RBFS HQoS Configuration Guide

Version 20.5.1-rc0, 25 May 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

©Copyright 2020 RtBrick, Inc. All rights reserved. The information contained herein is subject to change without notice. The trademarks, logos and service marks ("Marks") displayed in this documentation are the property of RtBrick in the United States and other countries. Use of the Marks are subject to RtBrick's Term of Use Policy, available at https://www.rtbrick.com/privacy. Use of marks belonging to other parties is for informational purposes only.

Table of Contents

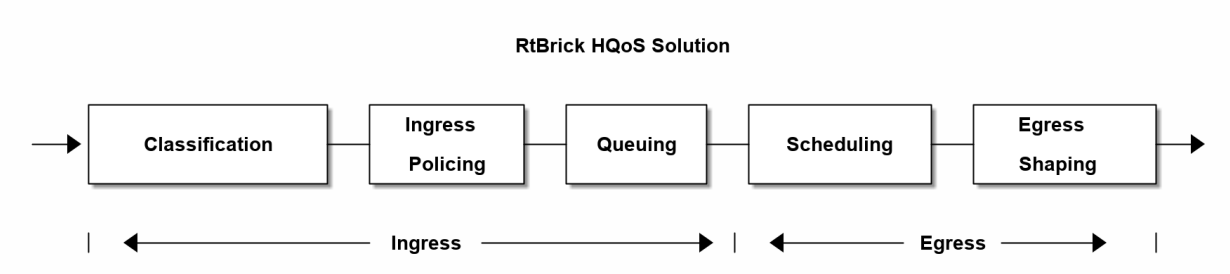
1. Introduction to Hierarchical Quality of Service (HQoS)

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 or drop. Re-marking packets based on color is not supported.
- **Shaper:** Shaper is implemented in egress to rate-limit the traffic.

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

1.1. Supported Hardware

- Edgecore AS5916-XKS

1.2. Guidelines and Limitations

- For this release, IPv4-TOS, IPv6-TC, and EXP based classification is supported.
- Uniform MPLS model is used to populate EXP bits at the time of MPLS encapsulation. With this model, MSB 3-bits from 8 bits IPv4-TOS or IPv6-TC are copied to EXP bits as shown 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

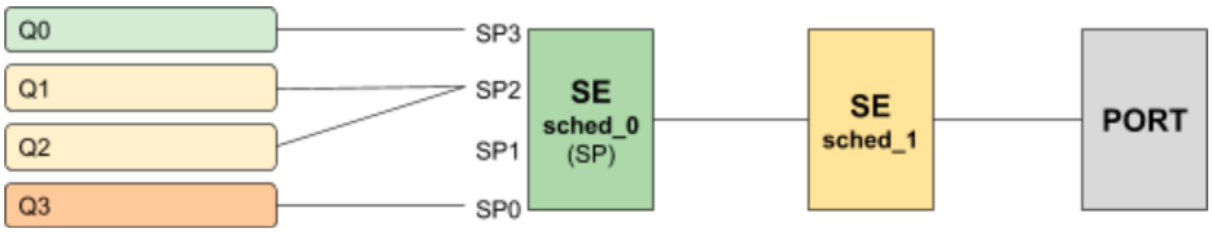
2. HQoS Features

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.



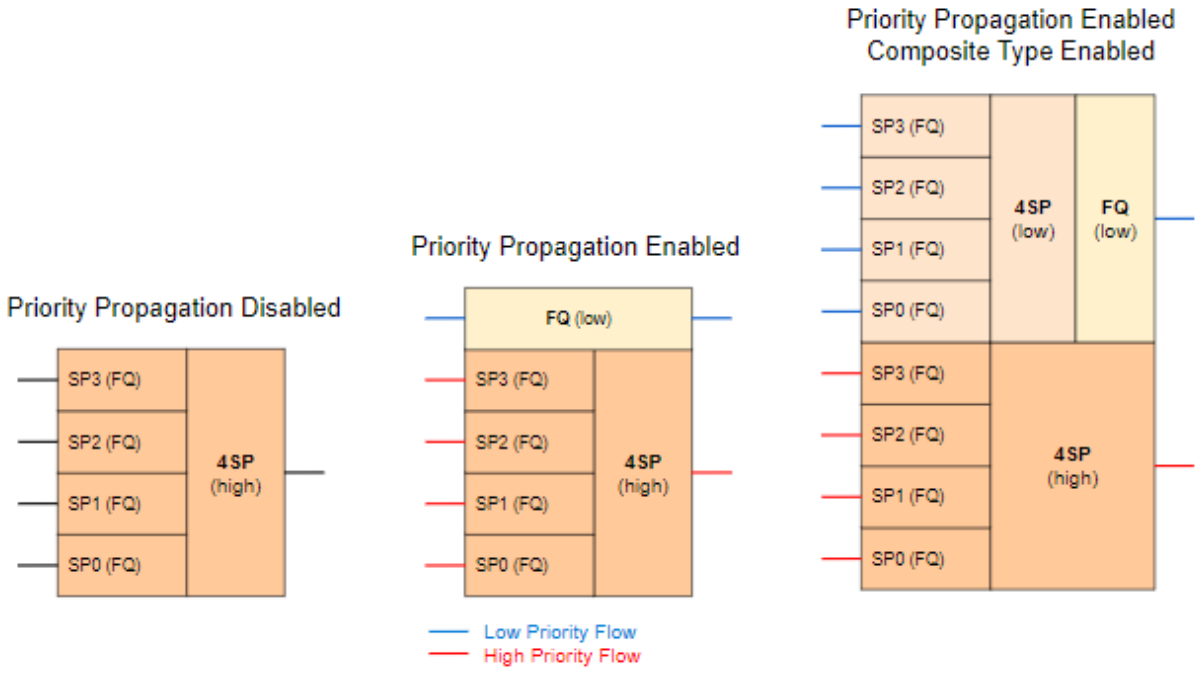
Without priority propagation each scheduler element can have multiple child connection points but just one parent connection point. Therefore traffic leaving a scheduler element can't be differentiated by the parent scheduling element. The parent scheduler element sched_1 receives the traffic from sched_0 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.

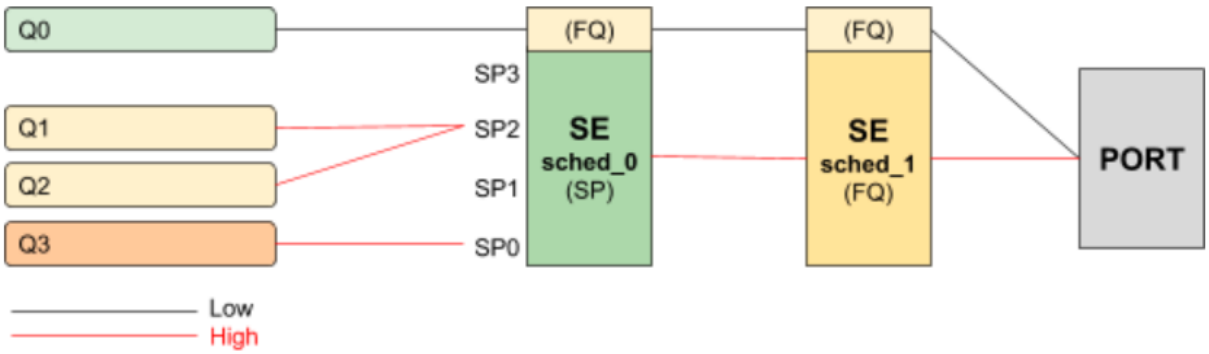


Without priority propagation enabled, each scheduler element consumes only one scheduler resource compared to two elements if enabled. The composite type consumes three scheduler elements.

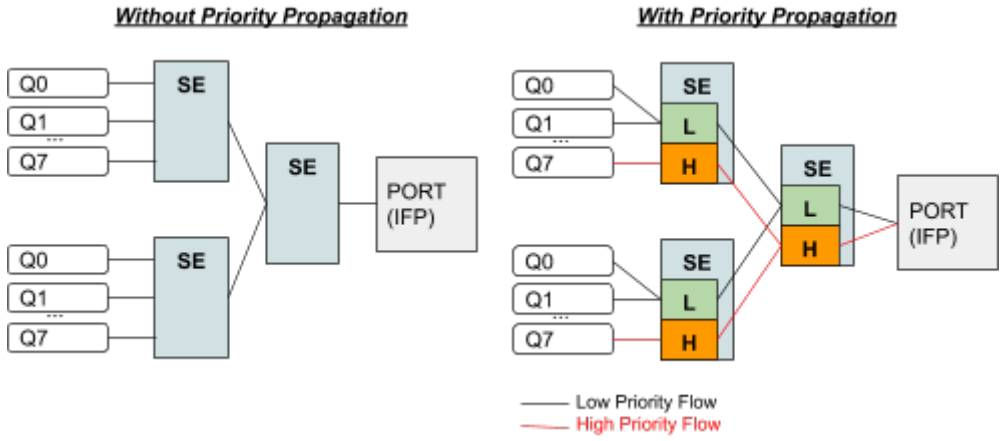
With priority propagation disabled, all traffic is considered as high priority flow.

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.



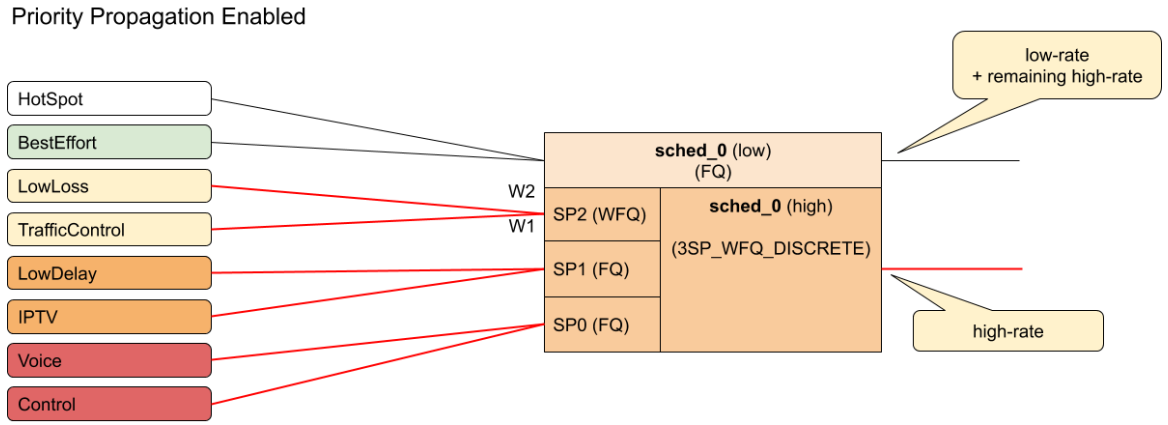
The figure below shows a typical multi level QoS configuration without priority propagation on the left and with priority propagation on the right side.

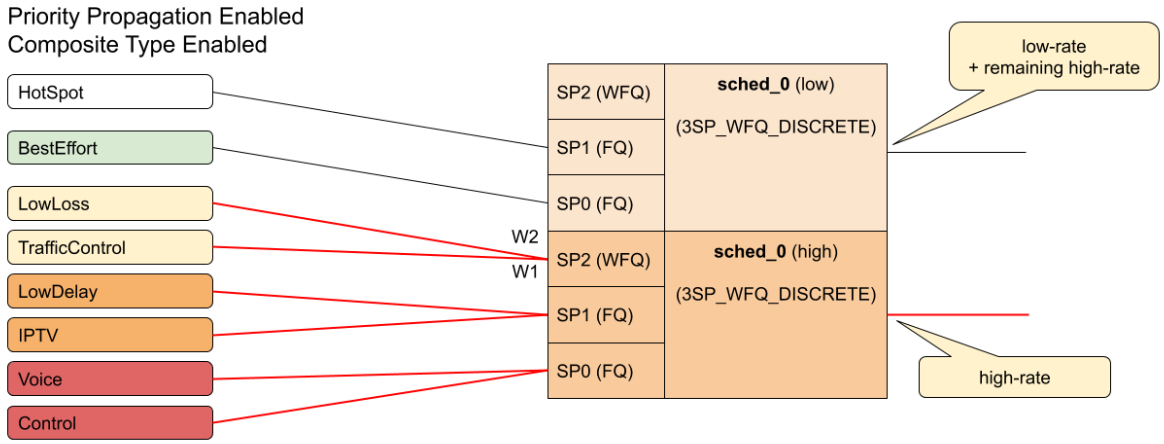


The credits generated from the physical interface will be consumed by high priority flow first and remaining credits will be available for low priority flow. The high flow traffic at any one element is scheduled based on type and connection point. Between schedulers it depends on how they are connected 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.

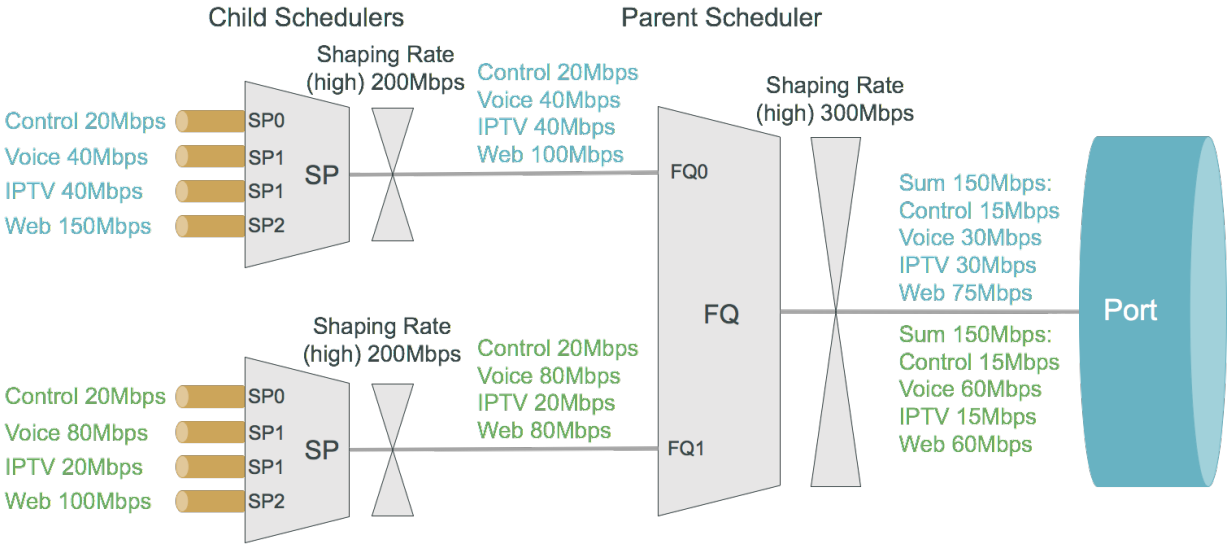
The following example shows a typical access service provider configuration with priority propagation enabled with and without composite type.





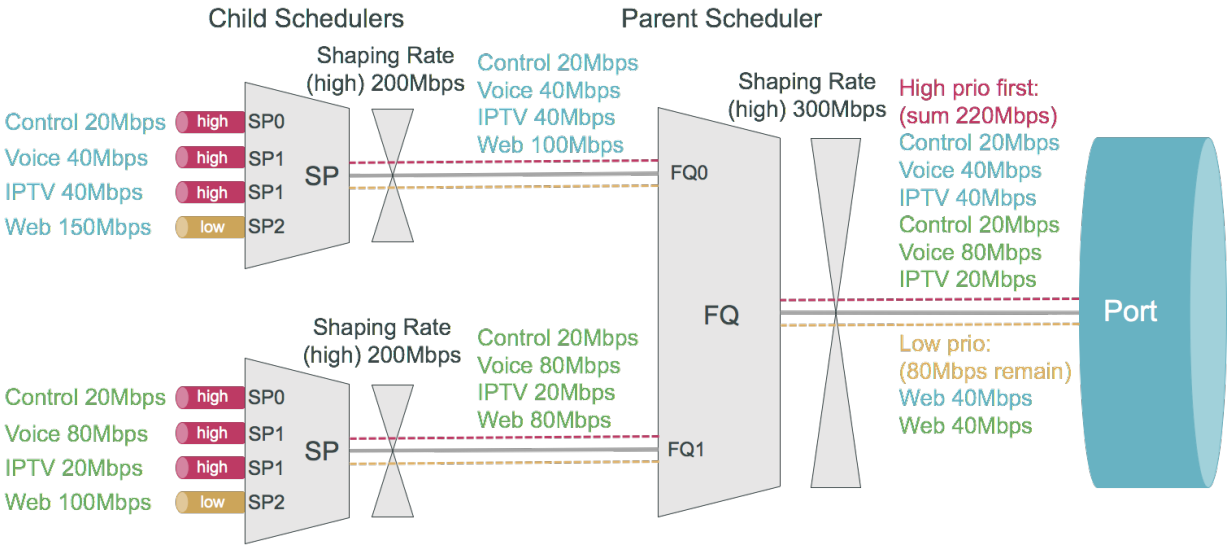
2.1.1. Simple Priority Propagation Scheduling Example

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

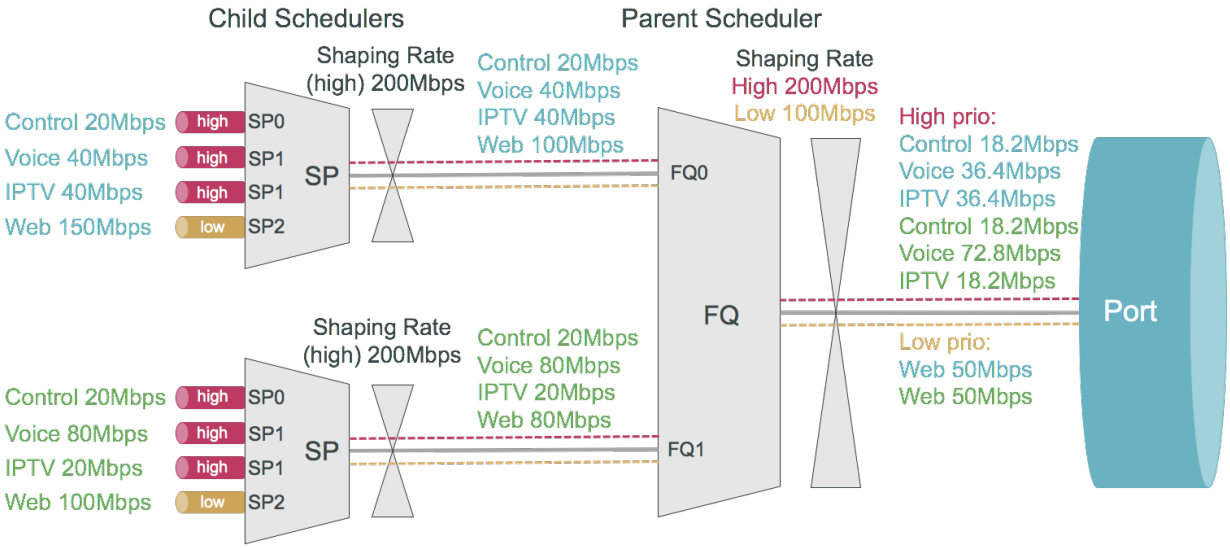


With priority propagation, the parent scheduler serves high priority flows first as

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.

- 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:

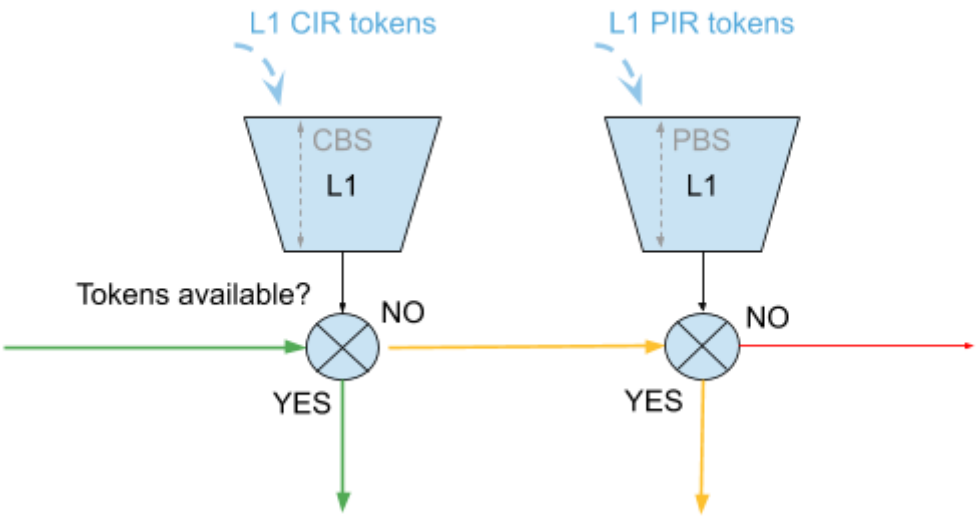
- For IPv4: Only ToS based classification is possible. DSCP based classification is not possible.
- For IPv6: Only TC based classification is possible. DSCP based classification is not possible.
- For EXP classification, the Uniform MPLS model is used to copy the MSB 3-bits from DSCP to EXP field at 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 *class-0* to *class-7*.

2.3. 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.

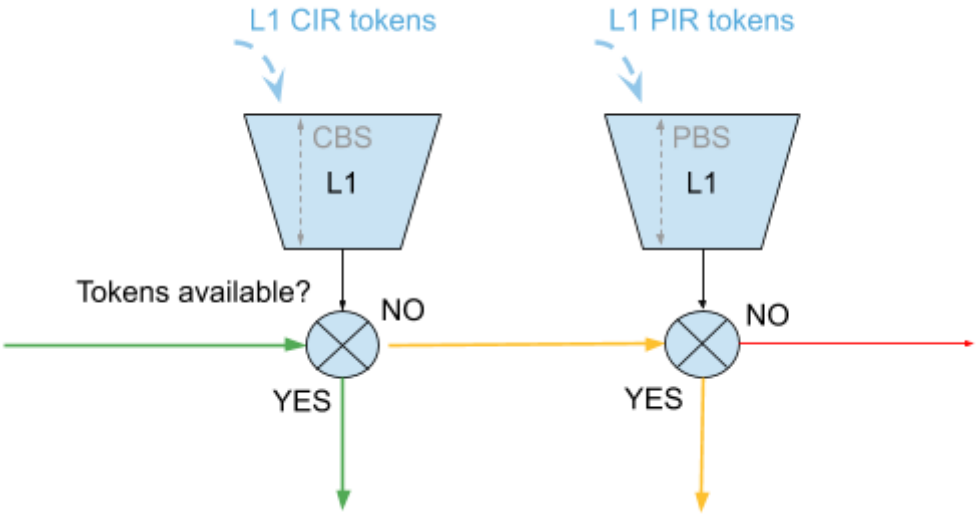
In RBFS, policers support "**two-rate, three-color**" type in a 4-levels cascaded mode. This means that each policer level has two rates (CIR and PIR) and three colors (green, yellow and red) with two token buckets as shown below.



This means that traffic below CIR is marked green. Traffic above CIR but below PIR is yellow and above PIR is red.

Traffic marked Yellow and Red is dropped.

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 us 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 (e.g. 3m in

this example). Obviously max CIR does not make sense 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.

In RBFS, policer level to class mapping is implicit:

- *class-1*: level-1
- *class-2*: level-2
- *class-3*: level-3
- *class-4*: level-4

2.4. 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.

2.5. Queue-Classifier

A queue-classifier defines the mapping of a class to a queue. Queue Classifier is attached to a Profile.

2.6. Queue-Group

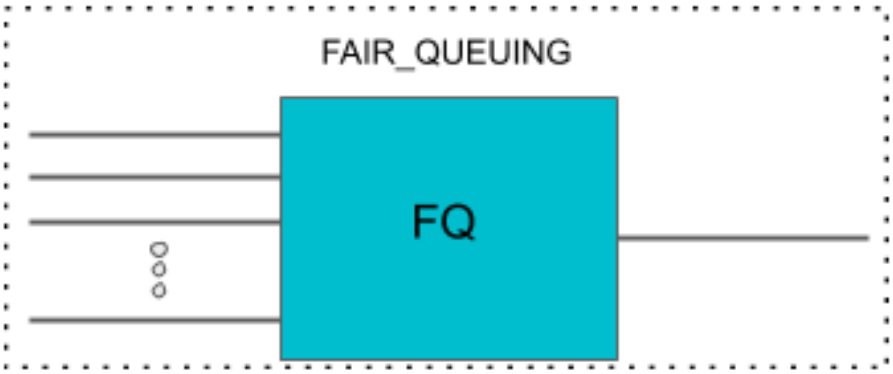
A Queue Group defines the Queue bundle. A Queue Group contains bundle of either 4 or 8 queues.

2.7. Scheduler

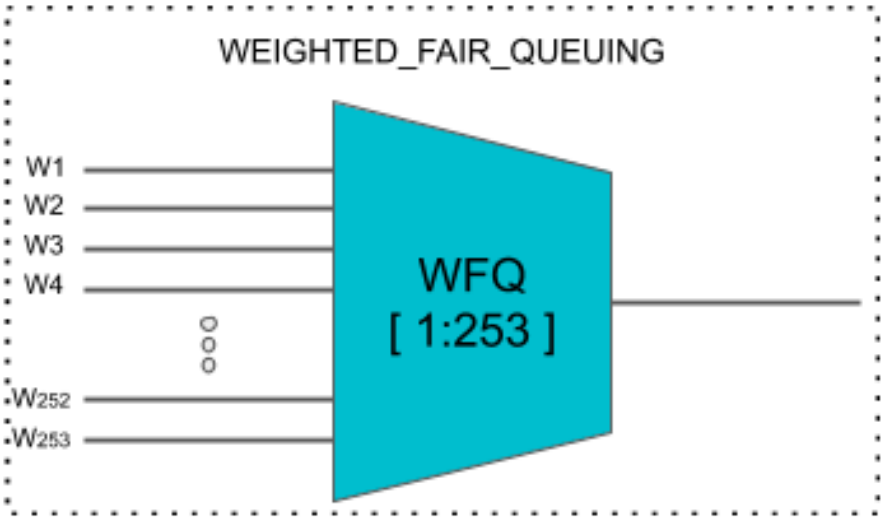
A scheduler configuration defines scheduler parameters such as type and shaping rate. The shaping rate defined for a scheduler applies to queue(s) associated with it.

The following scheduler types are supported:

- **FAIR_QUEUING (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_QUEUING* to create FQ scheduler.



- **WEIGHTED_FAIR_QUEUING(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_QUEUING* to create WFQ scheduler.
  - Supported weight: 1 to 253



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

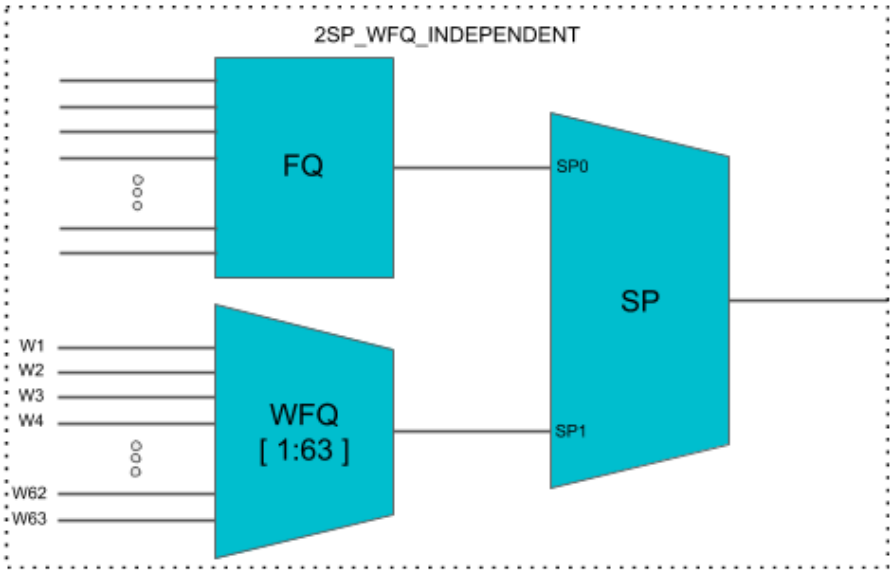
- **STRICT_PRIORITY(SP)**: Uses priority based approach to service the flow. SP schedulers are supported in “hybrid” mode only. Hybrid mode combines FQ-WFQ schedulers using strict priority.

The priority order for SP is: SP0 > SP1 > SP2 > SP3 (where SP0 being highest priority and SP3 being lowest).

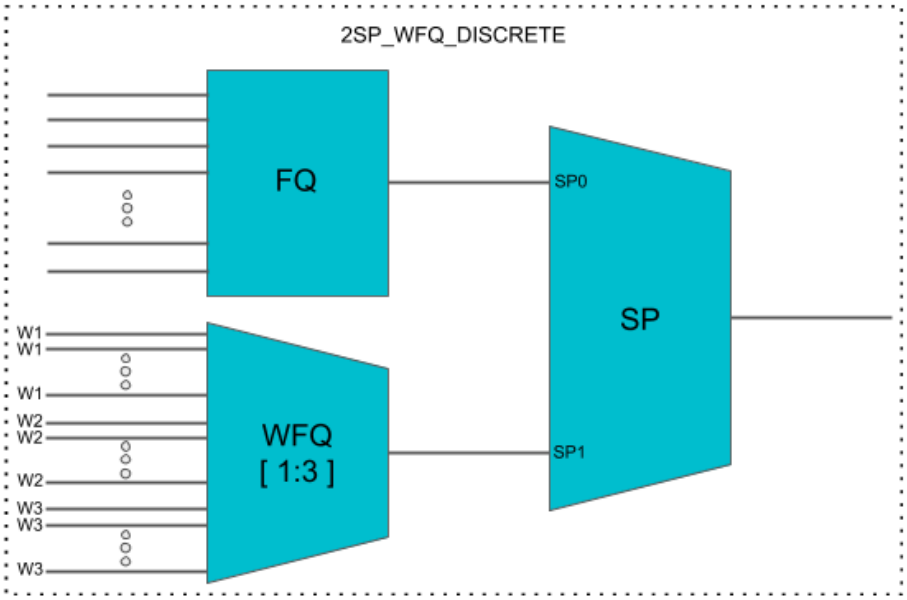
The following SP scheduler types are supported:

- **2SP**: Uses SP between 1-FQ and 1-WFQ. There are following types of 2SP hybrid schedulers:

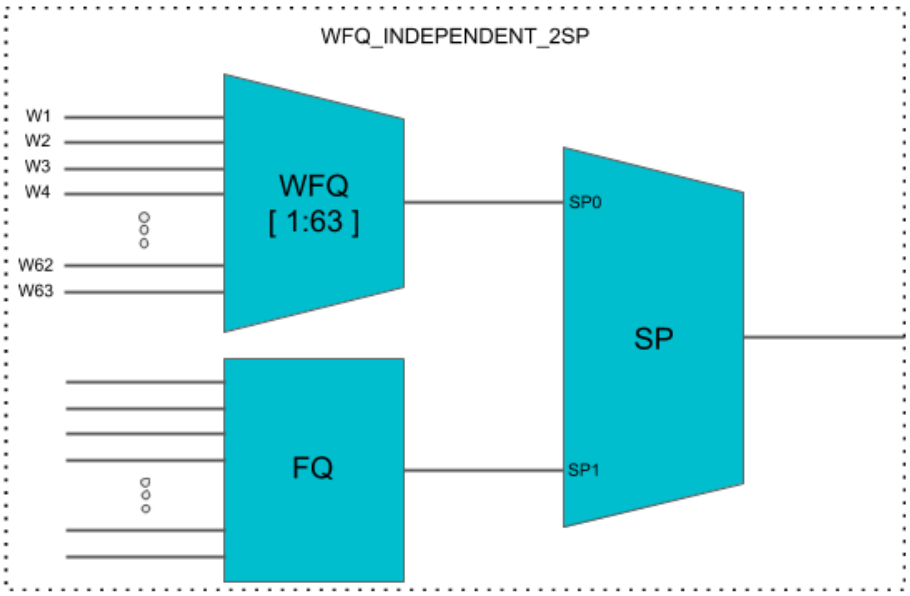
- type **“2SP_WFQ_INDEPENDENT”**
  - Supported weight: 1 to 63



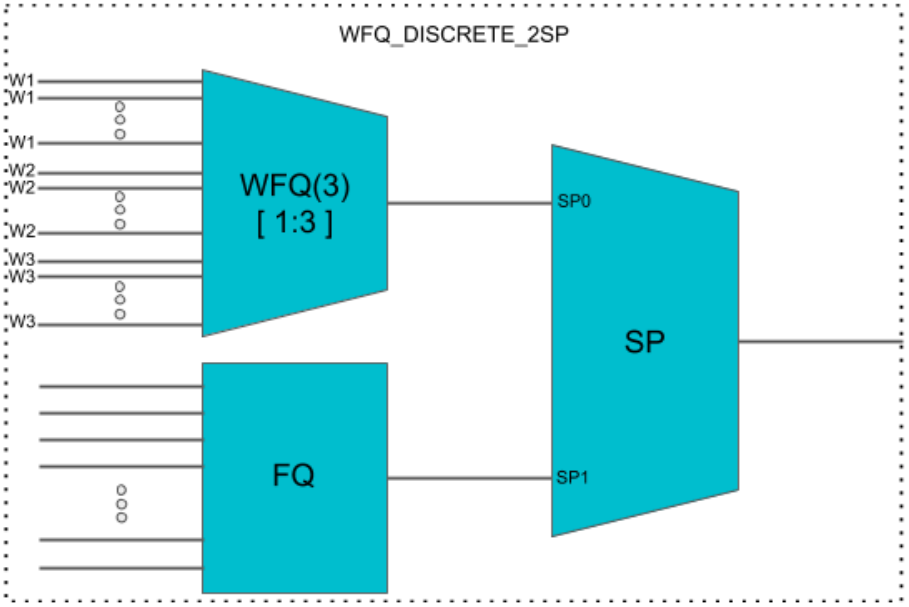
- type **“2SP_WFQ_DISCRETE”**
  - Supported weight: { 1, 2, 3 }



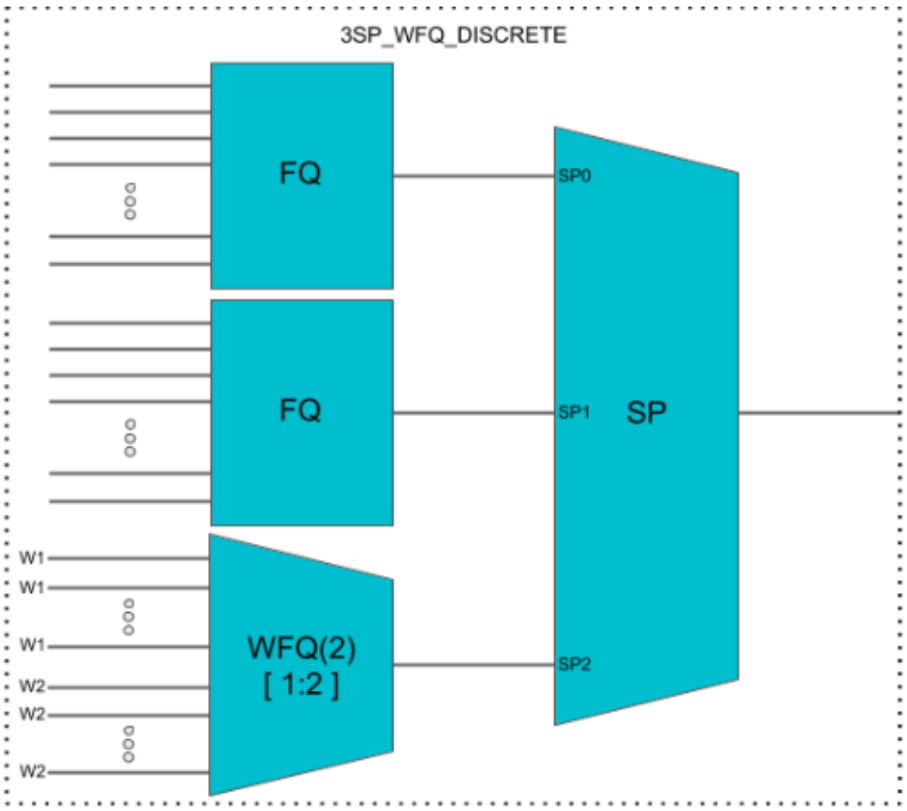
- type **“WFQ_INDEPENDENT_2SP”**
  - Supported weight: 1 to 63



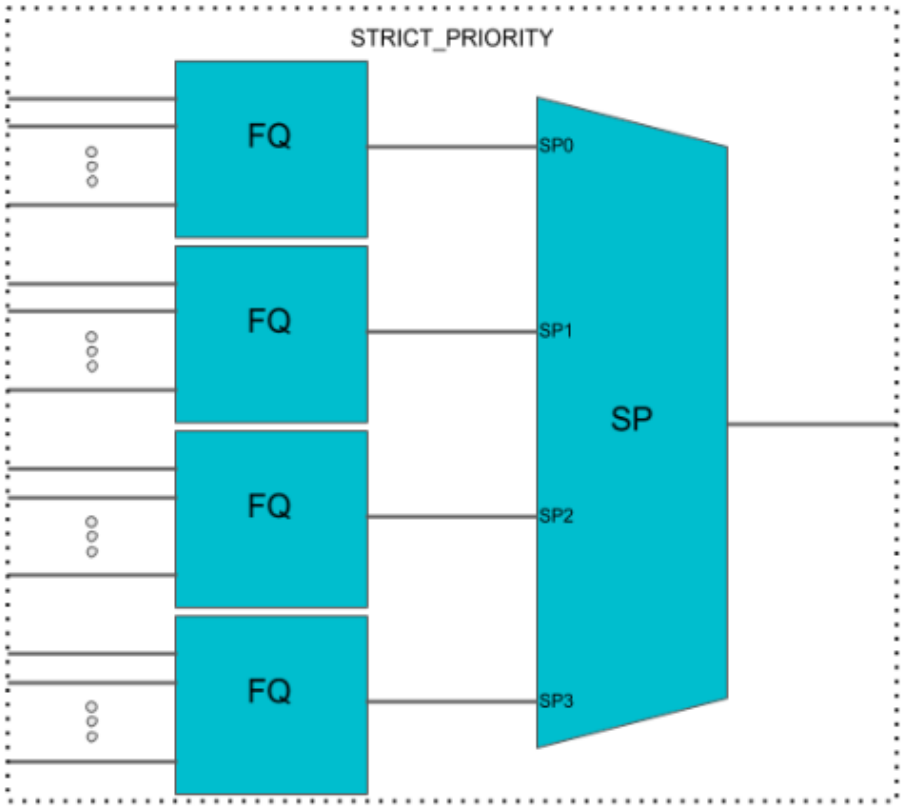
- type **“WFQ_DISCRETE_2SP”**
  - Supported weight: { 1, 2, 3 }



- **3SP**: maps 2-FQs and 1-WFQ
  - type: **“3SP_WFQ_DISCRETE“**
    - Supported weight: { 1, 2 }



- **4SP**: maps 4-FQs using SP
  - type **“STRICT_PRIORITY"**



2.8. 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).

2.9. 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.

2.10. 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:

- Scheduler-Map
- Queue Classifier
- Classifier
- Policer

2.11. 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.

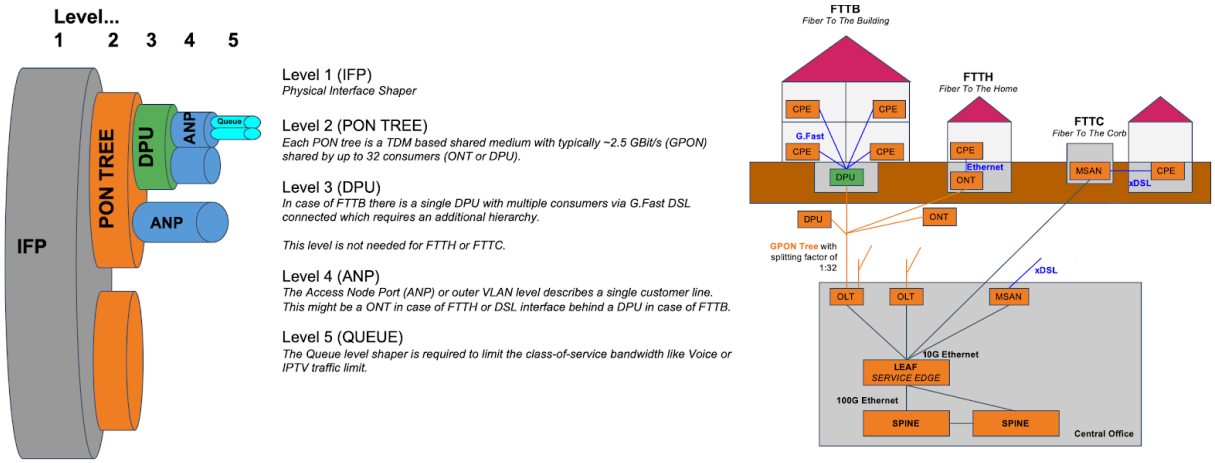
Level-4 (ANP or Session)

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.

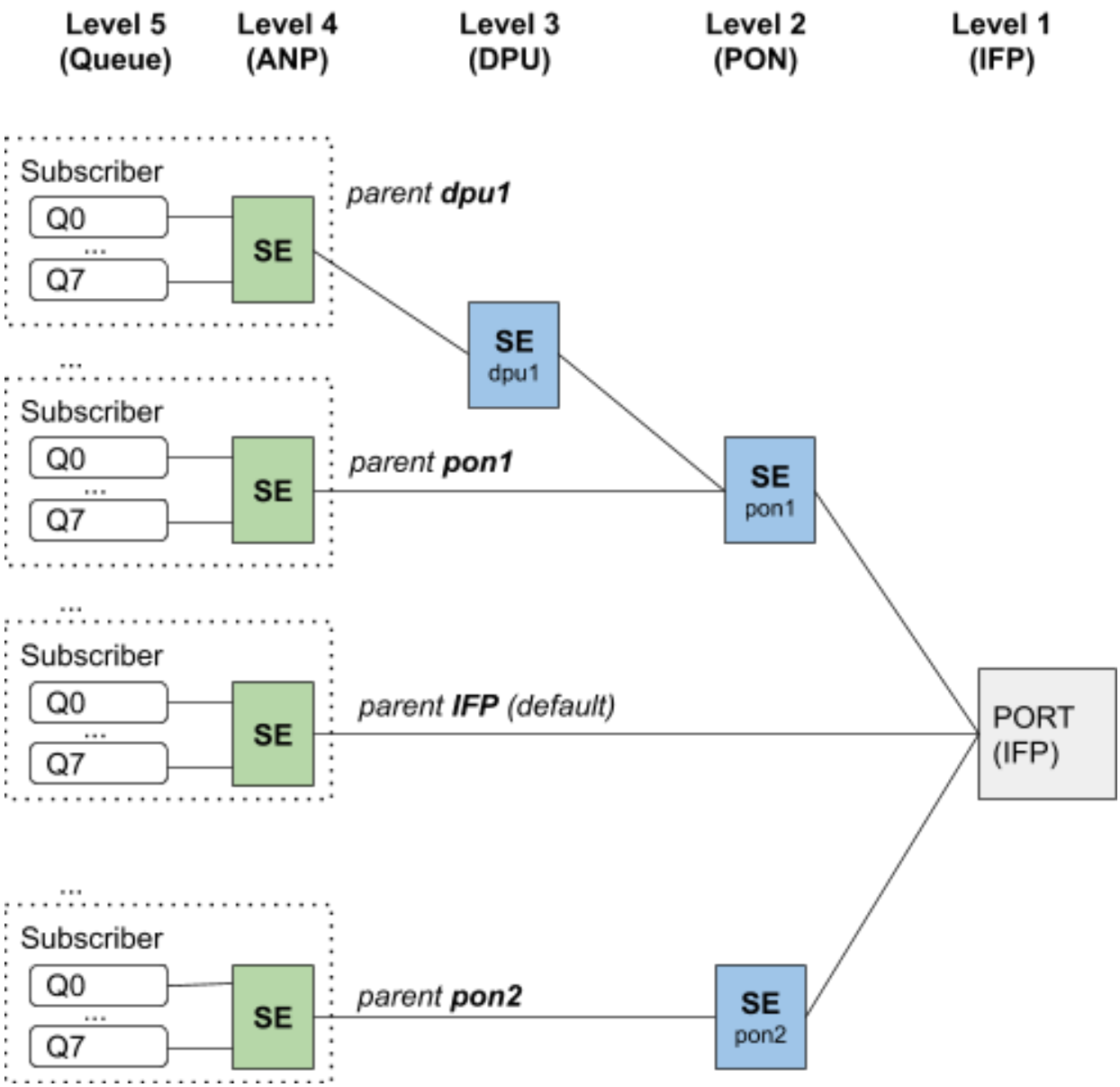


The levels 4 and 5 are configured per logical interface (i.e. subscriber-ifl or l3-ifl). Separate scheduler-map representing levels 1 to 3 connectivity shall be statically configured and mapped to corresponding physical interface (IFP).

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
Subscriber
Q0
...
Q7
SE
parent pon1
SE
pon1
Subscriber
Q0
...
Q7
SE
parent IFP (default)
PORT
(IFP)
Subscriber
Q0
...
Q7
SE
parent pon2
SE
pon2

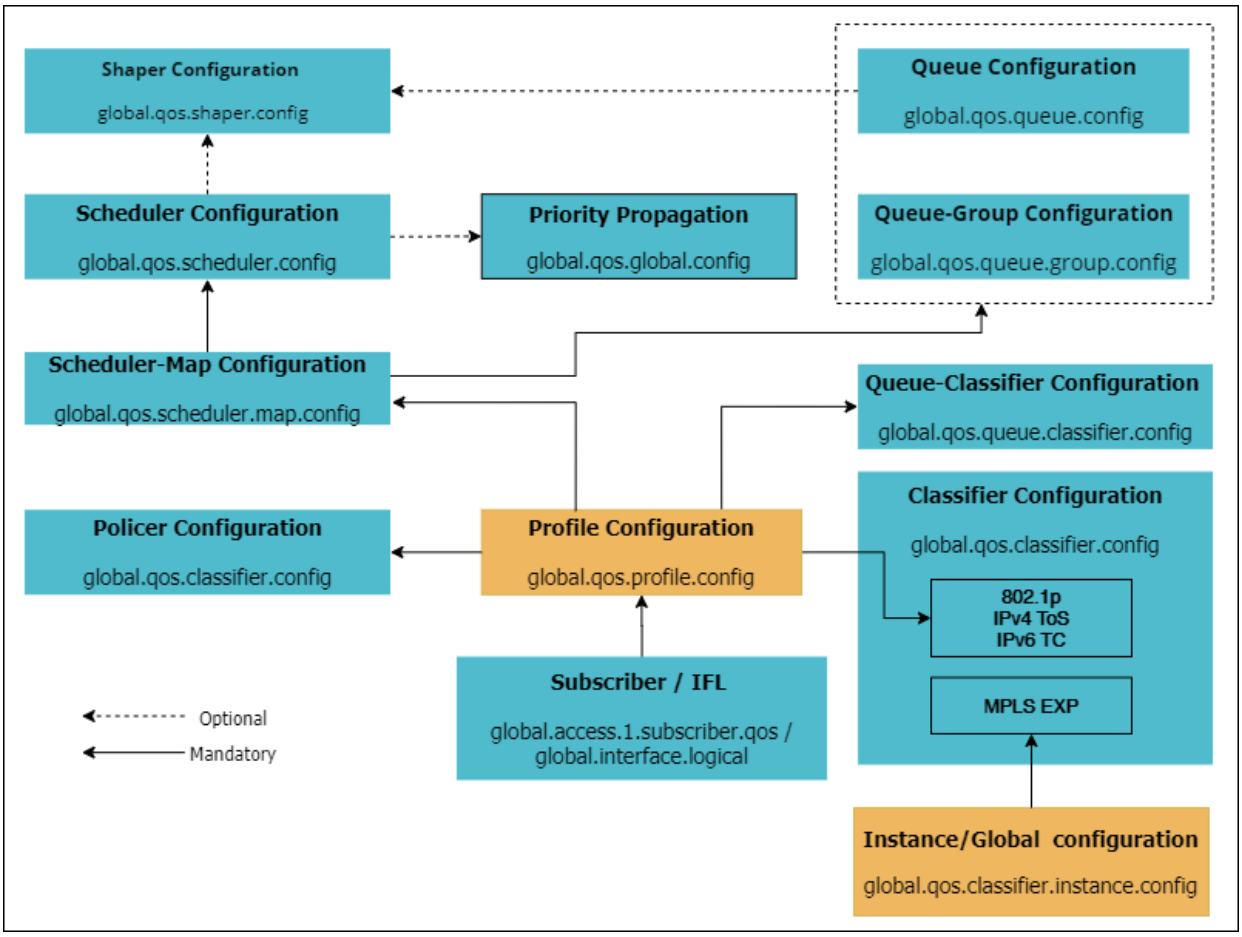
3. Configuring HQoS

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

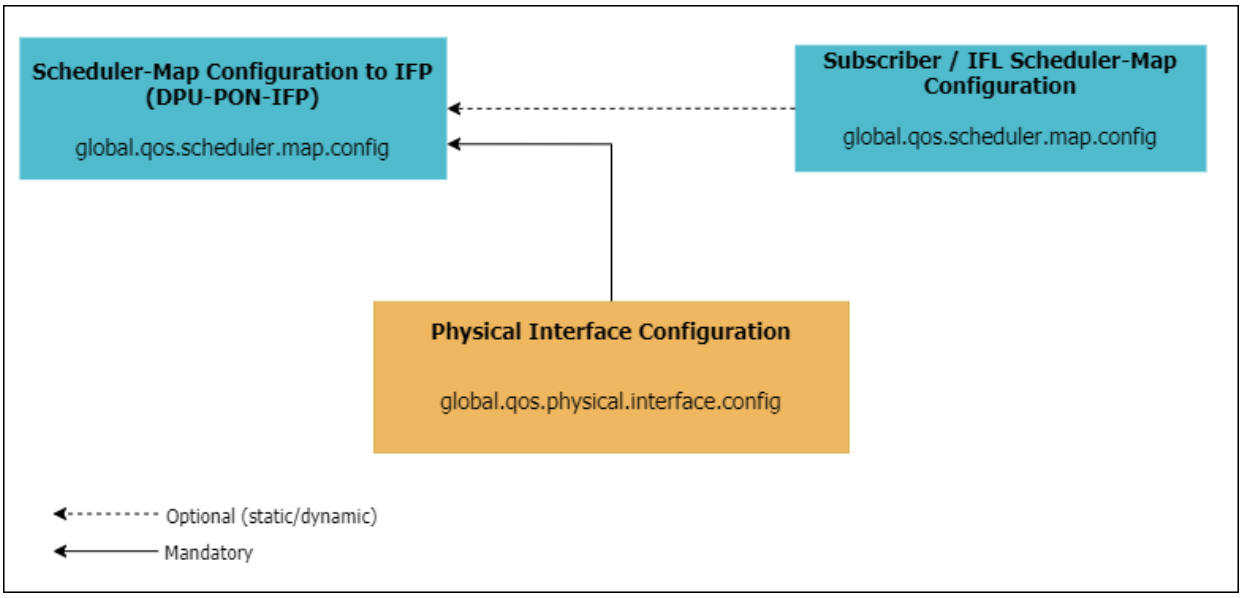
1. Create a Behavioral Aggregate (BA) classifier to classify the network traffic.
2. Create a policer to police the classified traffic at the ingress.
3. Create necessary queues with proper size to queue the classified traffic at the egress.
4. Create queue-groups and configure the queue numbers (4/8)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

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

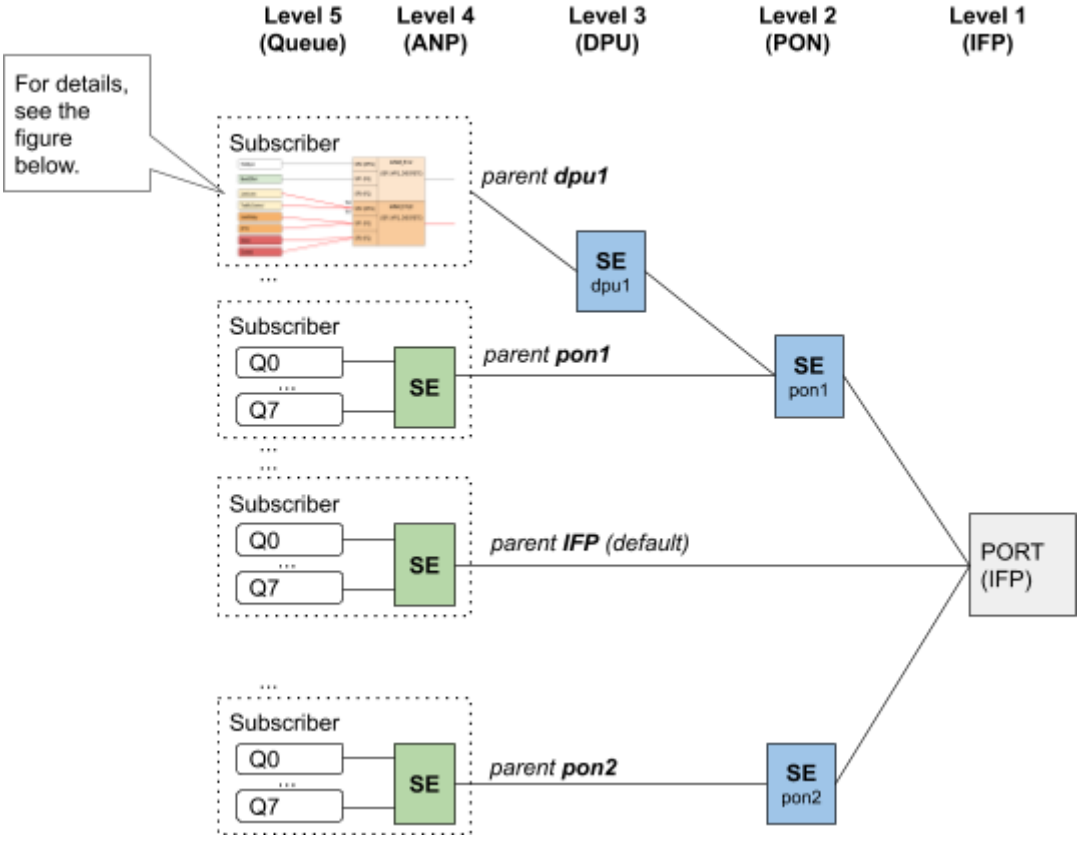
The figure below shows the dependencies between the various HQoS configuration elements.

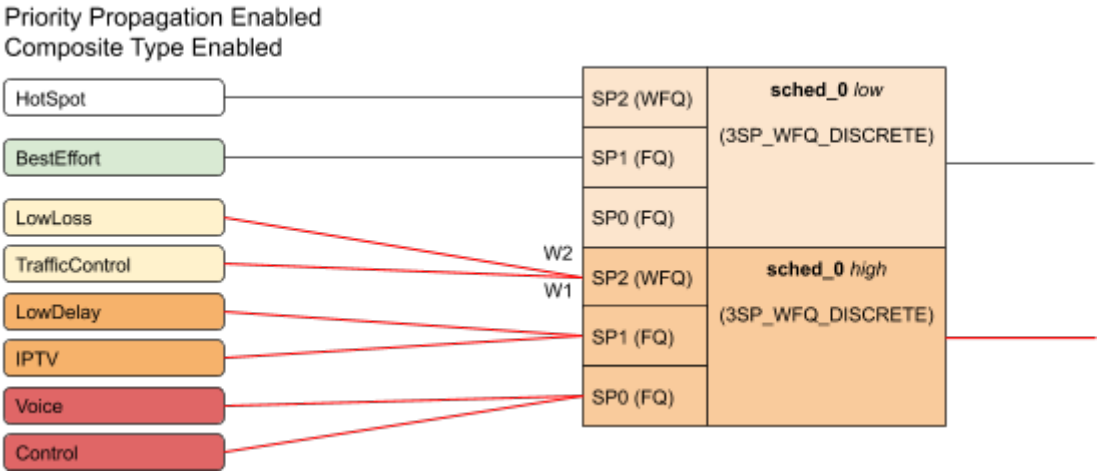


The figure below shows the additional dependencies for Multi-level HQoS.



The figures below show the scheduling hierarchy example.





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

- Classifier Configuration
- Policer Configuration
- Queue Configuration
- Queue-Classifier Configuration
- Queue-Group Configuration
- Scheduler Configuration
- Scheduler-Map Configuration

- Shaper Configuration
- Profiles Configuration
- Interface 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 name>

IPv4/v6 classifiers are applied on an interface—subscriber-ifl or l3ifl using the Profile Name.

Syntax

[forwarding-options class-of-service]
set profiles profile-name <QoS 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
<codepoint>	Specifies the code-point value based on match-type
<class name>	Specifies the traffic class as as class-0, class-1, class-2, class-3, class-4, class-5, class-6, and class-7

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. 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
<levels> Set levels
```

Level name configuration

[forwarding-options class-of-service policer policer-name voice_policer]
set level1-name <name>

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

Example

```
root@rtbrick:confd> set level1-na
<level1_name> Set level1 name
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set level1-name Platinum

root@rtbrick:confd> set level2-name Premium
[<Enter>] Execute the command
[ forwarding-options class-of-service policer policer-name voice_policer ]
root@rtbrick:confd> set level2-name Premium
```

Type configuration

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

Example

```
root@rtbrick:confd> set type Two-rate-three-color
[<Enter>] Execute the command
[ forwarding-options class-of-service policer policer-name voice_policer ]
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.
<level-name>	Specifies level name.
<type>	Specifies the policer type.
<flag>	Set flags.
<level1_cir>	Set committed information rate for level-1. The same is applicable for level-2 to level-4.
<level1_pir>	Set peak information rate for level-1. The same is applicable for level-2 to level-4.
<level1_cbs>	Set Committed burst size for level-1. The same is applicable for level-2 to level-4.
<level1_pbs>	Set peak burst size 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. The same is applicable for level-2 to level-4.
<level1_max_pir>	Set the maximum for the level-1 peak information rate. The same is applicable for level-2 to level-4.

3.3. 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 threshold on the queue
	<maximum-threshold>	Specifies the maximum threshold on the queue
	<drop_prob>	Drop Probability on the Queue

3.4. Queue-Classifier Configuration

Syntax

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

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

Class to Queue Mapping

set queue-name <queue_name>

Command arguments

<queue-classifier-name>	User-defined Queue-Classifier Name
<class>	class-0 to class-7
<queue-name>	User-define Queue-Name

Example

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

root@rtbrick:confd> edit queue-classifier-name queue-classifier-name class
class-1
[ forwarding-options class-of-service queue-classifier queue-classifier-name
queue-classifier-name class class-1 ]
[ forwarding-options class-of-service queue-classifier queue-classifier-name
queue-classifier-name class class-1 ]
root@rtbrick:confd> set queue-name Voice
```

3.5. 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
```

3.6. 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 rtbrick_sched_0
```

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
rtbrick_sched_0 ]
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 2SP_WFQ_INDEPENDENT 3SP_WFQ_DISCRETE FAIR_QUEUING STRICT_PRIORITY WEIGHTED_FAIR_QUEUING WFQ_DISCRETE_2SP 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_4
[ forwarding-options class-of-service scheduler scheduler-name sched_4 ]

[ forwarding-options class-of-service scheduler scheduler-name sched_4 ]
root@rtbrick:confd> set type 3SP_WFQ_DISCRETE

[ forwarding-options class-of-service scheduler scheduler-name sched_4 ]
root@rtbrick:confd> set shaper-name shaper_1
```

3.7. 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
root@rtbrick:confd> set parent-scheduler rtbrick_sched_0 parent-flow HIGH-
FLOW connection-point STRICT_PRIORITY_0
```

Configuring the parent-flow is optional.

3.7.1. Scheduler to Port Configuration

Syntax

set port-connection

Example

```
[ forwarding-options class-of-service scheduler-map scheduler-map-name
rtbrick_sched_map_residential queue-group-name queue_group_residential queue-
name IPTV ]
root@rtbrick:confd> set port-
port-connection Set port connection
port-flow Set port flow (High or Low)
root@rtbrick:confd> set port-connection
```

3.7.2. 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
```

3.7.3. Scheduler to Scheduler (different Scheduler-Map)

Syntax

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

3.7.4. 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
```

3.7.5. Queue to schedule

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
```

3.7.6. Scheduler to port

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
```

3.8. 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>
```

3.8.1. Shaper Rate

Syntax

set shaping-rate <shaping_rate>

High Flow Shaping Rate

To configure same the shaping value for high-flow and low-flow, enter the following command:

rtb confd set forwarding-options class-of-service shaper shaper-name <shaper_name> **shaping-rate** <shaping_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 different shaping values for high-flow and low-flow, enter the following command:

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

If priority propagation is not enabled, only high-flow value will be configured as shaping rate.

Example

```
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

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 ]
```

3.9. Priority Propagation

Syntax

rtb confd set forwarding-options class-of-service 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>
```

3.10. Profiles Configuration

Syntax

[forwarding-options class-of-service]
edit profiles profile-name <QoS Profile-Name>

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

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 ]
```

Policer

set policer-name <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 ]
```

Queue Classifier Name

set queue-classifier-name <queue_classifier_name>

Example

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

Scheduler-Map

set scheduler-map-name <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>
```

Command arguments

<QoS Profile-Name>	User-defined QoS Profile name
<classifier_name>	User-defined Classifier name
<policer_name>	User-defined Policer name
<queue_classifier_name>	User-defined Classifier name
<scheduler_map_name>	User-defined Scheduler map 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 ]
root@rtbrick:confd> set classifier-name Classifier_IP
root@rtbrick:confd> set policer-name Policer_Upstream
root@rtbrick:confd> set queue-classifier-name classifier_residential
root@rtbrick:confd> set scheduler-map-name rtbrick_sched_map_residential
```

3.11. 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:ClassifierMap_IP IPv4-TOS 0": {
                  "class": "class-1"
                },
                "classifier-name:ClassifierMap_IP IPv4-TOS 32": {
                  "class": "class-0"
                },
                "classifier-name:ClassifierMap_IP IPv4-TOS 64": {
                  "class": "class-2"
                },
                "classifier-name:ClassifierMap_IP IPv4-TOS 96": {
                  "class": "class-3"
                },
                "classifier-name:ClassifierMap_IP IPv4-TOS 128": {
                  "class": "class-4"
                },
                "classifier-name:ClassifierMap_IP IPv4-TOS 160": {
                  "class": "class-5"
                },
                "classifier-name:ClassifierMap_IP IPv4-TOS 192": {
                  "class": "class-6"
                },
                "classifier-name:ClassifierMap_IP IPv4-TOS 224": {
                  "class": "class-7"
                },
                "classifier-name:ClassifierMap_IP IPv6-TC 0": {
                  "class": "class-1"
                },
                "classifier-name:ClassifierMap_IP IPv6-TC 32": {
                  "class": "class-0"
                },
                "classifier-name:ClassifierMap_IP IPv6-TC 64": {
                  "class": "class-2"
                },
                "classifier-name:ClassifierMap_IP IPv6-TC 96": {
                  "class": "class-3"
                },
                "classifier-name:ClassifierMap_IP IPv6-TC 128": {
```

```
            "class": "class-4"
          },
          "classifier-name:ClassifierMap_IP IPv6-TC 160": {
            "class": "class-5"
          },
          "classifier-name:ClassifierMap_IP IPv6-TC 192": {
            "class": "class-6"
          },
          "classifier-name:ClassifierMap_IP IPv6-TC 224": {
            "class": "class-7"
          },
          "classifier-name:ClassifierMap_EXP EXP 0": {
            "class": "class-0"
          },
          "classifier-name:ClassifierMap_EXP EXP 1": {
            "class": "class-1"
          },
          "classifier-name:ClassifierMap_EXP EXP 2": {
            "class": "class-2"
          },
          "classifier-name:ClassifierMap_EXP EXP 3": {
            "class": "class-3"
          },
          "classifier-name:ClassifierMap_EXP EXP 4": {
            "class": "class-4"
          },
          "classifier-name:ClassifierMap_EXP EXP 5": {
            "class": "class-5"
          },
          "classifier-name:ClassifierMap_EXP EXP 6": {
            "class": "class-6"
          },
          "classifier-name:ClassifierMap_EXP EXP 7": {
            "class": "class-7"
          }
        }
      ],
      "policer": [
        {
          "policer_name:Policer_Upstream": {
            "type": "Two-rate-three-color",
            "flags": "Color-Blind",
            "levels": 4,
            "level1_name": "premium",
            "level1_cir": 9000,
            "level1_pir": 1500,
            "level1_cbs": 700,
            "level1_pbs": 750,
            "level2_name": "aggregate",
            "level2_cir": 2000,
            "level2_pir": 2500,
            "level2_cbs": 800,
            "level2_pbs": 850,
            "level3_name": "gold",
            "level3_cir": 3000,
            "level3_pir": 3500,
            "level3_cbs": 700,
            "level3_pbs": 750,
```

```
            "level4_name": "platinum",
            "level4_cir": 4000,
            "level4_pir": 4500,
            "level4_cbs": 800,
            "level4_pbs": 850
          }
        }
      ],
      "profiles": [
        {
          "profile_name:rtbrick_residential_profile": {
            "classifier_name": "ClassifierMap_IP",
            "policer_name": "Policer_Upstream",
            "queue_classifier_name": "classifier_residential",
            "scheduler_map_name": "rtbrick_sched_map_residential"
          }
        }
      ],
      "queue": [
        {
          "qos_queue:BestEffort": {
            "minimum_threshold": 5000,
            "maximum_threshold": 6000,
            "drop_probability": 50,
            "queue_size": 12500
          },
          "qos_queue:Control": {
            "queue_size": 12500,
            "shaper_name": "shaper_Control"
          },
          "qos_queue:HotSpot": {
            "queue_size": 12500
          },
          "qos_queue:IPTV": {
            "queue_size": 6250
          },
          "qos_queue:LowDelay": {
            "queue_size": 12500
          },
          "qos_queue:LowLoss": {
            "queue_size": 12500
          },
          "qos_queue:TrafficControl": {
            "queue_size": 12500
          },
          "qos_queue:Voice": {
            "minimum_threshold": 5000,
            "maximum_threshold": 6000,
            "drop_probability": 50,
            "queue_size": 12500,
            "shaper_name": "shaper_Voice"
          }
        }
      ],
      "shaper": [
        {
          "shaper-name:BE_shaper": {
            "shaper_rate_high": 4000,
```

```
            "shaper_rate_low": 5000
          },
          "shaper-name:session_shaper": {
            "shaper_rate_high": 10000,
            "shaper_rate_low": 5000
          },
          "shaper-name:shaper_Control": {
            "shaper_rate_high": 5500,
            "shaper_rate_low": 4000
          },
          "shaper-name:voice_shaper": {
            "shaper_rate_high": 6500,
            "shaper_rate_low": 4000
          }
        }
      ],
      "queue-classifier": [
        {
          "qos_queue_classfier:classifier_residential class-5": {
            "queue_name": "IPTV"
          },
          "qos_queue_classfier:classifier_residential class-6": {
            "queue_name": "Voice"
          },
          "qos_queue_classfier:classifier_residential class-7": {
            "queue_name": "Control"
          },
          "qos_queue_classfier:classifier_residential class-1": {
            "queue_name": "HotSpot"
          },
          "qos_queue_classfier:classifier_residential class-2": {
            "queue_name": "LowLoss"
          },
          "qos_queue_classfier:classifier_residential class-4": {
            "queue_name": "LowDelay"
          },
          "qos_queue_classfier:classifier_residential class-0": {
            "queue_name": "BestEffort"
          },
          "qos_queue_classfier:classifier_residential class-3": {
            "queue_name": "TrafficControl"
          }
        }
      ],
      "queue-group": [
        {
          "queue-group-name:queue_group_residential": {
            "queue_numbers": 8
          }
        }
      ],
      "scheduler": [
        {
          "scheduler_name:dpu_sched_1": {
            "type": "FAIR_QUEUING"
          },
          "scheduler_name:pon_sched_1": {
            "type": "FAIR_QUEUING"
```

```
                },
                "scheduler_name:rtbrick_sched_0": {
                  "shaper_name": "session_shaper",
                  "type": "3SP_WFQ_DISCRETE"
                }
              }
            ],
            "scheduler-map": [
              {
                "scheduler_map:dpu1 dpu_sched_1": {
                  "parent_scheduler_name": "pon_sched_1",
                  "connection_point": "NO_PRIORITY"
                },
                "scheduler_map:dpu1 pon_sched_1": {
                },
                "scheduler_map:rtbrick_sched_map_residential
rtbrick_sched_0": {
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential IPTV": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "parent_scheduler_flow": "HIGH-FLOW",
                  "connection_point": "STRICT_PRIORITY_1"
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential Voice": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "parent_scheduler_flow": "HIGH-FLOW",
                  "connection_point": "STRICT_PRIORITY_0"
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential Control": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "parent_scheduler_flow": "HIGH-FLOW",
                  "connection_point": "STRICT_PRIORITY_0"
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential HotSpot": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "parent_scheduler_flow": "LOW-FLOW",
                  "connection_point": "STRICT_PRIORITY_1"
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential LowLoss": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "parent_scheduler_flow": "HIGH-FLOW",
                  "connection_point": "STRICT_PRIORITY_2",
                  "weight": 1
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential LowDelay": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "parent_scheduler_flow": "HIGH-FLOW",
                  "connection_point": "STRICT_PRIORITY_1"
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential BestEffort": {
                  "parent_scheduler_name": "rtbrick_sched_0",
```

```
                  "parent_scheduler_flow": "LOW-FLOW",
                  "connection_point": "STRICT_PRIORITY_0"
                },
                "scheduler_map:rtbrick_sched_map_residential
queue_group_residential TrafficControl": {
                  "parent_scheduler_name": "rtbrick_sched_0",
                  "parent_scheduler_flow": "HIGH-FLOW",
                  "connection_point": "STRICT_PRIORITY_2",
                  "weight": 2
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

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$
```