BCM963XX

CPE Linux Ingress QoS

Application Note

Table of Contents

Chapter 1: Overview

The Ingress QoS feature provides a central control unit for managing different Ingress QoS needs, such as:

- Providing a mechanism to allow high priority traffic to be prioritized over lower priority traffic when there is CPU and/or packet processing resource congestion.
- Interacting with a hardware network processor to place packets in high priority receive queue, trap the packets in early packet classification stage, or drop the packets without informing the host processor.

NOTE: Within the scope of this document, “Congestion” means “CPU and/or packet processing resource congestion.” Both terms are used interchangeably.

This document is organized into the following sections:

- “Enable and Disable Ingress QoS” explains how to enable or disable the Ingress QoS feature.
- “Ingress QoS” explains the Ingress QoS features in detail.
- “CLI Commands” describes the Ingress QoS CLI commands.
- “APIs” describes various Ingress QoS APIs.
- “Performance” describes the impact of the Ingress QoS feature on low, high, or a combination of low and high priority traffic.

1.1 Purpose and Audience

The Ingress QoS feature allows high priority traffic to be prioritized when there is CPU congestion or resource limitations. This document describes the Ingress QoS feature from a user perspective, and is aimed at engineers using the BCM963XX reference design boards or designing with BCM63XX chips.

1.2 Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use. Acronyms and abbreviations in this document are shown in Table 1.

Table 1: Abbreviations

Abbreviation	*Definition*
CPE	Customer Premises Equipment
IP	Internet Protocol version 4
IQ	Ingress QoS
L4	Layer 4 protocol (TCP/UDP)
QoS	Quality of Service
RXBD	RX buffer descriptor
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Chapter 2: Enable and Disable Ingress QoS

By default, the Ingress QoS feature is enabled in all of the profiles.

NOTE: The Ingress QoS feature can be either statically built (*) with the Linux kernel or built as module (M) or be compiled out.

2.1 Enable Ingress QoS

To enable the Ingress QoS feature:

1. Use the `make menuconfig` command at the Linux command prompt before build.
   ```
   $ make menuconfig
   ```
2. From **menuconfig**, select **Datapath,** then **Buffer Pool Manager and Ingress QoS**, and then **Ingress QoS**.
3. Use the space bar to select the Ingress QoS feature.

2.2 Disable Ingress QoS

To disable the Ingress QoS feature:

1. Use the `make menuconfig` command at the Linux command prompt before build.
   ```
   $ make menuconfig
   ```
2. From **menuconfig**, select **Datapath,** then **Buffer Pool Manager and Ingress QoS**, and then **Ingress QoS**.
3. Use the space bar to deselect the Ingress QoS feature.

Chapter 3: Ingress QoS

The Ingress QoS feature protects high-priority traffic under CPU congestion or resource limitations by causing an early drop of low-priority traffic.

CPU and/or other packet processing resource congestion determination is platform dependent and kept outside the scope of this document.

Notes:

- Protection of high-priority packets means processing of all high-priority packets without drop subject to egress bandwidth limitations
- Packet priority (low or high) refers strictly to Ingress QoS packet priority and does not refer to any other priority like IEEE 802.1p, IPv4 ToS, etc. However, one can configure Ingress QoS based on any of these other priorities.
- Ingress QoS refers to QoS at the ingress/RX interface. This is not an end-to-end QoS (from RX to TX interface) feature. For an end-to-end QoS, Ingress QoS should be used along with the egress QoS feature to prioritize the flows.
- The maximum high-priority packet rate that is protected under congestion varies based on the platform and the maximum low-priority packet rate.

CAUTION! In a tunneled packet scenario with inner header parsing capability supported, such as IPv4-in-IPv6, IPv6-in-IPv4, or GRE, Ingress QoS for Layer 3 and Layer 4 fields works on inner header information rather than outer header information.

3.1 CPE Behavior — No Congestion State

Under normal conditions, the CPE behaves the same as when Ingress QoS is disabled except that a few cycles are spent finding the current congestion state and packets' Ingress QoS priority.

3.2 CPE Behavior — Congestion State

When the CPE is under congestion state, it either tries to free up processing cycles by dropping the low priority packets early during the RX processing stage, or the low priority packet processing is delayed to provide processing cycles to high priority traffic, in other words, by putting the Hi/Lo priority packets in different CPU queues.

Initially, when the CPE just enters the CPU congestion state, it will forward both high- and low-priority packets with some of the low priority packets dropped by Ingress QoS. Once the congestion state is reached as the congestion increases the low priority packet drop rate, but the high priority traffic is still protected. If the input packet rate keeps increasing there comes a point where all the low priority packets are dropped and if you go a little further, even the high priority traffic starts getting dropped.

3.3 Ingress QoS Packet Priority

Each received packet is assigned a low or high priority. A packet is assigned a high priority if meets one or more of the following criteria, otherwise it is assigned a low priority. This configuration is also relayed to the hardware network processor. If supported, the network processor will place packet in the proper receive queue based on the configured parameter when an Ingress QoS key hit occurs. (Criteria may vary per platform.)

- All multicast packets.
- Certain access level control packets, such as ARP, PPPOE Discovery and Session, and 802.1AG.
- SIP, RTSP, MGCP control packets (configured by VOIP).
- All of the RTP/RTCP connections established by SIP, RTSP, MGCP ALGs are also assigned high priority by adding the destination L4 (TCP/UDP) ports to the Ingress QoS table.
- Any entry added as high priority using Ingress QoS APIs.
- The ports that are added by default at initialization time or when an ALG is loaded are shown in Table 2.

Table 2: Default Entries Based on TCP/UDP + L4 Destination Ports

UDP/TCP	Destination Port	Comment
TCP	80, 8080	HTTP
	53	DNS
UDP	53	DNS
	67, 68, 546, 547	DHCP
	2427, 2727	MGCP

NOTE: By default all the L4 packets are assigned a low priority unless it has been assigned a high priority by adding the L4 destination port to the Ingress QoS table.

CAUTION! In routing mode firewall should be enabled to load the ALGs (SIP, RTSP, etc.).

CAUTION! In bridge mode the ALGs (SIP, RTSP, etc.) are not loaded therefore the voice and video connections will be treated as low priority by default. To treat them as high priority, the voice and video applications should add the L4 destination ports to the ingress QoS port table using the Ingress QoS CLI or APIs.

3.4 Force Trap

If a network processor is capable of supporting trapping certain types of packets early, then the configuration will be relayed to the network processor. When a packet is classified as *force trap*, it is classified as high priority into the CPU received queue.

NOTE: This feature is only supported on Runner-based platforms. The types of traffic that are supported are:

- ARP
- PPPOE Discovery
- PPPOE session
- PTP 1588
- DHCP
- 802.1ag

3.5 Drop

Drop can be forced with a key match. The drop can also be relayed to the network processor. In the case of Runner-based devices, if the drop can be done in the classification stage, it will be done early, or else the drop happens before putting the packet into the receive queue.

Chapter 4: CLI Commands

To see the list of Ingress QoS CLI commands just type **iq** at the shell prompt. (Commands vary per platform. The following display is a combination of all platforms. If a certain command is platform-specific, it will be mentioned in the later detailed section). Endianess for all the values should be in host order.

NOTE: "`iq`" is an alias for the "`iqctl`" CLI command.

```
# iq
Ingress QoS Control Utility:

::: Usage:

:::::: Ingress QoS SW System :
       iq status [-u]
          -u: new userspace status dump.
       iq enable
       iq disable
       iq flush
       iq addport
          --proto <0|1> --dport <1..65534> --ent <0|1> --prio <0|1>
          proto: 0 = TCP, 1 = UDP
          ent: 0 = dynamic, 1 = static
          prio: 0 = low, 1 = high
       iq remport --proto <0|1> --dport <1..65534> --ent <0|1>
          proto: 0 = TCP, 1 = UDP
          ent: 0 = dynamic, 1 = static
       iq getport --proto <0|1> --dport <1..65534>
          proto: 0 = TCP, 1 = UDP
       iq porttbl [-k]
          -k: dump from kernel
       iq addkeymask [field] --maskprio <0..15>
       iq remkeymask [field]
       iq keymasktbl
       iq addkey [field | action | attribute]
       iq remkey [field | action]
       iq getkey [field]
       iq keytbl
          field:
              --srcmac: MAC address in x:x:x:x:x:x format
              --dstmac: MAC address in x:x:x:x:x:x format
              --ethtype: ether type
              --outervid: Outer VLAN ID
              --outerpbit: Outer VLAN PBit
              --innervid: Inner VLAN ID
              --innerpbit: Inner VLAN PBit
              --l3proto: e.g., IPv4 (0x0800) or IPv6 (0x86DD)
              --ipproto: such as TCP (6) or UDP (17)
              --srcip: Source IPv4/6 Address]
              --dstip: Destination IPv4/6 Address
              --dscp: 16-bit IP->DSCP value
              --ipv6flowlabel: 16-bit IPv6->flowlabel value
              --srcport: 16-bit L4 SRC Port
              --dstport: 16-bit L4 DST Port
          action:
              --prio: 0 = low, 1 = high
              --drop: drop packet
              --trap: trap packet before flow lookup
          attribute:
              --ent: 0 = dynamic, 1 = static
              --maskprio: <0..15> priority value, larger value = higher priority
       iq setdefaultprio --prototype <0> --protoval <0..255> --prio <0/1>
          prototype: 0 = ipproto
          protoval: protocol value (0 to 255)
          prio: 0 = low, 1 = high
       iq remdefaultprio --prototype <0> --protoval <0..255>
          prototype: 0 = ipproto
          protoval: protocol value (0 to 255)
```

4.1 Status

Description This command displays the current status of Ingress QoS feature and related information. (Default is printing from kernel. [-u] option will make it print from userspace, but with limited information.)The fields described in Table 3 below are per interface for FAP-based platform.

Syntax (FAP)

```
#iq status
[NTC iq] iq_get_status: Ingress QoS status : enabled


----------------------IQ Status----------------------
     dev chnl loThr hiThr  used    dropped     cong
--- ---- ---- ----- ----- ----- ---------- --------
    ENET    0   396   450     0          0        0
    ENET    1   528   600     0          0        0
    XTM     0    84    96     0          0        0
    XTM     1    33    37     0          0        0
FAP ENET    0   396   450     0          0        0
FAP XTM     0   132   150     0          0        0
FAP XTM     1    10    12     0          0        0
```

Table 3: Status Fields for FAP-based (Per Interface)

Field	Description
FAP	Indicates if the device or channel is managed by FAP. If this field is blank it means the device or channel is managed by the host.
dev	RX interface or device.
chnl	A channel on the RX interface also referred as queue interchangeably.
loThr	RX queue low threshold. When the queue depth becomes less than loThr, CPU congestion is removed for this queue.
hiThr	RX queue high threshold. When the queue depth becomes more than hiThr, CPU congestion is declared for this queue.
used	This field shows how many RXBDs for the channel are used at that instant. After the traffic stops, this field should be 0.
dropped	Number of packets dropped by Ingress QoS for the RX queue because of the CPU congestion. The reasons for higher packet drop may be either the RX queue (ring) size is small, or the input packet rate is higher than CPE can handle.
cong	This field shows if any of the RX queues is experiencing congestion. This field displays separate congestion status for host and FAP queues in hex format. The same value is displayed for all the host entries, and similarly same FAP congestion status for all the FAP entries. The `cong` field is interpreted as given below. Bits [31:10] = For future use. Bits [9:8] = One bit for each CMF FWD RX channel. Bit-8 for channel-0, bit-9 for channel-1, … Bits [7:4] = One bit for each XTM RX channel. Bit-4 for channel-0, bit-5 for channel-1, … Bits [3:0] = One bit for each Ethernet RX channel. Bit-0 for channel-0, bit-1 for channel-1, …

Syntax (Runner)

```
Runner-based has up to 16 CPU RX queues, the output is based on QID.
# iq status
[NTC iq] iq_get_status: Ingress QoS status : enabled

----------------IQ Status------------------
 qidx   rxpkts  queued  dropped interrupts
-----------------------------------------
 0      0       0       0       0
 1      0       0       0       0
 2      0       0       0       0
 3      55      0       0       27
 4      6678    0       0       3481
 5      0       0       0       0
 6      0       0       0       0
 7      0       0       0       0
 8      0       0       0       0
 9      0       0       0       0
 10     0       0       0       0
 11     0       0       0       0
 12     0       0       0       0
 13     0       0       0       0
```

Syntax

```
# iq status -u
BCM Ingress QOS Status: enabled
```

4.2 Enable

Description This command enables the Ingress QoS feature.

Syntax

```
# iq enable
```

4.3 Disable

Description This command disables the Ingress QoS feature.

Syntax

```
# iq disable
```

4.4 Flush

Description This command flushes all the dynamic entries.

Syntax

```
# iq flush
```

4.5 Add Port

Description This command adds L4 destination port to the Ingress QoS port table. The fields are defined in Table 4

Syntax

```
# iq addport
        --proto <0|1> --dport <1..65534> --ent <0|1> --prio <0|1>
```

Table 4: Add Port Fields

Field	Description
proto	UDP or TCP: 0 = TCP 1 = UDP
dport	L4 destination port, whose packet priority needs to be added.
ent	Entry type 0: Dynamic entry 1: Static entry
prio	Ingress QoS priority assigned to all the packets received with `proto:dport` combination 0 = Low 1 = High

4.6 Remove Port

Description This command removes L4 destination port from the Ingress QoS port table.

Syntax

```
# iq remport
        --proto <0|1> --dport <1..65534> --ent <0|1>
```

4.7 Get Port

Description This command displays L4 destination port's entry type and packet priority from the Ingress. QoS port table. If a L4 destination port is not present in Ingress QoS port table, an entry `type=0` (dynamic) and `prio=0` (low) is returned.

Syntax

```
# iq getport
        --proto <0|1> --dport <1..65534>
```

4.8 Dump Key Mask and Key Table

Description This command dumps the key mask and key entries of the Ingress QoS table for the requested protocol. Option -k can be used to dump the same tables in a different format from kernel.

Syntax

```
# iq porttbl

BCM Ingress QOS
        Status: enabled
        Total Static Entry Count: 16
        Total Dynamic Entry Count: 0

 keymask table:
        index#0: prio = 14, refcnt = 4
                mask = --ethtype
        index#1: prio = 10, refcnt = 2
                mask = --ipproto --srcport
        index#2: prio = 9, refcnt = 0
                mask = --ipproto
        index#3: prio = 8, refcnt = 10
                mask = --ipproto --dstport

 key table:
        index#60: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x8864
                action = prio = 1
        index#101: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 68
                action = prio = 1
        index#154: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 547
                action = prio = 1
        index#171: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 6 --dstport 8080
                action = prio = 1
        index#206: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 2727
                action = prio = 1
        index#231: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 2427
                action = prio = 1
        index#252: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x0806
                action = prio = 1
        index#269: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 53
                action = prio = 1
        index#289: static, refcnt = 1, SW hitcnt = 74078
                field+value = --ipproto 6 --dstport 80
                action = prio = 1
        index#380: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --srcport 53
                action = prio = 1
        index#399: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 6 --srcport 53
                action = prio = 1
        index#401: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x8902
                action = prio = 1
        index#427: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 546
                action = prio = 1
        index#428: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x8863
               action = prio = 1
        index#498: static, refcnt = 1, SW hitcnt = 1
                field+value = --ipproto 17 --dstport 67
                action = prio = 1
        index#510: static, refcnt = 1, SW hitcnt = 1
                field+value = --ipproto 6 --dstport 53
                action = prio = 1
```

Syntax

```
# iq porttbl -k

key mask table
          index   | priority |  key mask   | key option | ref_cnt
         ---------|----------|-------------|------------|---------
         00000000 | 00000014 | 0x00000008 | 0x00000001 |       4
         00000001 | 00000010 | 0x00008400 | 0x00000001 |       2
         00000002 | 00000009 | 0x00000400 | 0x00000001 |       0
         00000003 | 00000008 | 0x00010400 | 0x00000001 |      10


key hash table
Total static Entry Count: 16
Total dynamic Entry Count: 0
          index   |  key mask   | key option |    action    | ref_cnt |  hits   | static
                   field = value
         ---------|-------------|------------|--------------|---------|---------|-------
         00000060 | 0x00000008 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 EtherType = 0x8864
         ---------------------------------
         00000101 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x11
                 DST Port = 68
         ---------------------------------
         00000154 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x11
                 DST Port = 547
         ---------------------------------
         00000171 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x06
                 DST Port = 8080
         ---------------------------------
         00000206 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x11
                 DST Port = 2727
         ---------------------------------
         00000231 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x11
                 DST Port = 2427
         ---------------------------------
         00000252 | 0x00000008 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 EtherType = 0x0806
         ---------------------------------
         00000269 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x11
                 DST Port = 53
         ---------------------------------
         00000289 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 074330 |  yes
                 IP PROTO = 0x06
                 DST Port = 80
         ---------------------------------
         00000380 | 0x00008400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x11
                 SRC Port = 53
         ---------------------------------
         00000399 | 0x00008400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x06
                 SRC Port = 53
         ---------------------------------
         00000401 | 0x00000008 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 EtherType = 0x8902
         ---------------------------------
         00000427 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 IP PROTO = 0x11
                 DST Port = 546
        -------------------------------
         00000428 | 0x00000008 | 0x00000001 | 0x00000047 | 0000001 | 000000 |  yes
                 EtherType = 0x8863
         ---------------------------------
         00000498 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000001 |  yes
                 IP PROTO = 0x11
                 DST Port = 67
         ---------------------------------
         00000510 | 0x00010400 | 0x00000001 | 0x00000047 | 0000001 | 000001 |  yes
                 IP PROTO = 0x06
                 DST Port = 53
```

4.9 Add Key Mask

Description This command adds key mask. A key mask must be added first before a key using the field set in the key mask (fields can be ORed) is added. **Ingress QoS must be flushed with no dynamic entries left and disabled before adding a key mask.** A unique priority value (from 0 to 15) must be configured with --maskprio. Adding a key mask will fail if attempts to add with a priority that exists in Ingress QoS.

Syntax (Runner)

```
# iq addkeymask [fields w/o value] -maskprio <0..15>

  field:
      --srcmac: MAC address in x:x:x:x:x:x format
      --dstmac: MAC address in x:x:x:x:x:x format
      --ethtype: ether type
      --outervid: Outer VLAN ID
      --outerpbit: Outer VLAN PBit
      --innervid: Inner VLAN ID
      --innerpbit: Inner VLAN PBit
      --l3proto: e.g., IPv4 (0x0800) or IPv6 (0x86DD)
      --ipproto: such as TCP (6) or UDP (17)
      --srcip: Source IPv4/6 Address
      --dstip: Destination IPv4/6 Address
      --dscp: 6-bit IP->DSCP value
      --ipv6flowlabel: 20-bit IPv6->flowlabel value
      --srcport: 16-bit L4 SRC Port
      --dstport: 16-bit L4 DST Port
  attribute:
      --maskprio: <0..15> priority value, larger value = higher priority
```

4.10 Remove Key Mask

Description This command removes key mask.

Syntax (Runner)

```
# iq remkeymask [fields w/o value]

  field:
      --srcmac: MAC address in x:x:x:x:x:x format
      --dstmac: MAC address in x:x:x:x:x:x format
      --ethtype: ether type
      --outervid: Outer VLAN ID
      --outerpbit: Outer VLAN PBit
      --innervid: Inner VLAN ID
      --innerpbit: Inner VLAN PBit
      --l3proto: e.g., IPv4 (0x0800) or IPv6 (0x86DD)
      --ipproto: such as TCP (6) or UDP (17)
      --srcip: Source IPv4/6 Addres
      --dstip: Destination IPv4/6 Address
      --dscp: 6-bit IP->DSCP value
      --ipv6flowlabel: 20-bit IPv6->flowlabel value
      --srcport: 16-bit L4 SRC Port
      --dstport: 16-bit L4 DST Port
```

4.11 Dump Key Mask Table

Description This command displays key mask table.

Syntax (Runner)

```
# iq keymasktbl
keymask table:
        index#0: prio = 14, refcnt = 4
                mask = --ethtype
        index#1: prio = 10, refcnt = 2
                mask = --ipproto --srcport
        index#2: prio = 9, refcnt = 0
                mask = --ipproto
        index#3: prio = 8, refcnt = 10
                mask = --ipproto --dstport
```

4.12 Add Key

Description This command adds key. Default action is nop. Entry will be created as dynamic by default.

Syntax (Runner)

```
# iq addkey [field | action | attribute]

  field:
      --srcmac: MAC address in x:x:x:x:x:x format
      --dstmac: MAC address in x:x:x:x:x:x format
      --ethtype: ether type
      --outervid: Outer VLAN ID
      --outerpbit: Outer VLAN PBit
      --innervid: Inner VLAN ID
      --innerpbit: Inner VLAN PBit
      --l3proto: e.g., IPv4 (0x0800) or IPv6 (0x86DD)
      --ipproto: such as TCP (6) or UDP (17)
      --srcip: Source IPv4/6 Address
      --dstip: Destination IPv4/6 Address
      --dscp: 6-bit IP->DSCP value
      --ipv6flowlabel: 20-bit IPv6->flowlabel value
      --srcport: 16-bit L4 SRC Port
      --dstport: 16-bit L4 DST Port
    action:
      --prio: 0 = low, 1 = high
      --drop: drop packet
      --trap: trap packet before flow lookup
  attribute:
      --ent: 0 = dynamic, 1 = static
```

4.13 Remove Key

Description This command removes key.

Syntax (Runner)

```
# iq remkey [field | action]

  field:
      --srcmac: MAC address in x:x:x:x:x:x format
      --dstmac: MAC address in x:x:x:x:x:x format
      --ethtype: ether type
      --outervid: Outer VLAN ID
      --outerpbit: Outer VLAN PBit
      --innervid: Inner VLAN ID
      --innerpbit: Inner VLAN PBit
      --l3proto: e.g., IPv4 (0x0800) or IPv6 (0x86DD)
      --ipproto: such as TCP (6) or UDP (17)
      --srcip: Source IPv4/6 Address
      --dstip: Destination IPv4/6 Address
      --dscp: 6-bit IP->DSCP value
      --ipv6flowlabel: 20-bit IPv6->flowlabel value
      --srcport: 16-bit L4 SRC Port
      --dstport: 16-bit L4 DST Port
    action:
      --prio: 0 = low, 1 = high
      --drop: drop packet
      --trap: trap packet before flow lookup
```

4.14 Get Key

Description This command gets key based on the field and prints out info about this key.

Syntax (Runner)

```
# iq getkey [field]

  field:
      --srcmac: MAC address in x:x:x:x:x:x format
      --dstmac: MAC address in x:x:x:x:x:x format
      --ethtype: ether type
      --outervid: Outer VLAN ID
      --outerpbit: Outer VLAN PBit
      --innervid: Inner VLAN ID
      --innerpbit: Inner VLAN PBit
      --l3proto: e.g., IPv4 (0x0800) or IPv6 (0x86DD)
      --ipproto: such as TCP (6) or UDP (17)
      --srcip: Source IPv4/6 Address
      --dstip: Destination IPv4/6 Address
      --dscp: 6-bit IP->DSCP value
      --ipv6flowlabel: 20-bit IPv6->flowlabel value
      --srcport: 16-bit L4 SRC Port
      --dstport: 16-bit L4 DST Port
```

4.15 Dump Key Table

Description This command prints all the keys.

Syntax (Runner)

```
# iq keytbl
key table:
        index#60: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x8864
                action = prio = 1
        index#101: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 68
                action = prio = 1
        index#154: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 547
                action = prio = 1
        index#171: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 6 --dstport 8080
                action = prio = 1
        index#206: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 2727
                action = prio = 1
        index#231: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 2427
                action = prio = 1
        index#252: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x0806
                action = prio = 1
        index#269: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 53
                action = prio = 1
        index#289: static, refcnt = 1, SW hitcnt = 75218
                field+value = --ipproto 6 --dstport 80
                action = prio = 1
        index#380: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --srcport 53
                action = prio = 1
        index#399: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 6 --srcport 53
                action = prio = 1
        index#401: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x8902
                action = prio = 1
        index#427: static, refcnt = 1, SW hitcnt = 0
                field+value = --ipproto 17 --dstport 546
                action = prio = 1
        index#428: static, refcnt = 1, SW hitcnt = 0
                field+value = --ethtype 0x8863
                action = prio = 1
        index#498: static, refcnt = 1, SW hitcnt = 1
                field+value = --ipproto 17 --dstport 67
                action = prio = 1
        index#510: static, refcnt = 1, SW hitcnt = 4
                field+value = --ipproto 6 --dstport 53
                action = prio = 1
```

4.16 Set Default Priority for IP Protocol Value

Description This command configures the default priority value given an IP Protocol value.

Syntax (FAP)

```
# iq setdefaultprio --prototype <0> --protoval <0..255> --prio <0/1>
  prototype: 0 = ipproto
  protoval: protocol value (0 to 255)
  prio: 0 = low, 1 = high
```

4.17 Remove Configured Default Priority of an IP Protocol Value

Description This command removes the previously configured default priority set on an IP Protocol Value.

Syntax (Runner)

```
# iq remdefaultprio --prototype <0> --protoval <0..255>
  prototype: 0 = ipproto
  protoval: protocol value (0 to 255)
```

4.18 Example

The following example will add the required keymask and key entry to drop packet with outervid = 0x20 and srcip = 12.34.56.78.

```
# iq keymasktbl
 keymask table:
        index#0: prio = 14, refcnt = 4
                mask = --ethtype
        index#1: prio = 10, refcnt = 2
                mask = --ipproto --srcport
        index#2: prio = 9, refcnt = 0
                mask = --ipproto
        index#3: prio = 8, refcnt = 10
                mask = --ipproto --dstport
# iq disable
# iq flush
# iq addkeymask --outervid --srcip --maskprio 13
# iq enable
# iq addkey --outervid 0x20 --srcip 12.34.56.78 --drop
# iq keymasktbl
 keymask table:
        index#0: prio = 14, refcnt = 4
                mask = --ethtype
        index#1: prio = 13, refcnt = 1
                mask = --outervid --srcip
        index#2: prio = 10, refcnt = 2
                mask = --ipproto --srcport
        index#3: prio = 8, refcnt = 10
                mask = --ipproto --dstport
# iq getkey --outervid 0x20 --srcip 12.34.56.78
        index#0: dynamic, refcnt = 0, SW hitcnt = 0
                field+value = --outervid 32 --srcip 12.34.56.78
                action = drop
```

4.19 Example

The following example will modify the default ARP packet trapped to high priority CPU receive queue action to a new early CPU trap action for all ARP packets.

```
# iq getkey --ethtype 0x0806
        index#0: static, refcnt = 0, SW hitcnt = 0
                field+value = --ethtype 0x0806
                action = prio = 1
# iq remkey --ethtype 0x0806
# iq addkey --ethtype 0x0806 --trap --ent 1
# iq getkey --ethtype 0x0806
        index#0: static, refcnt = 0, SW hitcnt = 0
                field+value = --ethtype 0x0806
                action = trap
```

Chapter 5: APIs

Ingress QoS exports APIs for the user in kernel space, such as drivers or modules, to interact and configure Ingress QoS. Ingress QoS API prototypes are defined in CommEngine/kernel/linux/include/linux/iqos.h. Endianess for all the values should be in host order.

5.1 Add L4 Destination Port

Description This API adds a L4 destination port, protocol, packet priority, and the entry type to the Ingress QoS table. (Legacy API, it is backward compatible.)

Syntax
```
int iqos_add_L4port
    (uint8_t ipProto, uint16_t destPort, iqos_ent_t ent, iqos_prio_t prio);
```

Parameters

ipProto	IP L4 protocol TCP or UDP
destPort	L4 destination port.
ent	Entry type: static or dynamic.
prio	Packet priority assigned to the incoming packet.

Return Value Success: 0
Failure: Otherwise

5.2 Remove L4 Destination Port

Description This API removes a previously added L4 destination port, protocol, and the entry type from the Ingress QoS table. (Legacy API, it is backward compatible.)

Syntax
```
int iqos_rem_L4port(uint8_t ipProto, uint16_t destPort, iqos_ent_t ent);
```

Parameters

ipProto	IP L4 protocol TCP or UDP
destPort	L4 destination port.
ent	Entry type: static or dynamic

Return Value Success: 0
Failure: Otherwise

5.3 Get L4 Destination Port Priority

Description This API returns the current assigned packet priority for a L4 destination port, and protocol from the Ingress QoS table. (Legacy API, it is backward compatible.)

Syntax
```
int iqos_prio_L4port(uint8_t ipProto, uint16_t destPort);
```

Parameters

ipProto	IP L4 protocol TCP or UDP
destPort	L4 destination port

Return Value Success: packet priority IQOS_PRIO_LOW/IQOS_PRIO_HIGH
Failure: packet priority IQOS_PRIO_LOW

5.4 Initialize Keymask Parameter

Description This API is used to initialize parameter holder for configuring Ingress QoS Keymask.

Syntax `int iqos_keymask_param_start(iqos_param_t *param);`

Parameters param Ingress QoS parameter holder

Return Value Success: 0, parameter holder is initialized and set to be used for Keymask operation.
Failure: Otherwise

5.5 Set Field and Value to the Keymask Parameter

Description This API sets field and its value to the Ingress QoS parameter holder for configuring Ingress QoS Keymask.

Syntax `int iqos_keymask_param_field_set(iqos_param_t *param, uint32_t field, uint32_t *val_ptr);`

Parameters param Ingress QoS parameter holder
field Ingress QoS field type (details of field type and value are in later table)
val_ptr value holder

Return Value Success: 0
Failure: Otherwise

5.6 Commit and Add the Keymask Parameter

Description This API commits the Ingress QoS parameter holder for a Keymask add event.

Syntax `int iqos_keymask_commit_and_add(iqos_param_t *param, uint8_t prio);`

Parameters param Ingress QoS parameter holder
prio Priority

Return Value Success: 0
Failure: Otherwise

5.7 Commit and Delete the Keymask Parameter

Description This API commits the Ingress QoS parameter holder for a Keymask delete event.

Syntax `int iqos_keymask_commit_and_delete(iqos_param_t *param);`

Parameters param Ingress QoS parameter holder

Return Value Success: 0
Failure: Otherwise

5.8 Initialize Key Parameter

Description This API is used to initialize parameter holder for configuring Ingress QoS Key.

Syntax `iqos_key_param_start(iqos_param_t *param);`

Parameters

param	Ingress QoS parameter holder

Return Value Success: 0, parameter holder is initialized and set to be used for Key operation.
Failure: Otherwise

5.9 Set Field and Value to the Key Parameter

Description This API sets field and its value to the Ingress QoS parameter holder for configuring Ingress QoS Key.

Syntax
```
int iqos_key_param_field_set(iqos_param_t *param, uint32_t field, uint32_t *val_ptr, uint32_t
val_size);
```

Parameters

param	Ingress QoS parameter holder
field	Ingress QoS field type (details of field type and value are in later table)
val_ptr	Value holder
val_size	Size of data in val_ptr

Return Value Success: 0
Failure: Otherwise

5.10 Set Action and Value to the Key Parameter

Description This API sets action and its value to the Ingress QoS parameter holder for configuring Ingress QoS Key.

Syntax `int iqos_key_param_action_set(iqos_param_t *param, uint32_t action, uint32_t value);`

Parameters

param	Ingress QoS parameter holder
action	Ingress QoS action type (details of action type and value ars in later table)
value	Value used for the action

Return Value Success: 0
Failure: Otherwise

5.11 Commit and Add the Key Parameter

Description This API commits the Ingress QoS parameter holder for a Key add event.

Syntax `int iqos_key_commit_and_add(iqos_param_t *param, uint8_t type);`

Parameters

param	Ingress QoS parameter holder
type	dynamic (0) or static (1)

Return Value Success: 0
Failure: Otherwise

5.12 Commit and Delete the Key Parameter

Description This API commits the Ingress QoS parameter holder for a Key delete event.

Syntax `int iqos_key_commit_and_delete(iqos_param_t *param, uint8_t type);`

Parameters
param Ingress QoS parameter holder
type Dynamic (0) or static (1)

Return Value
Success: 0
Failure: Otherwise

5.13 Commit and Get the Key Parameter

Description This API commits the Ingress QoS parameter holder for a Key get event.

Syntax `int iqos_key_commit_and_get(iqos_param_t *param);`

Parameters param Ingress QoS parameter holder

Return Value
Success: 0 when entry is found, and update action and its value in the parameter holder
Failure: Otherwise

5.14 Flush All Dynamic Key Entries

Description This API flushes all dynamic key entries and key mask if there is no key referred to the specific key mask.

Syntax `int iqos_flush(void);`

Parameters None

Return Value None

5.15 Set Status

Description This API flushes all dynamic key entries and key mask if there is no key referred to the specific key mask.

Syntax `int iqos_set_status(uint32_t status);`

Parameters Status 0 = disable, 1 = enable

Return Value Success: 0
Failure: Otherwise

Table 5: Fields Table

Field	Description
IQOS_FIELD_SRC_MAC	Source MAC Address. If used for configuring key, val_ptr must point to starting of a 6 bytes memory and val_size given should be 6.
IQOS_FIELD_DST_MAC	Destination MAC Address. If used for configuring key, val_ptr must point to starting of a 6 bytes memory and val_size given should be 6.
IQOS_FIELD_ETHER_TYPE	Ether_type. 2-byte EtherType value in the Ethernet header.
IQOS_FIELD_OUTER_VID	Outer VLAN ID. 2-byte VLAN ID value in the outer VLAN tag. (Also used in single VLAN tag scenario.)
IQOS_FIELD_OUTER_PBIT	Outer PBIT. 1-byte PBIT value in the outer VLAN tag. (Also used in single VLAN tag scenario.)
IQOS_FIELD_INNER_VID	Inner VLAN ID. 2-byte VLAN ID value in the inner VLAN tag. (Used in double-tagged scenario. There is no support for 3 or more VLAN tags.)
IQOS_FIELD_INNER_PBIT	Inner PBIT. 1-byte PBIT value in the inner VLAN tag. (Used in double-tagged scenario. There is no support for 3 or more VLAN tags.)
IQOS_FIELD_L3_PROTO	L3 Protocol. 2-byte L3 protocol value, i.e., IPv4 = 0x8000.
IQOS_FIELD_IP_PROTO	IP Protocol. 1-byte IP protocol value, i.e., TCP = 6.
IQOS_FIELD_SRC_IP	Source IP address. 4-byte IP address for IPv4 or 16-byte for IPv6.
IQOS_FIELD_DST_IP	Destination IP address. 4-byte IP address for IPv4 or 16-byte for IPv6.
IQOS_FIELD_DSCP	DSCP. 1-byte DSCP value from IPv4 header.
IQOS_FIELD_IPV6_FLOW_LABEL	IPv6 Flow label. 4-byte Flow Label value from IPv6 header.
IQOS_FIELD_SRC_PORT	L4 Source Port. 2-byte source port from L4 (TCP/UDP) header.
IQOS_FIELD_DST_PORT	L4 Destination Port. 2-byte destination port from L4 (TCP/UDP) header.

Table 6: Actions Table

Action	Description
IQOS_ACTION_NOP	No operation.
IQOS_ACTION_PRIO	Setting the received queue priority when Ingress QoS key hits. Value is either 0 for low priority or 1 for high priority.
IQOS_ACTION_DROP	Dropping the packet when key hits.
IQOS_ACTION_TRAP	Trapping the packet early in classification to CPU when key hits. (If given key is supported by hardware network processor, it usually traps to high priority receive queue.)

5.16 Configuration Sequence

There are two important criteria that one has to follow:

1. A matching key mask must be configured before adding a key.
2. In order to add a new keymask, Ingress QoS must have no existing dynamic entry and it has to be in disabled stage.

The following shows an example of how to add a new key mask and a key to Ingress QoS from a kernel space user module.

```
Int rc;
iqos_param_t iqos_param;
uint8_t src_mac[6] = {0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc};
uint32_t dst_ip = 0xc0a80120;
uint16_t dst_port = 123;

rc = iqos_set_status(0);/* to disable Ingress QoS */
if (rc)
   return rc;
iqos_flush();
rc = iqos_keymask_param_start(&iqos_param);
rc = rc? rc : iqos_keymask_param_field_set(&iqos_param, IQOS_FIELD_SRC_MAC, NULL);
rc = rc? rc : iqos_keymask_param_field_set(&iqos_param, IQOS_FIELD_DST_IP, NULL);
rc = rc? rc : iqos_keymask_param_field_set(&iqos_param, IQOS_FIELD_DST_PORT, NULL);
rc = rc? rc : iqos_keymask_commit_and_add(&iqos_param, 8);
if (rc)
   printk("error!\n");
rc = iqos_set_status(1);/* to enable Ingress QoS */

rc = rc? rc : iqos_key_param_start(&iqos_param);
rc = rc? rc : iqos_key_param_field_set(&iqos_param, IQOS_FIELD_SRC_MAC, (uint32_t *)src_mac, 6);
rc = rc? rc : iqos_key_param_field_set(&iqos_param, IQOS_FIELD_DST_IP, (uint32_t *)&dst_ip, 4);
rc = rc? rc : iqos_key_param_field_set(&iqos_param, IQOS_FIELD_DST_PORT, (uint32_t *)&dst_port, 2);
rc = rc? rc : iqos_key_commit_and_add(&iqos_param, 1);/* static entry */
```

Chapter 6: Performance

Under normal conditions, with no CPU congestion, all packets are forwarded.

Under CPU congestion only high-priority packets are forwarded and all low priority packets are dropped by Ingress QoS.

6.1 All High-Priority Flows

If all of the flows are at the same priority, none of the flows gets priority. In this scenario, Ingress QoS tries to protect all high-priority flows. However, none of them are protected because there is no priority among them. Once congestion is reached, packets from all the flows are dropped proportionately because the accelerator/CPU cannot handle the packet rate.

Behavior in this scenario is very similar to when the Ingress QoS feature was not implemented (before 4.10 release) or the feature is disabled.

The overall performance should be slightly lower than when the feature is disabled.

6.2 All Low-Priority Flows

When all of the flows are of same priority (low) none of the flows get priority. In this scenario, Ingress QoS attempts to protect a high priority flow, but there are no high priority flows. After congestion, packets from all the low priority flows are dropped proportionately by the Ingress QoS to preserve CPU cycles for a high-priority flow.

NOTE: The maximum forwarding rate should be slightly lower than when the feature is disabled, but after congestion is hit the output rate should come down steeply with the increase in the low-priority input packet rate.

CAUTION! As the low-priority input rate increases, the low-priority output rate decreases after CPU congestion is reached. This is contrary to what most people expect and differs from the behavior when Ingress QoS feature was not implemented.

6.3 Mix of High- and Low-Priority Flows

In this scenario, Ingress QoS tries to protect a high priority flow at the cost of low-priority flows under CPU congestion. Packets from a low priority flow are dropped when there is CPU congestion. When there is no congestion all packets are forwarded. In addition, when a packet is classified as high priority by the network processor, it will be placed into high priority receive queue. The Ethernet driver will always serve the high priority queue first. This ensures the handling of high priority packets, but there could be cases of tail drop in the low priority receive queue.

6.4 Early Trap and Drop

(Applicable only for the Runner-based platform.) When using these two features, it will enable ingress filter in Runner. This ingress filter is executed on every packet in the critical data path in the Runner. Therefore, it has negative impact on performance of other traffic.

Revision History

963XX-AN403-R, December 18, 2018

- Updated Introduction to Chapter 1, Overview
- Added Note to Chapter 1, Overview
- Updated Introduction to Chapter 3, Ingress QoS
- Added Caution note to Introduction of Chapter 3, Ingress QoS
- Updated content and changed heading to "CPE Behavior — No Congestion State" on page 7
- Updated content and changed heading to "CPE Behavior — Congestion State" on page 7
- Updated "Force Trap" on page 8
- Updated introduction to Chapter 4, CLI Commands
- Added Note to Chapter 4, CLI Commands
- Updated `l3proto` value for IPv6 throughout
- Updated "Example" on page 20
- Added "Example" on page 21
- Updated introduction to Chapter 5, APIs

963XX-AN402-R, July 11, 2018

- Updated introduction to Chapter 1, Overview
- Updated introduction to Chapter 2, Enable and Disable Ingress QoS
- Updated "Enable Ingress QoS" on page 6
- Updated "Disable Ingress QoS" on page 6
- Updated introduction to Chapter 3, Ingress QoS
- Updated "Ingress QoS Packet Priority" on page 8
- Updated Table 2, Default Entries Based on TCP/UDP + L4 Destination Ports
- Added "Force Trap" on page 8
- Added "Drop" on page 8
- Updated Chapter 4, CLI Commands
- Added "Status" on page 10
- Updated "Flush" on page 11
- Updated "Dump Key Mask and Key Table" on page 13
- Added "Add Key Mask" on page 15
- Added "Remove Key Mask" on page 15
- Added "Dump Key Mask Table" on page 16
- Added "Add Key" on page 16
- Added "Remove Key" on page 17
- Added "Get Key" on page 17
- Added "Dump Key Table" on page 18
- Added "Set Default Priority for IP Protocol Value" on page 19
- Added "Remove Configured Default Priority of an IP Protocol Value" on page 19
- Added "Example" on page 19
- Updated introduction to Chapter 5, APIs
- Updated "Add L4 Destination Port" on page 20
- Updated "Remove L4 Destination Port" on page 20

- Updated “Get L4 Destination Port Priority” on page 20
- Added “Set Field and Value to the Keymask Parameter” on page 21
- Added “Commit and Add the Keymask Parameter” on page 21
- Added “Commit and Delete the Keymask Parameter” on page 21
- Added “Set Field and Value to the Keymask Parameter” on page 21
- Added “Initialize Key Parameter” on page 22
- Added “Set Field and Value to the Key Parameter” on page 22
- Added “Set Action and Value to the Key Parameter” on page 22
- Added “Commit and Add the Key Parameter” on page 22
- Added “Commit and Delete the Key Parameter” on page 23
- Added “Commit and Get the Key Parameter” on page 23
- Added “Flush All Dynamic Key Entries” on page 23
- Added “Set Status” on page 24
- Updated “Mix of High- and Low-Priority Flows” on page 26
- Added “Early Trap and Drop” on page 26

Previous Release History

Revision	Date	Change Description
963XX-AN101-R	01/18/15	**Updated:** ■ “Add Port” on page 12 ■ “Dump Port Table” on page 13 ■ “Add L4 Destination Port” on page 14
963XX-AN400-R	12/14/10	Initial release