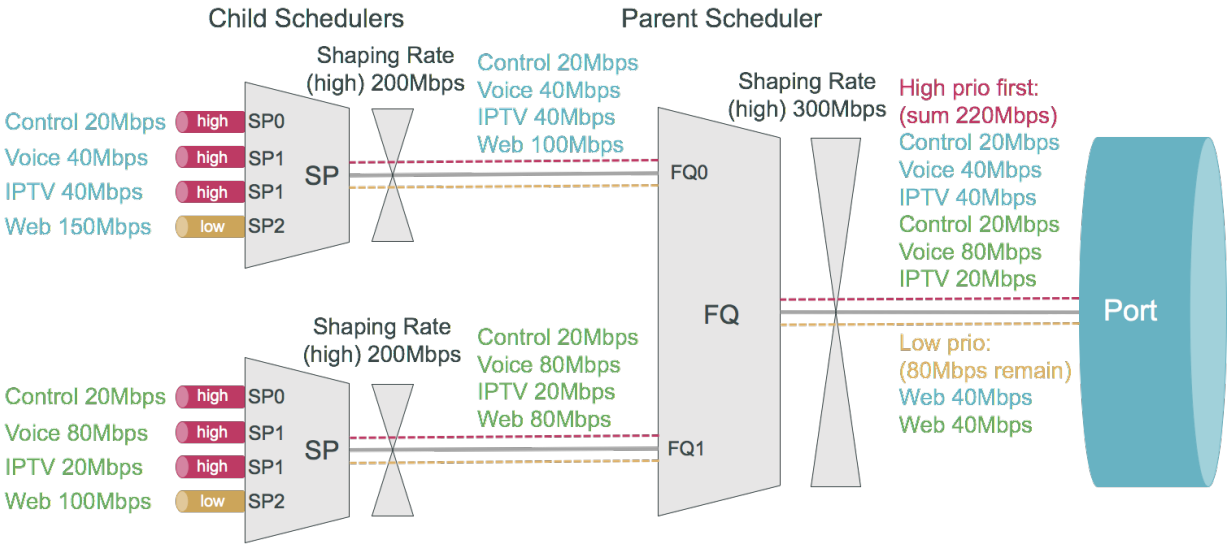


2.1.1. Simple Priority Propagation Scheduling Example

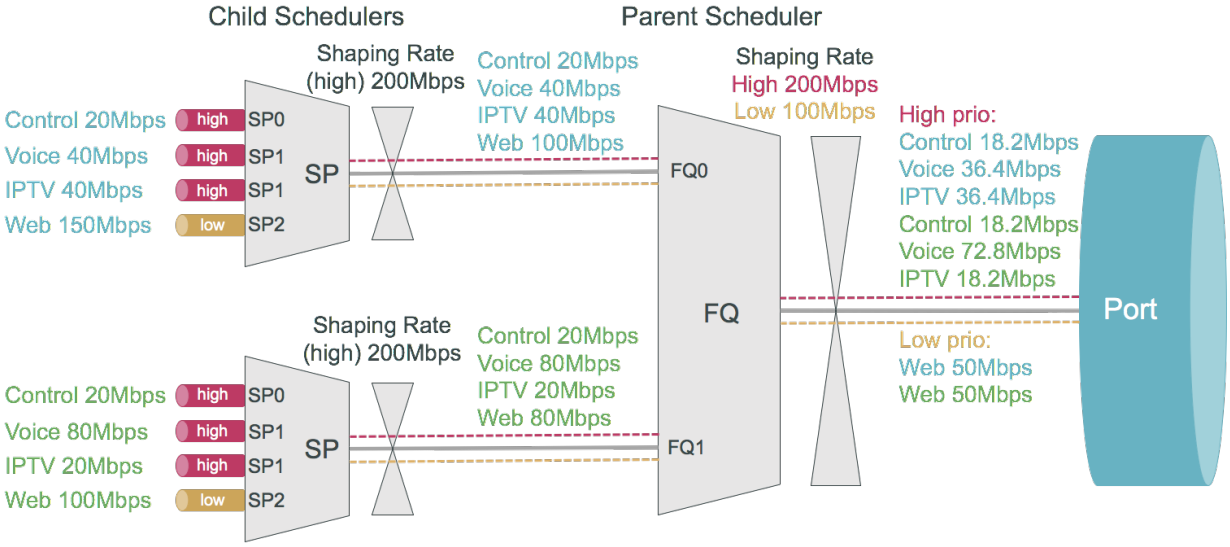
Without priority propagation, the parent scheduler drops traffic equally from all classes as it is unaware of priorities:



With priority propagation, the parent scheduler serves high priority flows first as shown in the figure below:



With priority propagation and dual-flow shaping, the parent scheduler serves high priority flows first up to the high flow shaping rate:



2.2. Classifier

Classifiers identify the class to which a packet belongs. It is performed on the ingress and maps incoming packet codepoint to a predefined class. In RBFS the classification of packets is based on Behavior Aggregate (BA) classifier. BA Classification relies upon markings (that is, codepoint) placed in the headers of incoming packets:

- Ethernet: 802.1p bits - 3 bits.
- IPv4: Type of Service byte (ToS) - 8 bits.
- IPv6: Traffic Class (TC) - 8 bits.
- MPLS: Experimental bits (EXP) - 3 bits.

- 802.1p and IPv4/IPv6 classifiers are applied on either Subscriber IFL or L3 IFL by attaching the classifier to a profile.
- MPLS Exp classifiers are applied either globally or per-instance (to support multiple VPN marking schemes) by attaching the classifier globally or to an instance.

Classifier configuration has the following guidelines and limitations:

- For IPv4: Only ToS based classification is possible. DSCP based classification is not possible.
- For IPv6: Only TC based classification is possible. DSCP based classification is not possible.
- For EXP classification, RBFS uses the Uniform MPLS Model to copy MSB 3-btis from DSCP to EXP field at the time of MPLS encapsulation at the remote box.
- IPv4/IPv6 Classifiers do not match on labelled traffic. MPLS Classifier is required for the same.

- Default class for Queueing is *class-0*. If classification is not configured for an incoming packet's codepoint, the packet will be classified as *class-0*.
- RBFS supports 8 **classes**: *class-0* to *class-7*.

2.3. Remarking

The packet markers set the codepoint in a packet to a particular value, adding the marked packet to a particular behavior aggregate. When the marker changes the codepoint in a packet, it "remarks" the packet. The codepoint in a packet can be IPv4-ToS, IPv6-TC, or MPLS-EXP field.

In RBFS remarking can be performed at the ingress or egress:

- Ingress remarking is achieved by configuring the **remark-codepoint** field in the Classifier. Ingress remarking rewrites the IPv4-ToS or IPv6-TC at the ingress side with configured remark-codepoint. The configured remark-codepoint can be modified again at the egress side using remark-map.
- Egress remarking is achieved by configuring the remark-map. Remark Map is the mapping of **match-codepoint** and **color** to **remark-codepoint**. Egress remarking helps to either remark the IPv4-ToS / IPv6-TC field in the IP header or to write the EXP field in the MPLS label(s). If the remark-codepoint is not configured in the Classifier, match-codepoint in the remark-map is the ToS/TC value of the incoming IP packet. If the remark-codepoint is configured in the Classifier, match-codepoint is the same value as the remark-codepoint in Classifier. Here color is optional, and can be used to set different remark-codepoint for same match-codepoint based on the color marked by the Policer

(that is, GREED or YELLOW).

- In the tunnel termination cases, the remark-codepoint in classifier is not useful. Since RBFS supports Uniform MPLS mode, the IPv4-ToS or IPv6-TC is derived from the EXP of the terminated MPLS header. Then remarking of the outgoing IP packet can be performed using remark-map, where match-codepoint shall be the ToS/TC derived from EXP.
- For IPv4: Only ToS based remarking is possible. DSCP based remarking is not possible.
- For IPv6: Only TC based remarking is possible. DSCP based remarking is not possible.

Remarking is supported for:

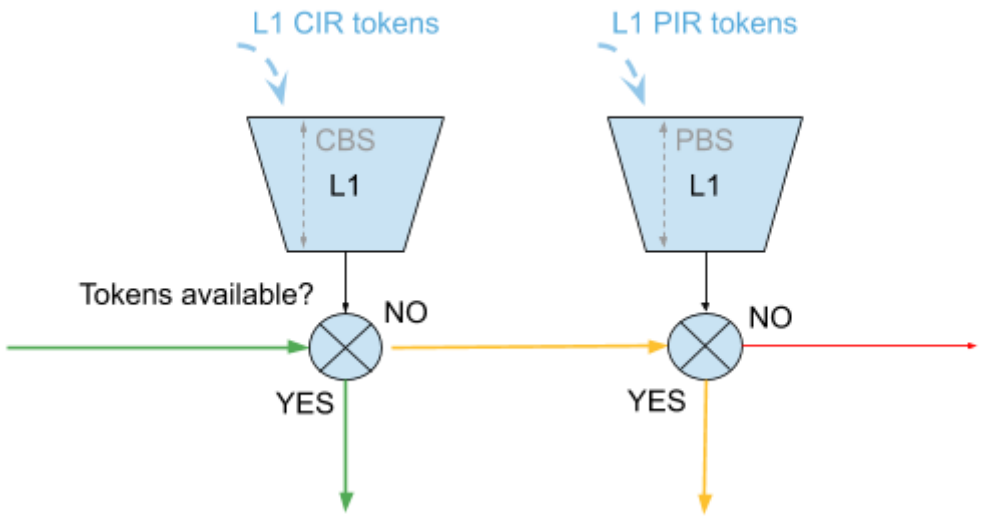
- IPv4: Type of Service byte (ToS) - 8 bits.
- IPv6: Traffic Class (TC) - 8 bits.
- MPLS Over IPv4: MPLS Experimental bits (EXP) - 3 bits.
- MPLS Over IPv6: MPLS Experimental bits (EXP) - 3 bits.

The IPv4 or IPv6 remark-map is applied on an interface, that is, subscriber-ifl or l3ifl using Profile Name. MPLS-Over-IPv4 or MPLS-Over-IPv6 remark-map is applied either globally or per-instance (to support multiple VPN marking schemes) using Remark-Map Name.

2.4. Policer

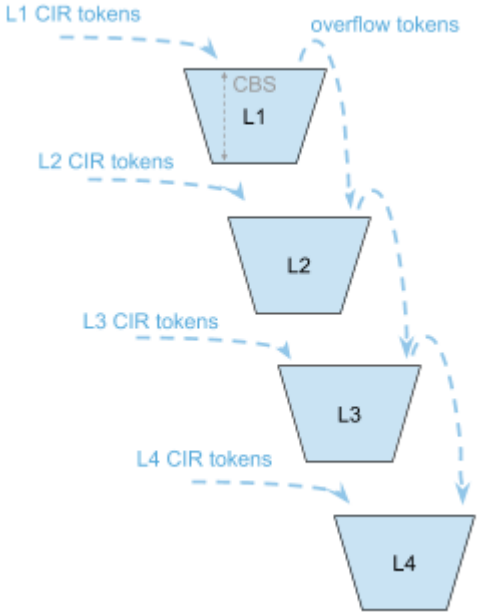
Policer defines the rate at which certain applications can access the hardware resource. So as to rate-limit the traffic from an application, policer hard-drops the unwanted packets in the ingress side.

In RBFS, policers support “**two-rate, three-color**” type in a 4-levels cascaded mode. This means that each policer level has two rates (CIR and PIR) and three colors (green, yellow and red) with two token buckets as shown below.



This means that traffic below CIR is marked green. Traffic above CIR but below PIR is yellow and above PIR is red. Traffic marked red will be dropped. Traffic marked yellow can be demoted by changing ToS, TC, or EXP using remark-map.

In 4 level cascade mode, unused tokens can be passed from higher priority levels to lower priorities where level 1 has highest and level 4 has the lowest priority as shown in the figure below.



Therefore a lower level configured with CIR 0 can still serve traffic if higher priority levels are not consuming all available tokens.

The available tokens per level are calculated by remaining CIR credits from upper levels and additional credits based on configured CIR per level. Per default the resulting tokens are not limited. The optional max CIR rate attribute allows to limit the sum of tokens from CIR and upper levels. Let's assume level 1 and 2 are both configured with a CIR of 2m. Without max CIR (default behaviour) level 2 can reach

up to 4m (level 1 CIR plus level 2 CIR). This can be limited by max CIR (for example, 3m). Obviously max CIR is not relevant for level 1.

Example

	CIR	RX	TX	CIR	RX	TX
L1	4m	1m	1m	4m	1m	1m
L2	6m	20m	9m	6m / max CIR 8	20m	8m
L3	0m	20m	0m	0m	20m	1m
L4	0m	20m	0m	0m	20m	0m
SUM	10m	61m	10m	10m	61m	10m

In the columns 2 through 4 of the preceding example table, L1 consumes only 1m of the available 4m and passes the remaining 3m to L2 which adds additional 6m based on their own configured CIR resulting in 9m.

In the columns 5 through 7 of the preceding example table, L1 consumes only 1m of the available 4m and passes the remaining 3m to L2 which adds additional 6m based on their own configured CIR resulting in 9m. But because of the CIR limit set to 8m, only 8m of 9m can be used at this level. The remaining 1m is now passed to L3 which does not add additional CIR based credits. In both examples L4 would be able to reach up to 10m if upper levels are not consuming credits.

2.5. Class-Policer-Map

Since RBFS supports up to 8 classes but only 4 policer levels, there is a need to map multiple classes to the same policer level. A *class-policer-map* defines such mappings. Using class-policer-map configuration, one can map any class to any supported policer level (that includes mapping multiple or all classes to the same level). Similar to policer, a class-policer-map is attached to a profile.

If class to level mapping is not configured, no policing will be applied to the traffic for that class.

2.6. Queueing

Queuing helps to drop the unwanted traffic in advance at the ingress side in case of congestion. This is to ensure bandwidth for qualified services.

RBFS supports the following queueing techniques:

- Tail Drop (TD): This is a conventional congestion avoidance technique. When the network is congested, drop subsequent packets from the queue.

- Weighted Random Early Detection (WRED): This technique requires configuring “Minimum Threshold”, “Maximum Threshold” and “Drop Probability”, which define the start and end range where packets may get discarded. When the average queue size is below the min threshold, no packets will be discarded. The drop_probability parameter can be used to specify the drop probability at the max threshold. When the average queue size is between the min and max threshold, the drop probability increases linearly from zero percent (at the min threshold) to drop_probability percent (at the max threshold). When the average queue size is greater than the max threshold, all packets are discarded.
  - When the average queue size is less than the “Minimum Threshold”, no packets will be discarded.
  - When the average queue size is greater than the “Maximum Threshold”, all packets are discarded.
  - When the average queue size is between “Minimum Threshold” and “Maximum Threshold”, the drop probability increases linearly from zero percent (at the min threshold) to drop probability (at the max threshold).

- Default queue within a queue group is the one mapped to *class-0*. If classification is not configured for an incoming packet’s codepoint, the packet will be classified as *class-0*. Thus will be mapped to queue mapped to *class-0* in *Queue Classifier*. For more information, see [queue_classifier].
- Maximum supported Queue size depends upon DRAM/OCB memory. Since OCB is external memory, hardware does not limit the size that can be configured per Queues.

2.7. Class-Queue-Map

A class-queue-map defines the mapping of classes and queues. Class Queue Map is attached to a profile.

2.8. Queue-Group

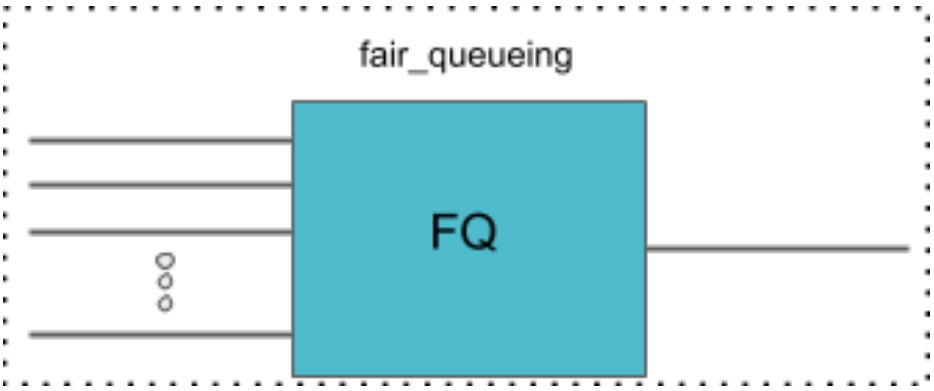
A Queue Group defines the Queue bundle. A Queue Group contains bundle of either 4 or 8 queues.

2.9. Scheduler

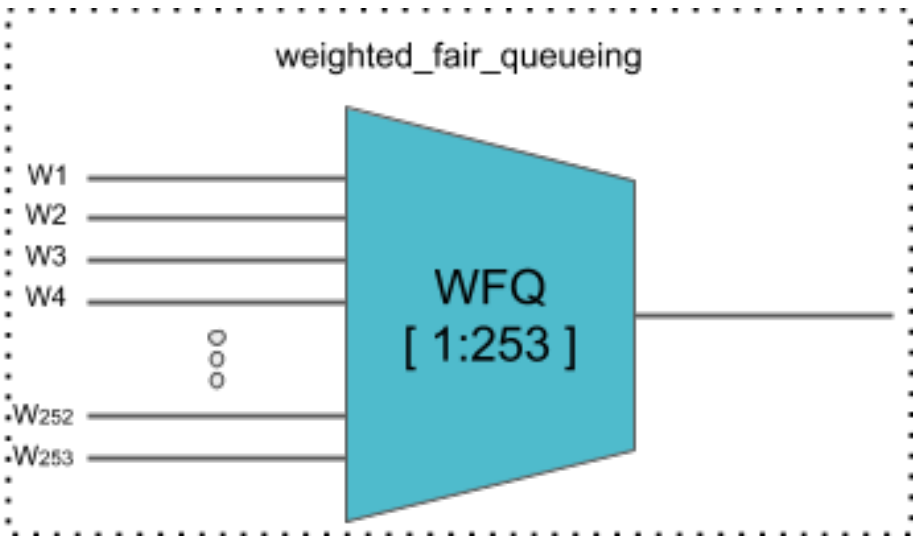
A scheduler configuration defines scheduler parameters such as type and shaping rate. The shaping rate defined for a scheduler applies to queue(s) associated with it.

The following scheduler types are supported:

- **Fair Queueing (FQ)**: Uses round-robin approach to select the next packet to service. This method ensures that all the flows are serviced equally. Configure scheduler type as *fair_queueing* to create FQ scheduler.



- **Weighted Fair Queueing (WFQ)**: Uses round-robin approach but with no guarantee of flow being serviced equally (like in FQ). The rotation of the next packet to service is based on the weight that is assigned to each flow. Configure scheduler type as *weighted_fair_queueing* to create WFQ scheduler.
  - Supported weight: 1 to 253



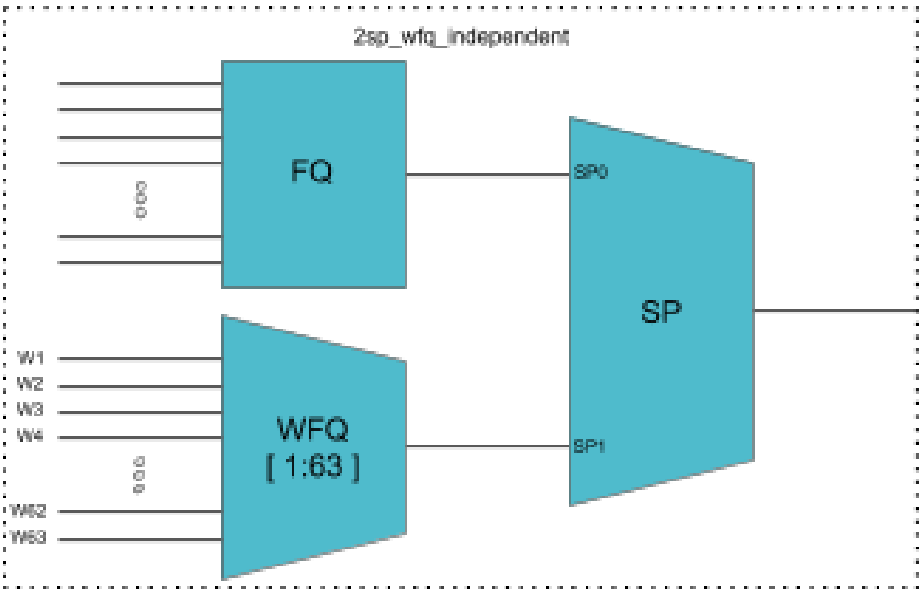
In any WFQ scheduler the lower the weight, the higher the bandwidth portion is awarded.

- **Strict Priority (SP)**: Uses priority based approach to service the flow. SP schedulers are supported in “hybrid” mode only. Hybrid mode combines FQ-WFQ schedulers using strict priority.

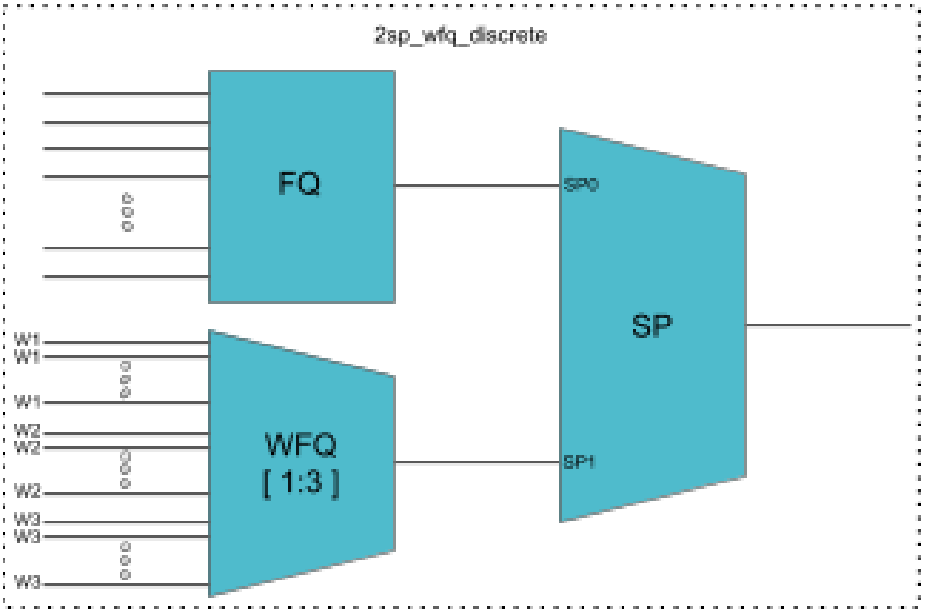
The priority order for SP is: **strict_priority_0 > strict_priority_1 > strict_priority_2 > strict_priority_3** (where **strict_priority_0** being highest priority and **strict_priority_3** being lowest)

The following SP scheduler types are supported:

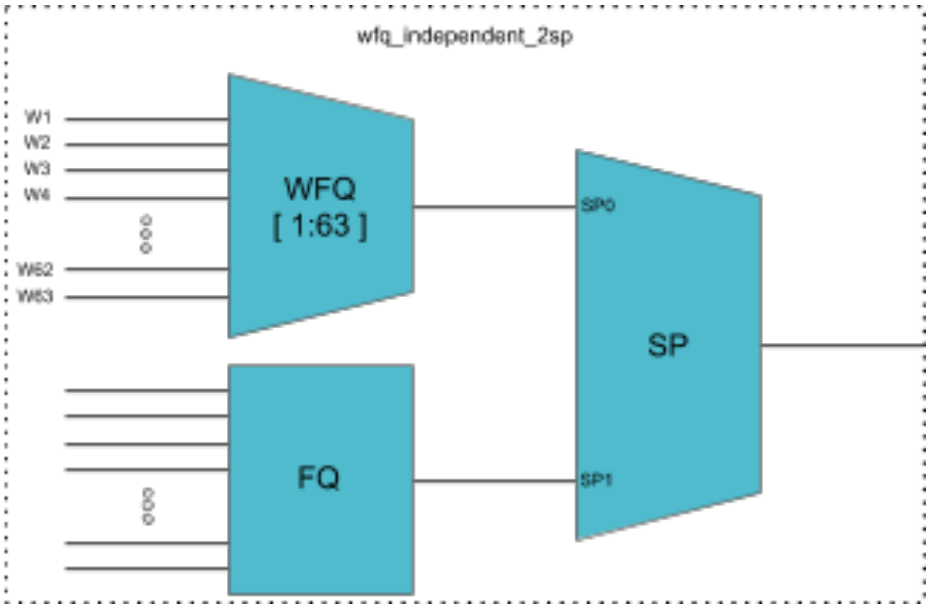
- **2 Strict Priority (2SP)**: Uses SP between 1-FQ and 1-WFQ. There are following types of 2SP hybrid schedulers:
  - type **“2sp_wfq_independent”**
    - Supported weight: 1 to 63



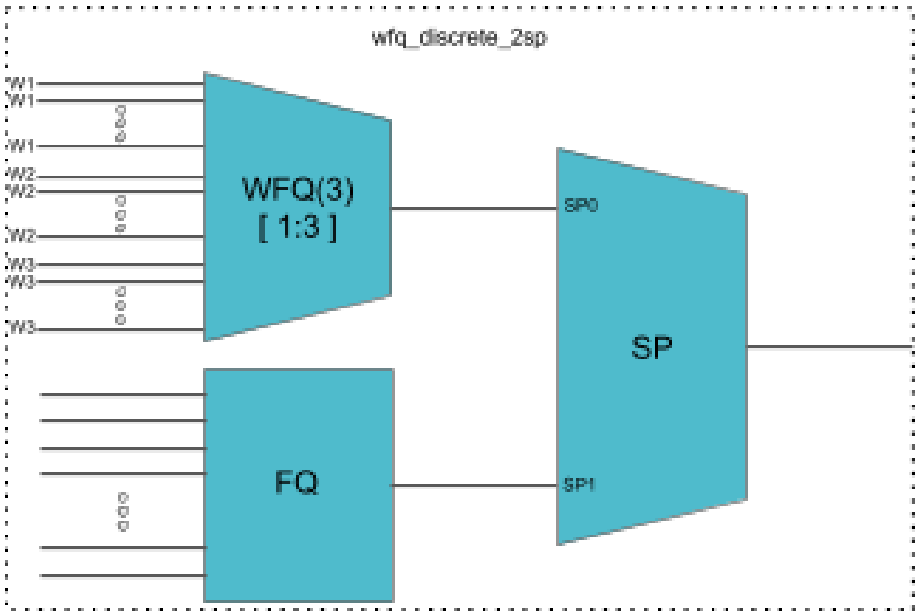
- type **“2sp_wfq_discrete”**
  - Supported weight: { 1, 2, 3 }



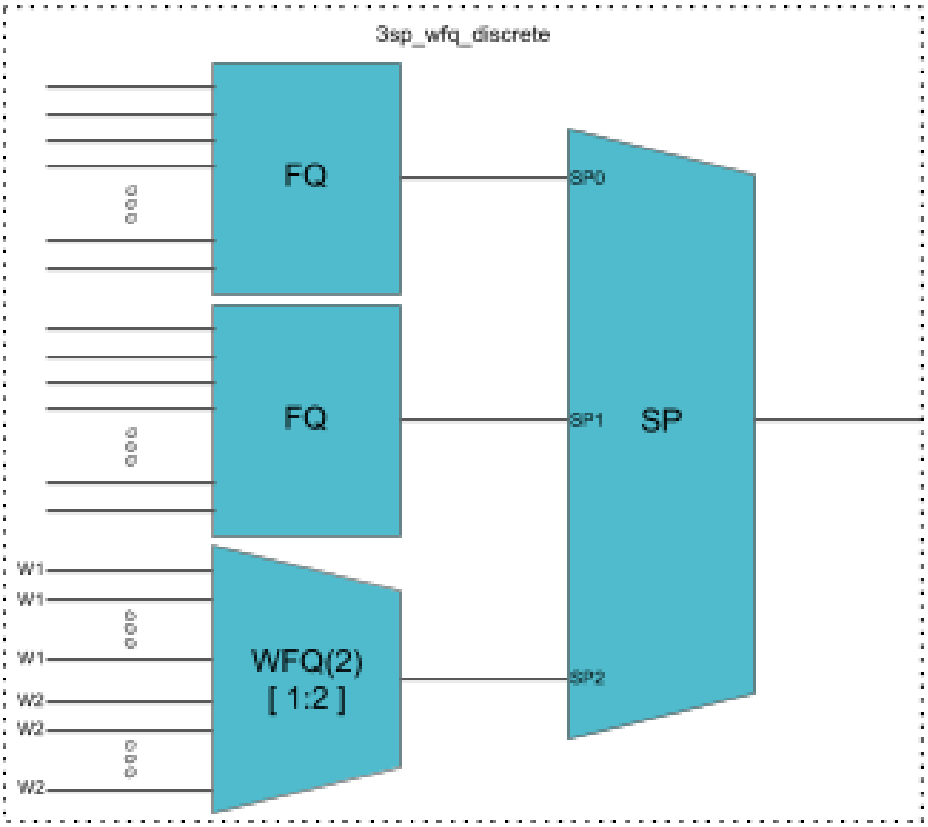
- type **“wfq_independent_2sp”**
  - Supported weight: 1 to 63



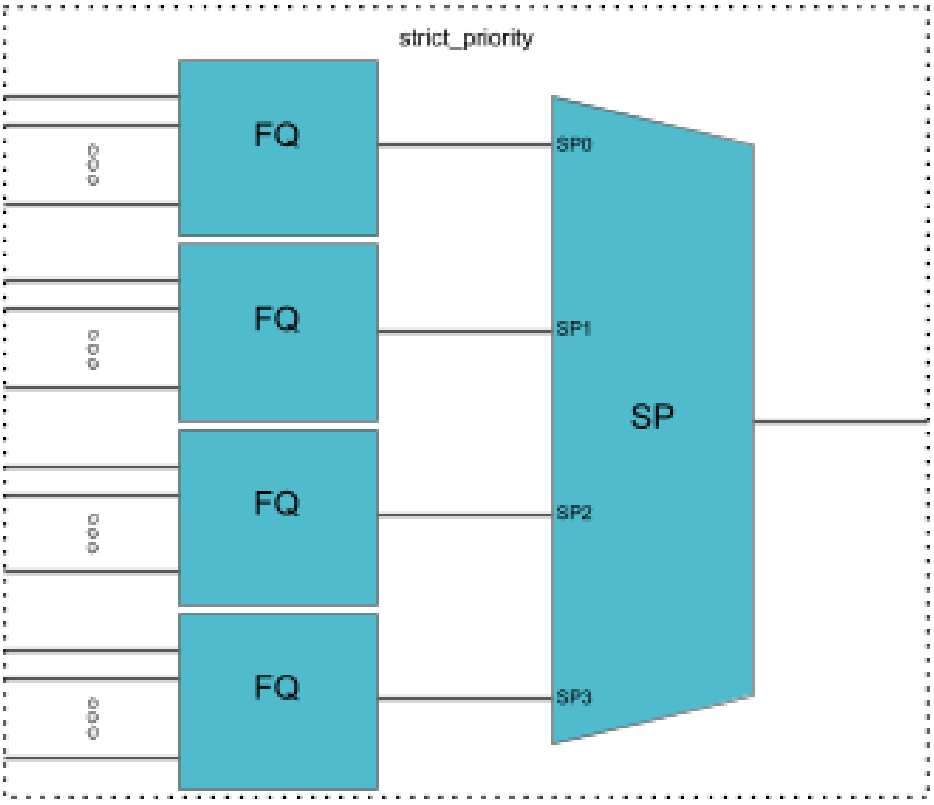
- type **“wfq_discrete_2sp”**
  - Supported weight: { 1, 2, 3 }



- **3 Strict Priority (3SP)**: maps 2-FQs and 1-WFQ
  - type: **“3sp_wfq_discrete“**
    - Supported weight: { 1, 2 }



- **4 Strict Priority (4SP)**: maps 4-FQs using SP
  - type **“strict_priority"**



2.10. Scheduler-Map

Scheduler Map defines the set of relationships between parents and children in egress scheduling hierarchy. Child in a Scheduler Map configuration can be either Queue or Scheduler. Whereas parents in a Scheduler Map configuration can be either Port or Scheduler (from the same Scheduler-Map or physical interface Scheduler-Map).

Connection Point and Weight

Child-queue or child-scheduler in a scheduler map configuration is connected to the parent-scheduler at "**connection point (CP)**". Connection point configuration also has "**weight**" associated with it if the parent has a WFQ scheduler corresponding to that connection point. Valid connection point value for a child to connect to parent **WFQ/FQ** scheduler is **no_priority** and to connect to parent **SP/Hybrid** scheduler is between **strict_priority_0** to **strict_priority_3** (based on number of Strict Priority points in parent scheduler).

Connection Types

There are five connection types in a scheduler map entry:

- queue_to_port
- queue_to_scheduler
- scheduler_to_scheduler
- scheduler_map_to_scheduler_map
- scheduler_to_port

- For the **queue_to_port** connection type, the scheduler has no role.
- **scheduler_map_to_scheduler_map** is used to statically map schedulers from 2 different scheduler-maps (that is, per IFL scheduler-map scheduler to Physical Interface scheduler-map scheduler).

2.11. Shaper

A shaper configuration defines the shaping rate in kilo-bits-per-seconds (kbps).

RADIUS Controlled Dynamic Shapers

For RBFS RADIUS services, dynamic shaper updates are required. The dynamic shaper values will be provided via existing subscriber QoS objects and must affect only the QoS instance of the corresponding subscriber but not other subscribers using the same QoS profile.

2.12. Profiles

A profile configuration defines the QoS profile that is attached to either a Subscriber interface or an L3 interface.

Profile maps the following QoS constructs to a Subscriber or an L3 interface:

- Classifier
- Class Policer Map
- Policer
- Class Queue Map
- Scheduler Map
- Remark Map

2.13. Multi-level H-QoS : Level-1 to Level-5

The following HQoS levels are required to build internet access services like FTTH, FTTC, or FTTB:

Level-1 (IFP)

Physical Interface Shaper.

Level-2 (PON TREE)

Each PON tree is a TDM based shared medium with typically ~2.5 GBit/s (GPON) shared by up to 32 consumers (ONT or DPU).

Level-3 (DPU)

In case of FTTB there is a single DPU with multiple consumers via G.Fast DSL connected which requires an additional hierarchy. This level is not needed for FTTH or FTTC.

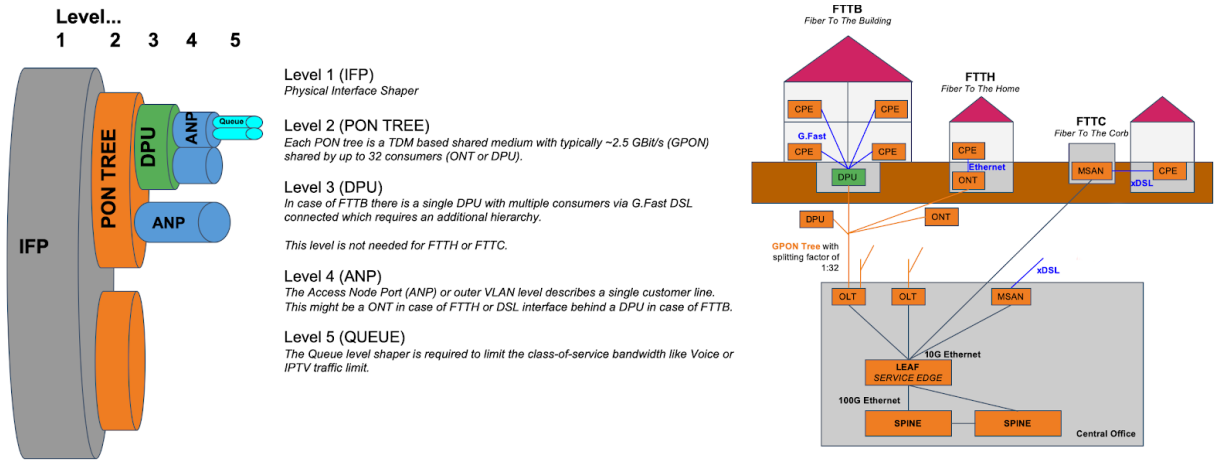
Level-4 (ANP or Session)

The Access Node Port (ANP) or outer VLAN level describes a single customer line. This might be an ONT in case of FTTH or DSL interface behind a DPU in case of FTTB. This level can be also represented on PPPoE sessions as long as just one session is permitted per VLAN.

Level-5 (QUEUE)

The Queue level shaper is required to limit the class-of-service bandwidth like Voice or IPTV traffic limit.

The figure below shows the diagram along with QoS representing Level-1 to Level-5 Hierarchical scheduling.

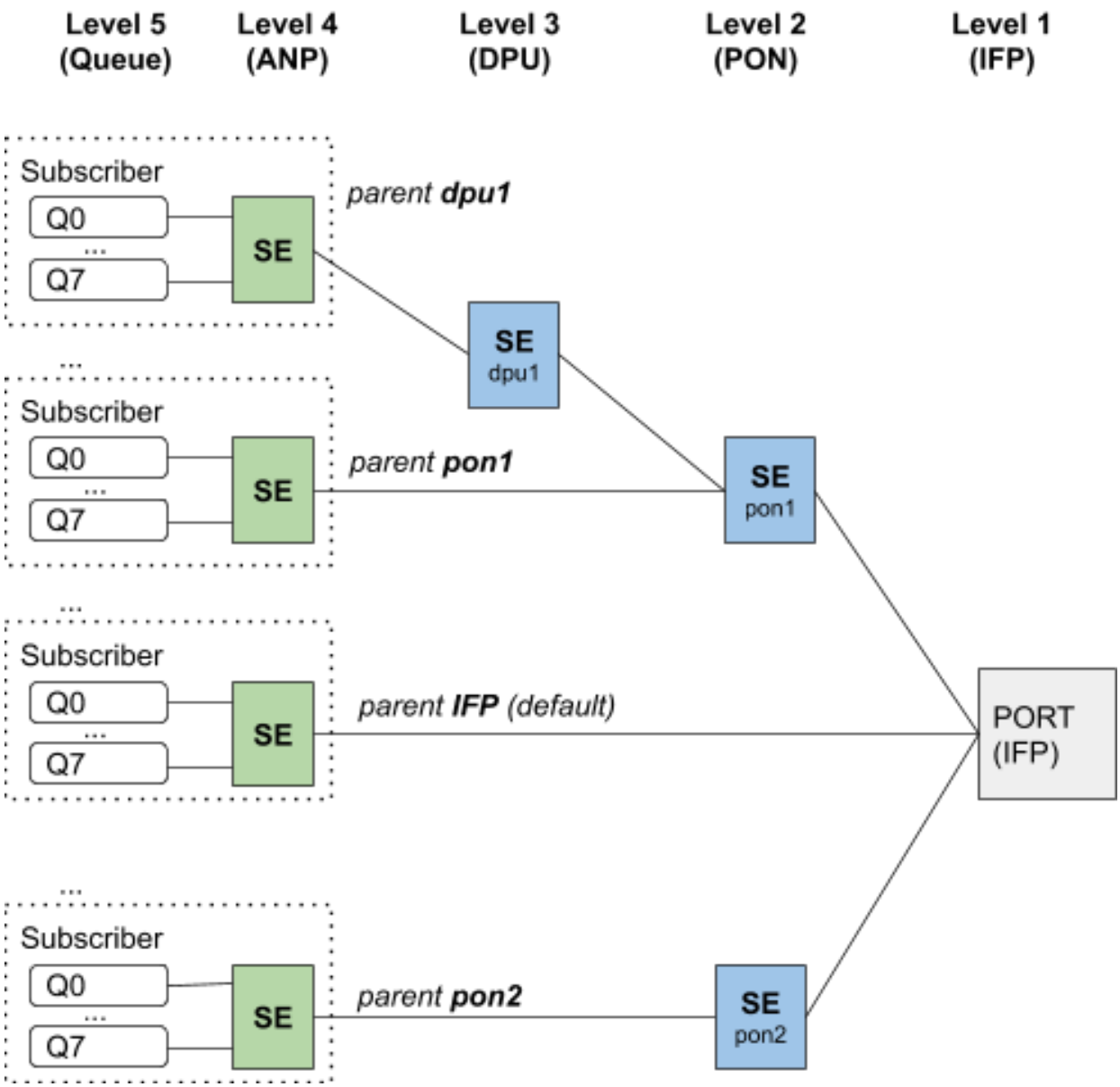


The levels 4 and 5 are configured per logical interface (i.e. subscriber-ifl or l3-ifl). Separate scheduler-map representing levels 1 to 3 connectivity shall be statically configured and mapped to corresponding physical interface (IFP).

Child scheduler in a subscriber scheduler-map is connected to parent scheduler in physical interface scheduler-map in following two ways:

- Statically using CLI.
- Dynamically via RADIUS in case of dynamic subscribers like PPPoE sessions (Subscriber-IFL).

The figure below shows the same details as the preceding figure before with the different levels but from the DPU-PON-IFP scheduler-map point of view.


Level 5
(Queue)
Level 4
(ANP)
Level 3
(DPU)
Level 2
(PON)
Level 1
(IFP)
Subscriber
Q0
...
Q7
SE
parent dpu1
SE
dpu1
parent pon1
SE
pon1
parent IFP (default)
PORT
(IFP)
parent pon2
SE
pon2

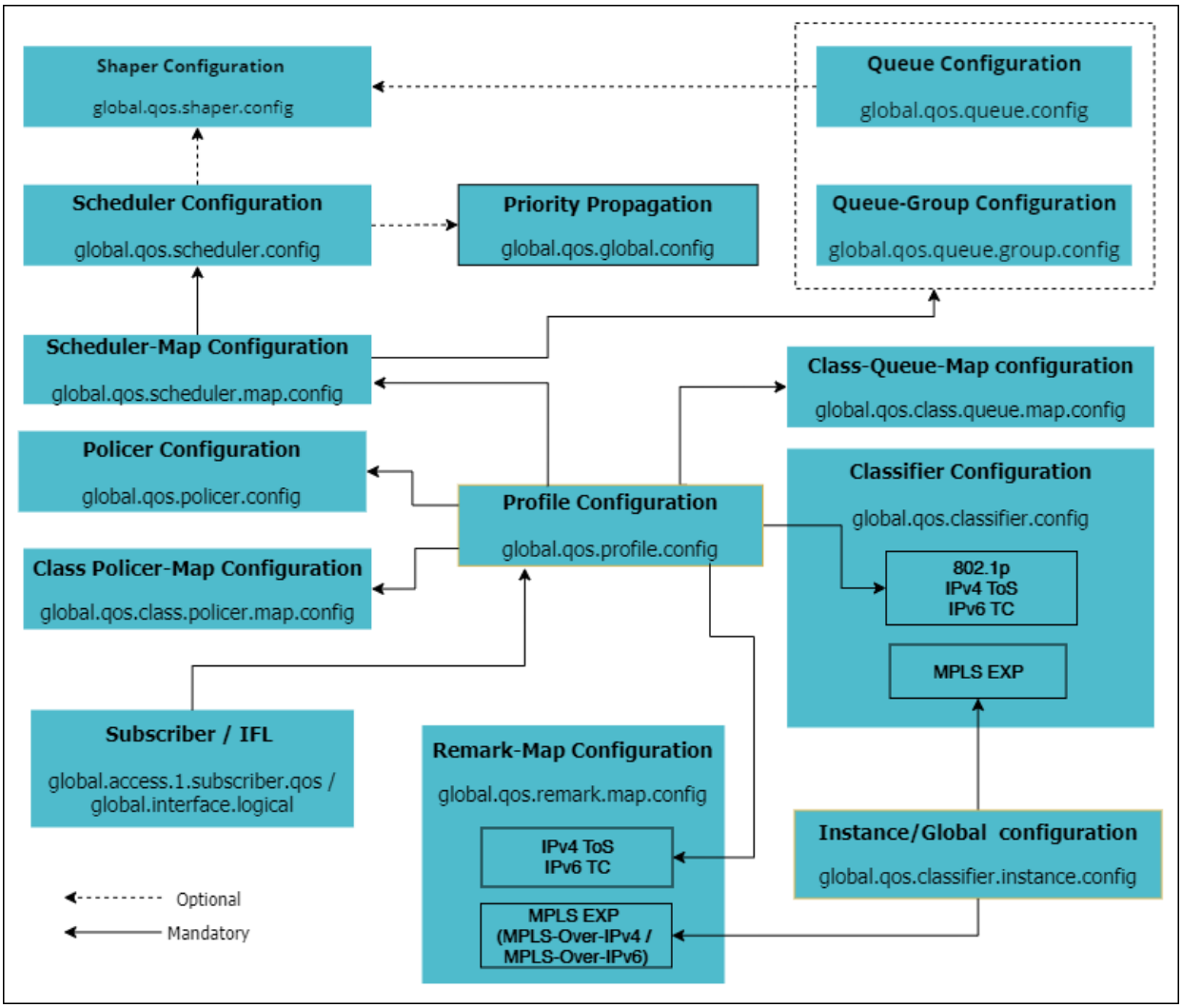
3. Configuring HQoS

To configure HQoS, perform the following steps which include creating a QoS profile and enabling QoS on a PPP Subscriber-Interface or L3-Interface.

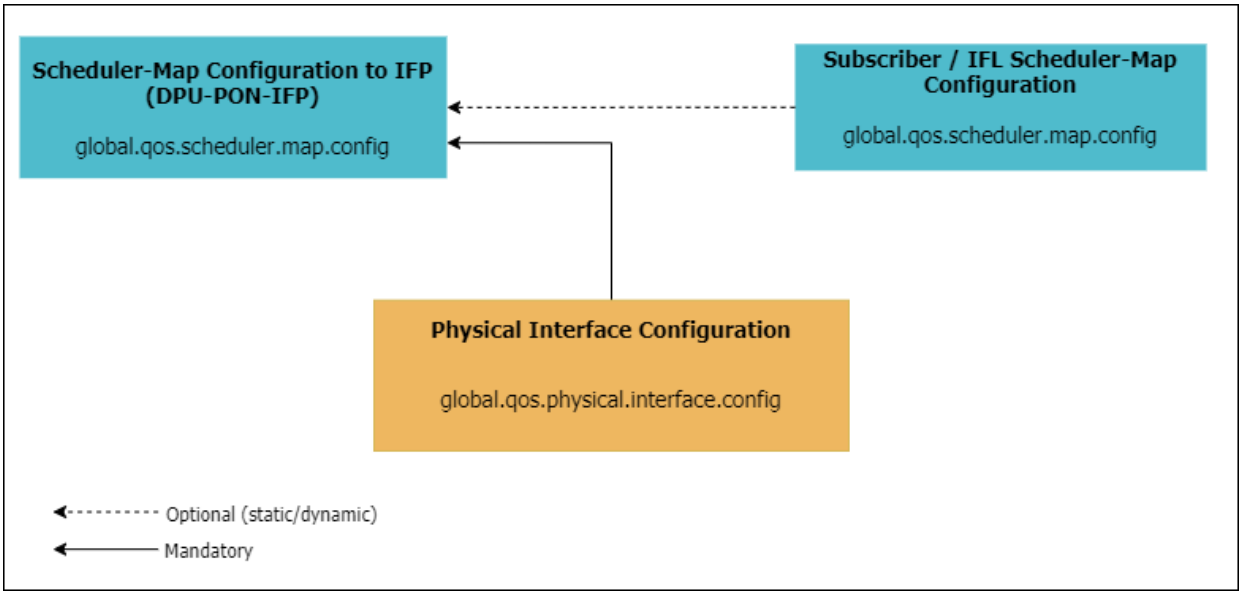
1. Create a Behavioral Aggregate (BA) classifier to classify the network traffic at the ingress.
2. Create necessary class-to-policer-map to map the classes to policer-levels (mandatory for policing).
3. Create a policer to police the classified traffic at the ingress.
4. Create necessary class-to-queue-map to map the classes to queues (mandatory for queuing).
5. Create queue-groups and configure the queue numbers (4/8) in the queue group.
6. Create necessary queues with proper size to queue the classified traffic at the egress.
7. Specify scheduler(s) with type as required.
8. (Optional) Attach a shaper to queue(s) and/or scheduler(s).
9. Specify a scheduler map to define set of relationships between parent (scheduler or port) and child (queue/queue-group or scheduler) at the egress.
10. Define a QoS profile with classifier, class-policer-map, policer, class-queue-map, scheduler-map, and remark-map based on user requirements.
11. Specify another scheduler map to represent level-3 to level-5 hierarchy in multi-level HQoS.
12. Map the MPLS EXP classifier either to an instance or configure it as global entity.
13. Map the MPLS-Over-IPv4/IPv6 remark-map either to an instance or configure it as global entity.

Priority propagation is enabled by default. To disable the Priority Propagation, we recommend doing this at the beginning and not during an active session.

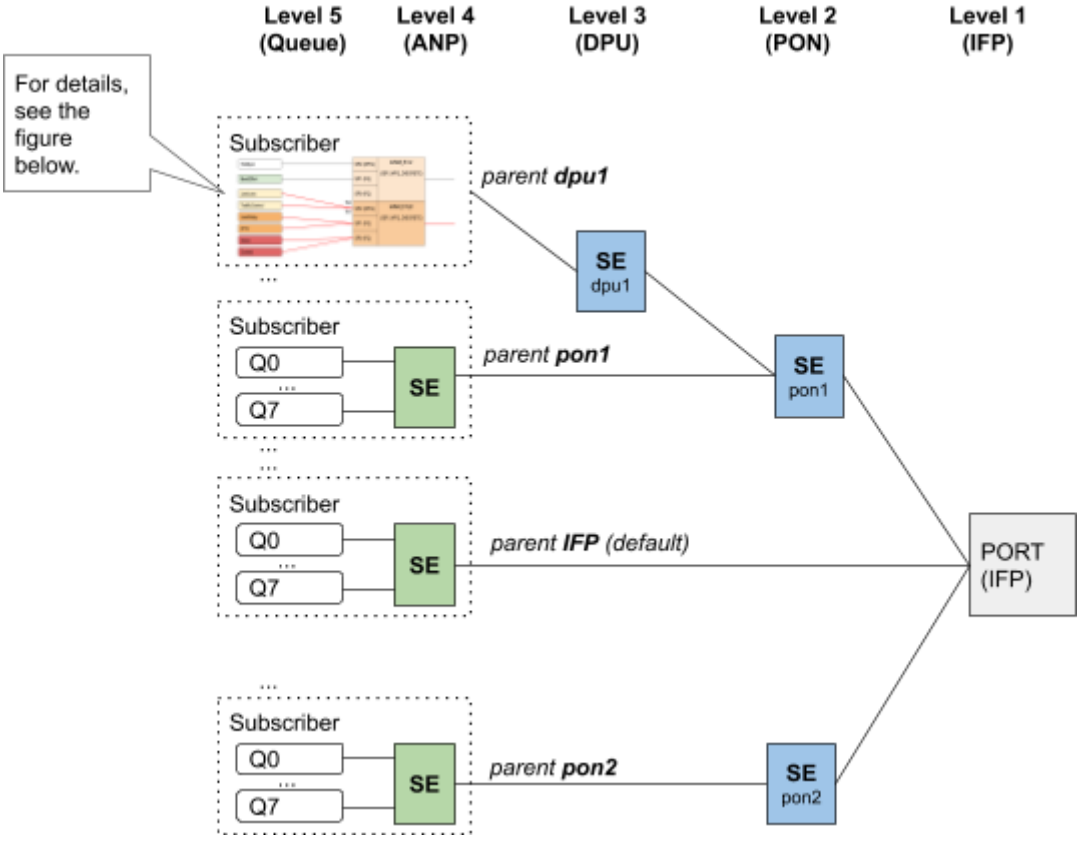
The figure below shows the dependencies between the various HQoS configuration elements.

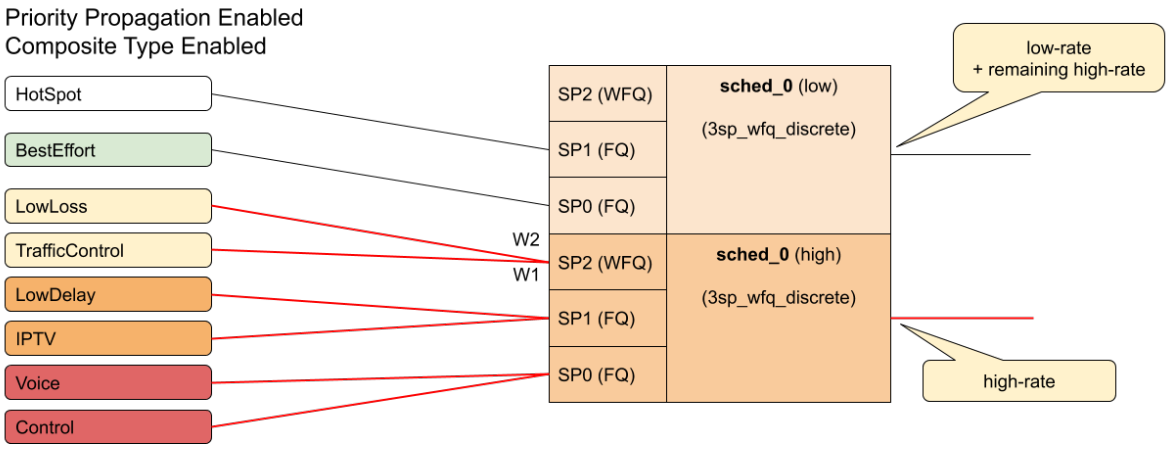


The figure below shows the additional dependencies for Multi-level HQoS.



The figures below show the scheduling hierarchy example.





The following sections provide the commands and examples for configuring HQoS.

- Classifier Configuration
- Remark-Map configuration
- Policer Configuration
- Class Policer-Map Configuration
- Queue Configuration
- Class Queue-Map Configuration
- Queue-Group Configuration

3.1. Classifier Configuration

Syntax

[forwarding-options class-of-service]
set classifier classifier-name <classifier_name> **match-type** <match_type> **codepoint** <codepoint> **class** <class> [**remark-codepoint** <remark_codepoint>]

IPv4/IPv6 classifiers are applied on a subscriber-ifl or l3ifl using the Profile Name.

Syntax

[forwarding-options class-of-service]
set profiles profile-name <profile_name> **classifer-name** <classifer_name>

MPLS Exp classifier is applied either globally or per-instance (to support multiple VPN marking schemes) using Classifier Name.

Command arguments

<classifier_name>	Specifies the classifier user-defined name
<match_type>	Specifies the type of traffic to classify, that is, ipv4-tos, ipv6-tc, ieee-802.1, exp
<codepoint>	Specifies the code-point value based on match-type
<class>	Specifies the traffic class as as class-0, class-1, class-2, class-3, class-4, class-5, class-6, and class-7
<remark_codepoint>	Specifies the remarking codepoint

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set classifier classifier-name TC_voice1 match-type ipv6-
tc codepoint 96 class class-1
```

Global Configuration

[forwarding-options class-of-service]
set global classifier-name <classifier_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global classifier-name TC_voice1
```

Instance Configuration

[forwarding-options class-of-service]
set instance instance-name <instance_name> **classifier-name** <classifier_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set instance instance-name ip2vrf classifier-name
TC_voice1
```

3.2. Remark-Map configuration

Syntax

[forwarding-options class-of-service]
set forwarding-options class-of-service remark-map remark-map-name <remark_map_name> **remark-type** <remark_type> **match-codepoint** <match_codepoint> **color** <color> **remark-codepoint** <remark_codepoint>

Command arguments

<remark_map_name>	Specifies the remaking map name

<match_codepoint>	Specifies the match code point
<color>	Indicates the colour - all, green, yellow. Colour is used to set different remark-codepoint for same match-codepoint based on color marked by the Policer.
<remark_type>	Specifies the remarking type - ipv4-tos, ipv6-tc, mpls-over-ipv4, mpls-over-ipv6
<remark_codepoint>	Specifies the remarking codepoint

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set remark-map remark-map-name remark-exp remark-type
ipv6-tc match-codepoint 100 color green remark-codepoint 224
```

Global Configuration

[forwarding-options class-of-service]
set global remark-map-name <remark_map_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global remark-map-name remark-exp
```

Instance Configuration

[forwarding-options class-of-service]
set instance instance-name <instance_name> **remark-map-name** <remark_map_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set instance instance-name ip2vrf remark-map-name remark-
exp
```

3.3. Policer Configuration

Syntax

[forwarding-options class-of-service]
root@rtbrick:confd> **edit policer policer-name** <policer_name>
[forwarding-options class-of-service policer policer-name voice_policer]

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit policer policer-name voice_policer
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd>
```

Levels configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set levels <levels>

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set levels 4
```

Type configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set type <type>

Example

```
[ forwarding-options class-of-service policer policer-name Policer_Upstream ]
root@rtbrick:confd> set type two-rate-three-color
```

Flag configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set flags <flag>

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set flags color-blind
```

Level rates configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set level1-rates cir <level1_cir> **pir** <level1_pir> **cbs** <level1_cbs> **pbs** <level1_pbs>

The same is applicable for level-2 to level-4.

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set level1-rates cir 4000 pir 5000 cbs 400 pbs 600

root@rtbrick:confd> set level2-rates cir 4000 pir 5000 cbs 400 pbs 600
```

Level rates and max-rates configuration

set level1-rates cir <level1_cir> **pir** <level1_pir> **cbs** <level1_cbs> **pbs** <level1_pbs> **max-cir** <level1_max_cir> **max-pir** <level1_max_pir>

The same is applicable for level-2 to level-4.

Example

```
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set level2-rates cir 4000 pir 5000 cbs 400 pbs 600 max-
cir 4500 max-pir 5500
```

Command Arguments

<policer-name>	Specifies the policer name.
<levels>	Specifies levels in the Policer. Levels will be from 1 to 4.
<type>	Specifies the policer type.
<flag>	Set flags.
<level1_cir>	Set committed information rate (CIR) in kilobits per second (kbps) for level-1. The same is applicable for level-2 to level-4.
<level1_pir>	Set peak information rate (PIR) in kilobits per second (kbps) for level-1. The same is applicable for level-2 to level-4.
<level1_cbs>	Set Committed burst size (CBS) in kilobits for level-1. The same is applicable for level-2 to level-4.
<level1_pbs>	Set peak burst size (PBS) in kilobits for level-1. The same is applicable for level-2 to level-4.
<level1_max_cir>	Set the maximum for the level-1 committed information rate (CIR) in kilobits per second (kbps). The same is applicable for level-2 to level-4.
<level1_max_pir>	Set the maximum for the level-1 peak information rate (PIR) in kilobits per second (kbps). The same is applicable for level-2 to level-4.

3.4. Class Policer-Map Configuration

Syntax

[forwarding-options class-of-service]
root@rtbrick:confd> **edit class-policer-map class-policer-map-name** <class_policer_map_name> **class <class> policer-level** <policer_level>

Command arguments

class_policer_map_name>	Specifies the class policer map name,
<class>	Specifies the class such as class-0, class-1, class-2, class-3, class-4, class-5, class-6, class-7
<policer_level>	level-1, level-2, level-3, level-4

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set class-policer-map class-policer-map-name
classifier_residential class class-4 policer-level level-2
[ forwarding-options class-of-service ]
root@rtbrick:confd> set class-policer-map class-policer-map-name
classifier_residential class class-7 policer-level level-4
```

3.5. Queue Configuration

Syntax

[forwarding-options class-of-service]
edit queue queue-name <queue_name>
[forwarding-options class-of-service queue queue-name]

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit queue queue-name IPTV
```

Queue Size

set queue_size <queue_size>

Example

```
[ forwarding-options class-of-service queue queue-name IPTV ]
root@rtbrick:confd> set queue-size 6000
```

Queue WRED Profile

set wred minimum-threshold <minimum_threshold> **maximum-threshold** <maximum_threshold> **drop-probability** <drop_prob>

Example

```
[ forwarding-options class-of-service queue queue-name IPTV ]
root@rtbrick:confd> set wred minimum-threshold 15000 maximum-threshold 18000
drop-probability 70
```

Queue Shaper

set shaper-name <shaper_name>

Example

```
[ forwarding-options class-of-service queue queue-name IPTV ]
root@rtbrick:confd> set shaper-name session_shaper
```

Command arguments

<queue_name>	Specifies the user-defined queue name	
<queue_size>	Specifies the size of the queue in bytes	
<shaper_name>	(Optional) Specifies the shaper that is associated with the queue	
WRED	<minimum_threshold>	Specifies the minimum average queue size to apply WRED in bytes
	<maximum_threshold>	Specifies the maximum average queue size to apply WRED in bytes
	<drop_prob>	WRED drop probability applied at the maximum threshold

3.6. Class Queue-Map Configuration

3.6.1. Class to Queue mapping

Syntax

[forwarding-options class-of-service]
edit class-queue-map class-queue-map-name <class_queue_map_name>
class <class> **queue_name** <queue_name>

Command arguments

<class_queue_map_name>	Specifies the class queue map name
<class>	Specifies the class such as class-0, class-1, class-2, class-3, class-4, class-5, class-6, class-7

Example

```
[ forwarding-options class-of-service class-queue-map class-queue-map-name
classifier_residential class class-4 ]
root@rtbrick:confd> set queue-name BestEffort
[ forwarding-options class-of-service class-queue-map class-queue-map-name
classifier_residential class class-4 ]
```

3.7. Queue-Group Configuration

Queue group size - 4 or 8

Syntax

[forwarding-options class-of-service]
edit queue-group queue-group-name <queue_group_name>
set queue-numbers <queue_numbers>

Command arguments

<queue_group_name>	User-defined name for the queue-group
<queue_numbers>	Specifies the number of queues in a Queue Group

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit queue-group queue-group-name queue_group_residential
[<Enter>] Execute the command
[ forwarding-options class-of-service ]
[ forwarding-options class-of-service queue-group queue-group-name
queue_group_residential ]
root@rtbrick:confd> set queue-numbers 8
```

3.8. Scheduler Configuration

Syntax

[forwarding-options class-of-service]
edit scheduler scheduler-name <scheduler_name>
[forwarding-options class-of-service scheduler scheduler-name sched_1]

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit scheduler scheduler-name sched_1
```

Scheduler Type

set type <type> [composite]

Example

```
[ forwarding-options class-of-service scheduler scheduler-name
rtbrick_sched_0 ]
root@rtbrick:confd> set type 3sp_wfq_discrete composite
```

Scheduler Shaper

set shaper-name <shaper_name>

Example

```
[ forwarding-options class-of-service scheduler scheduler-name sched_1 ]
root@rtbrick:confd> set shaper-name session_shaper
```

Command arguments

<scheduler-name>	User-defined Scheduler Name
<shaper_name>	(Optional) User-defined Shaper Name
<type>	Specifies the Scheduler Type 2sp_wfq_discrete 3sp_wfq_discrete strict_priority wfq_discrete_2sp 2sp_wfq_independent fair_queueing weighted_fair_queueing wfq_independent_2sp
<type> composite	Optional keyword to specify the scheduler as composite type

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit scheduler scheduler-name sched_1
[ forwarding-options class-of-service scheduler scheduler-name sched_1 ]

[ forwarding-options class-of-service scheduler scheduler-name
rtbrick_sched_0 ]
root@rtbrick:confd> set type 3sp_wfq_discrete composite

[ forwarding-options class-of-service scheduler scheduler-name sched_1 ]
root@rtbrick:confd> set shaper-name shaper_1
```

3.9. Scheduler-Map Configuration

Syntax

[forwarding-options class-of-service]
root@rtbrick:confd> **edit scheduler-map scheduler-map-name** <scheduler_map_name> [**queue-group-name** <queue_group_name> **queue-name** <queue_name> | **scheduler-name** <scheduler_name>]

Command arguments

<scheduler_map_name>	User-defined Scheduler-Map Name
<scheduler_name>	User-defined Scheduler Name
<queue_group_name>	User-defined Queue-Group Name
<queue_name>	User-defined Queue-Name

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit scheduler-map scheduler-map-name
rtbrick_sched_map_residential queue-group-name queue_group_residential queue-
name IPTV
```

3.9.1. Scheduler to Port Configuration

Syntax

set scheduler-map-name <scheduler_map_name> **scheduler-name** <scheduler_name> **port-connection**

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set scheduler-name rtbrick_sched_0 port-connection
```

3.9.2. Scheduler to Scheduler (same Scheduler-Map)

Syntax

set parent-scheduler <parent-scheduler> **connection-point** <connection_point> [weight <weight>]

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set scheduler-name rtbrick_sched_0 parent-scheduler
rtbrick_sched_1 connection-point strict_priority_0 weight 1
```

3.9.3. Scheduler to Scheduler (different Scheduler-Map)

Syntax

set parent-scheduler <parent-scheduler> **connection-point** <connection_point> [weight <weight>] **scheduler-map-connection**

3.9.4. Queue to port

Syntax

set queue-group-name <queue_group_name> **queue-name** <queue_name> **port-connection**

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set queue-group-name queue_group_residential queue-name
IPTV port-connection
```

3.9.5. Queue to scheduler

Syntax

set queue-group-name <queue_group_name> **queue-name** <queue_name> **parent-scheduler** <parent_scheduler_name> **parent-flow** <parent_flow> **connection-point** <connection_point> [weight <weight>]

The parent-flow configuration is optional.

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential ]
root@rtbrick:confd> set queue-group-name queue_group_residential queue-name
IPTV parent-scheduler rtbrick_sched_0 connection-point strict_priority_0
weight 1
```

3.10. Shaper Configuration

Syntax

[forwarding-options class-of-service]
edit shaper shaper-name <shaper_name>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit shaper shaper-name session_shaper
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd>
```

High Flow Shaping Rate

To configure only high-flow shaping rate, enter the following command:

rtb confd set forwarding-options class-of-service shaper shaper-name <shaper_name> **shaping-rate-high** <high_flow_rate>

Example

```
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd> set shaping-rate-high 9000
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
```

Low Flow Shaping Rate

To configure only low-flow shaping rate, enter the following command:

rtb confd set forwarding-options class-of-service shaper shaper-name <shaper_name> **shaping-rate-low** <low_flow_rate>

if priority propagation is not enabled, high-flow shaping value will be considered for shaper.

Example

```
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd> set shaping-rate-low 10000
[ forwarding-options class-of-service shaper shaper-name session_shaper ]
root@rtbrick:confd>
```

Command Arguments

<shaper_name>	User-defined shaper name
<high_flow_rate>	High flow shaping rate in kilobits per second
<low_flow_rate>	Low flow shaping rate in kilobits per second

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit shaper shaper-name shaper_1
[ forwarding-options class-of-service shaper shaper-name shaper_1 ]
root@rtbrick:confd> set
 shaping-rate                              Set Rate in kbps


root@rtbrick:confd> set shaping-rate 10000
[ forwarding-options class-of-service shaper shaper-name shaper_1 ]
```

3.11. Priority Propagation

Syntax

rtb confd set forwarding-options class-of-service global priority-propagation <enable|disable>

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global priority-propagation enable
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global priority-propagation dis
[<Enter>] Execute the command
[ forwarding-options class-of-service ]
root@rtbrick:confd> set global priority-propagation disable
[ forwarding-options class-of-service ]
root@rtbrick:confd>
```

3.12. Profiles Configuration

Syntax

[forwarding-options class-of-service]
edit profiles profile-name <profile_name>

[forwarding-options class-of-service profiles profile-name <profile_name>]

Command arguments

<profile_name>	User-defined QoS Profile name

Example

```
[ forwarding-options class-of-service ]
root@rtbrick:confd> edit profiles profile-name rtbrick_residential_profile
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
```

Classifier Name

set classifier-name <classifier_name>

Command arguments

<classifier_name>	User-defined Classifier name

Example

```
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd> set classifier-name Classifier_IP
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
```

Class Policer-Map

set class-policer-map-name <class_policer_map_name>

Command arguments

<class_policer_map_name>	User-defined policer map name

Example

```
[ forwarding-options class-of-service class-policer-map ]
root@rtbrick:confd> set class-policer-map-name class-policer1 class class-4
```

Policer

set policer-name <policer_name>

Command arguments

<policer_name>	User-defined Policer name

Example

```
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd> set policer-name Policer_Upstream
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
```

Class Queue-Map

set class-queue-map-name <class_queue_map_name>

Command arguments

<class_queue_map_name>	User-defined queue map name

Example

```
[ forwarding-options class-of-service class-queue-map ]
root@rtbrick:confd> set class-queue-map-name classifier_residential class
class-4
```

Scheduler-Map

set scheduler-map-name <scheduler_map_name>

Command arguments

<scheduler_map_name>	User-defined Scheduler map name

Example

```
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd> set scheduler-map-name rtbrick_sched_map_residential
[ forwarding-options class-of-service profiles profile-name
rtbrick_residential_profile ]
root@rtbrick:confd>
```

Remark-Map

set remark-map-name <remark_map_name>

Command arguments

<remark_map_name>	Remarking map name

Example

```
[ forwarding-options class-of-service remark-map ]
root@rtbrick:confd> set remark-map-name remark-exp
```

3.13. Interface Configuration

Configuring L3 Interface

QOS Profile can be mapped to an L3 interface (that is, IFL).

Syntax

root@rtbrick:confd> **edit interface physical ifp-0/0/2 logical unit 1**
[interface physical ifp-0/0/2 logical unit 1]

root@rtbrick:confd> **set class-of-service profile-name profile_up**
[interface physical ifp-0/0/2 logical unit 1]

root@rtbrick:confd>

Command arguments

<ifp-Name>	Physical Interface Name
<logical-unit-id>	Logical Interface Unit

<profile-Name>	Name of the QoS Profile

Example

```
root@rtbrick:confd> edit interface physical ifp-0/0/4 logical unit 0
[ interface physical ifp-0/0/4 logical unit 0 ]
root@rtbrick:confd> set class-of-service profile-name
rtbrick_residential_profile
[ interface physical ifp-0/0/4 logical unit 0 ]
root@rtbrick:confd>
```

4. HQoS Show Running-Configuration

To display the running configuration, use the **show running-configuration** command.

Syntax

show running-configuration

```
"forwarding-options": [
      {
        "class-of-service": [
          {
            "classifier": [
              {
                "classifier-name:Classifier_EXP": {
                  "match-type:exp": {
                    "codepoint:0": {
                      "class": "class-0"
                    },
                    "codepoint:1": {
                      "class": "class-1"
                    },
                    "codepoint:2": {
                      "class": "class-2"
                    },
                    "codepoint:3": {
                      "class": "class-3"
                    },
                    "codepoint:4": {
                      "class": "class-4"
                    },
                    "codepoint:5": {
                      "class": "class-5"
                    },
                    "codepoint:6": {
                      "class": "class-6"
                    },
                    "codepoint:7": {
                      "class": "class-7"
                    }
                  }
                },
                "classifier-name:Classifier_IP": {
                  "match-type:ipv4-tos": {
                    "codepoint:8": {
                      "class": "class-0"
                    },
                    "codepoint:16": {
                      "class": "class-1"
                    },
                    "codepoint:32": {
```

```
                    "class": "class-2"
                  },
                  "codepoint:64": {
                    "class": "class-3"
                  },
                  "codepoint:96": {
                    "class": "class-4"
                  },
                  "codepoint:164": {
                    "class": "class-5"
                  },
                  "codepoint:196": {
                    "class": "class-6"
                  },
                  "codepoint:224": {
                    "class": "class-7"
                  }
                }
              },
              "classifier-name:Classifier_PBIT": {
                "match-type:ieee-802.1": {
                  "codepoint:0": {
                    "class": "class-0"
                  },
                  "codepoint:1": {
                    "class": "class-1"
                  },
                  "codepoint:2": {
                    "class": "class-2"
                  },
                  "codepoint:3": {
                    "class": "class-3"
                  },
                  "codepoint:4": {
                    "class": "class-4"
                  },
                  "codepoint:5": {
                    "class": "class-5"
                  },
                  "codepoint:6": {
                    "class": "class-6"
                  },
                  "codepoint:7": {
                    "class": "class-7"
                  }
                }
              }
            }
          ],
          "remark-map": [
            {
              "remark-map-name:remark-exp": {
                "remark-type:mpls-over-ipv4": {
                  "match-codepoint:64": {
                    "color:green": {
                      "remark_codepoint": 6
                    }
                  }
```

```
                },
                "remark-type:mpls-over-ipv6": {
                  "match-codepoint:64": {
                    "color:green": {
                      "remark_codepoint": 6
                    }
                  }
                }
              }
            }
          ],
          "class-policer-map": [
            {
              "class-policer-map-name:residential_policer_classifier": {
                "class:class-0": {
                  "policer_level": "level-2"
                },
                "class:class-1": {
                  "policer_level": "level-1"
                },
                "class:class-2": {
                  "policer_level": "level-1"
                },
                "class:class-3": {
                  "policer_level": "level-1"
                },
                "class:class-4": {
                  "policer_level": "level-1"
                },
                "class:class-5": {
                  "policer_level": "level-1"
                },
                "class:class-6": {
                  "policer_level": "level-1"
                },
                "class:class-7": {
                  "policer_level": "level-1"
                }
              }
            }
          ],
          "policer": [
            {
              "policer-name:Policer_Upstream": {
                "type": "two-rate-three-color",
                "flags": "color-blind",
                "levels": 4,
                "level1_cir": 1000,
                "level1_pir": 2000,
                "level1_cbs": 500,
                "level1_pbs": 750,
                "level1_max_cir": 2000,
                "level1_max_pir": 2000
              }
            }
          ],
          "queue": [
            {
```

```
            "queue-name:BestEffort": {
              "queue_size": 25000
            },
            "queue-name:Control": {
              "queue_size": 25000
            },
            "queue-name:HotSpot": {
              "queue_size": 25000
            },
            "queue-name:IPTV": {
              "queue_size": 25000
            },
            "queue-name:LowDelay": {
              "queue_size": 25000
            },
            "queue-name:LowLoss": {
              "queue_size": 25000
            },
            "queue-name:TrafficControl": {
              "queue_size": 25000
            },
            "queue-name:Voice": {
              "queue_size": 25000
            }
          }
        ],
        "shaper": [
          {
            "shaper-name:control_shaper": {
              "shaper_rate_high": 3000,
              "shaper_rate_low": 6000
            },
            "shaper-name:session_shaper": {
              "shaper_rate_high": 2000,
              "shaper_rate_low": 4000
            },
            "shaper-name:voice_shaper": {
              "shaper_rate_high": 2500,
              "shaper_rate_low": 5000
            }
          }
        ],
        "class-queue-map": [
          {
            "class-queue-map-name:classifier_residential": {
              "class:class-0": {
                "queue_name": "HotSpot"
              },
              "class:class-1": {
                "queue_name": "BestEffort"
              },
              "class:class-2": {
                "queue_name": "LowLoss"
              },
              "class:class-3": {
                "queue_name": "TrafficControl"
              },
              "class:class-4": {
```

```
                "queue_name": "LowDelay"
              },
              "class:class-5": {
                "queue_name": "IPTV"
              },
              "class:class-6": {
                "queue_name": "Voice"
              },
              "class:class-7": {
                "queue_name": "Control"
              }
            }
          }
        ],
        "queue-group": [
          {
            "queue-group-name:queue_group_residential": {
              "queue_numbers": 8
            }
          }
        ],
        "scheduler": [
          {
            "scheduler-name:dpu_sched_1": {
              "type": "fair_queueing"
            },
            "scheduler-name:pon_sched_1": {
              "type": "fair_queueing"
            },
            "scheduler-name:rtbrick_sched_0": {
              "shaper_name": "session_shaper",
              "type": "3sp_wfq_discrete",
              "is_composite": true
            }
          }
        ],
        "profile": [
          {
            "profile-name:rtbrick_residential_profile": {
              "classifier_name": "Classifier_IP",
              "class_policer_map_name": "residential_policer_classifier",
              "policer_name": "Policer_Upstream"
            }
          }
        ],
        "interface": [
          {
            "physical:ifp-0/0/3": {
              "scheduler_map_name": "dpu1"
            }
          }
        ],
        "instance": [
          {
            "instance-name:ip2vrf": {
              "remark_map_name": "remark-exp"
            }
          }
```

```
            ],
            "scheduler-map": [
              {
                "scheduler-map-name:dpu1": {
                  "scheduler-map-scheduler:dpu_sched_1": {
                    "parent_scheduler_name": "pon_sched_1"
                  },
                  "scheduler-map-scheduler:pon_sched_1": {
                  }
                },
                "scheduler-map-name:rtbrick_sched_map_residential": {
                  "scheduler-map-queue:queue_group_residential IPTV": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_1"
                  },
                  "scheduler-map-queue:queue_group_residential Voice": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_0"
                  },
                  "scheduler-map-queue:queue_group_residential Control": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_0"
                  },
                  "scheduler-map-queue:queue_group_residential HotSpot": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "parent_scheduler_flow": "low-flow",
                    "connection_point": "strict_priority_1"
                  },
                  "scheduler-map-queue:queue_group_residential LowLoss": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_2",
                    "weight": 2
                  },
                  "scheduler-map-queue:queue_group_residential LowDelay": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_1"
                  },
                  "scheduler-map-queue:queue_group_residential BestEffort": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "parent_scheduler_flow": "low-flow",
                    "connection_point": "strict_priority_0"
                  },
                  "scheduler-map-queue:queue_group_residential
TrafficControl": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_2",
                    "weight": 1
                  },
                  "scheduler-map-scheduler:rtbrick_sched_0": {
                  },
                  "scheduler-map-scheduler:rtbrick_sched_1": {
                    "parent_scheduler_name": "rtbrick_sched_0",
                    "connection_point": "strict_priority_0"
                  }
                }
              }
            ]
          }
```

```
        ]
      }
    ]
```

5. HQoS Show Commands

5.1. rtb ifmd show qos queue counters

This command displays the queue statistics of the dropped and received traffic.

Syntax

rtb ifmd show qos queue counters

Example output

```
ubuntu@rtbrick:~/QOS_115CONFIG/QoSProfile_IPv6$ rtb ifmd show qos queue
counters
+--------------------------------+--------------------+--------------------
+---------------+---------------+---------------+---------------+
             IFL Name                    Queue Grp               Queue
Rx Bytes        Rx Pkts      Dropped Bytes    Dropped Pkts
+--------------------------------+--------------------+--------------------
+---------------+---------------+---------------+---------------+
    ppp-0/0/4/72339069014638594   queue_group_resid...         IPTV
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...         Voice
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...        Control
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...        HotSpot
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...        LowLoss
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...       LowDelay
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...      BestEffort
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...    TrafficControl
0               0               0               0
ubuntu@rtbrick:~/QOS_115CONFIG/QoSProfile_IPv6$
```

5.2. rtb ifmd clear qos queue counters

This command allows you to clear the queue counters.

Syntax

rtb ifmd clear qos queue counters

Example output

```
ubuntu@rtbrick:~/QOS_115CONFIG/QoSProfile_IPv6$ rtb ifmd clear qos queue
counters
ubuntu@rtbrick:~/QOS_115CONFIG/QoSProfile_IPv6$
ubuntu@rtbrick:~/QOS_115CONFIG/QoSProfile_IPv6$ rtb ifmd show qos queue
counters
+--------------------------------+--------------------+--------------------
+---------------+---------------+---------------+---------------+
             IFL Name                    Queue Grp              Queue
Rx Bytes        Rx Pkts      Dropped Bytes    Dropped Pkts
+--------------------------------+--------------------+--------------------
+---------------+---------------+---------------+---------------+
    ppp-0/0/4/72339069014638594   queue_group_resid...         IPTV
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...         Voice
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...        Control
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...        HotSpot
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...        LowLoss
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...       LowDelay
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...      BestEffort
0               0               0               0
    ppp-0/0/4/72339069014638594   queue_group_resid...    TrafficControl
0               0               0               0
ubuntu@rtbrick:~/QOS_115CONFIG/QoSProfile_IPv6$
```