# Intel® Ethernet 800 Series Linux Flow Control

## Configuration Guide for RDMA Use Cases

**Ethernet Products Group (EPG)**

***Rev. 1.1***

***September 2022***

# Contents

Did this document
help answer your questions?

# Figures

Did this document
help answer your questions?

👍 👎

# Tables

# Revision History

| Revision | Date | Comments |
|---|---|---|
| 1.1 | September 16, 2022 | Changes to this document include the following:<br>• Updated QoS/Flow Control Limitations on the 800 Series<br>• Updated Flow Control in RDMA Networks<br>• Updated Types of Flow Control: LFC vs. PFC<br>• Updated DCB Willing vs. Non-willing Modes<br>• Updated Determining PFC Priority Mode: PCP vs. DSCP<br>• Updated Assigning an Application to a Traffic Class<br>• Updated Step 1: Determine Flow Control Design Needed<br>• Updated Step 2: Kernel Priority (sk_prio) Or DSCP to UP Mapping<br>• Updated Step 3: UP to TC<br>• Added Step 4: Set ToS in the Application (or for All RoCEv2 Traffic)<br>• Updated Firmware vs. Software DCB<br>• Added Software DCB Willing Mode<br>• Updated Separating and Prioritizing Traffic Streams<br>• Updated Configuring PFC<br>• Added Option A: PCP PFC - Set Priorities for Drop or No-drop<br>• Added Option B: DSCP PFC<br>• Updated Run Applications with the Right Priority<br>• Updated Example 2 - Adapters Connected Through a Switch – Willing Mode on Adapters, DCB Configured on Switch<br>• Added Example 3 - DSCP PFC with Non-Default TCs - Non-Willing Mode on Adapters, ECN Configured<br>• Updated Example 4 - PCP PFC with Multiple TCs (1 for RDMA, 1 for LAN) – No VLANs |
| 1.0 | April 22, 2021 | Initial public release. |

# 1.0 Introduction

This document contains information on using Ethernet flow control on Intel® Ethernet 800 Series Network Adapters, with a focus on best practices for Linux RDMA traffic.

It includes:

- Background on Ethernet flow control and Data Center Bridging (DCB).
- Differences between Link-level Flow Control (LFC) and Priority Flow Control (PFC).
- Configuration steps for each type on 800 Series Linux hosts.
- Verification tips.

## 1.1 QoS/Flow Control Limitations on the 800 Series

- Although the 800 Series hardware supports eight Traffic Classes (TCs), the maximum supported configuration is four TCs per port. Only one TC can have Priority Flow Control enabled per port.

| Number of Adapter Ports | Traffic Class Recommendation | RDMA |
|---|---|---|
| 1, 2, or 4 | Up to four TCs, with one of them enabled with PFC. | Supported |
| More than 4 | No DCB Support | Not Supported |

- In RoCEv2 mode, if no flow control is detected (either LFC or PFC), the driver automatically de-tunes. This is an intentional design to allow RoCEv2 to operate without flow control, but with lower performance.
- When the 800 Series is in firmware Link Layer Discovery Protocol (LLDP) mode, only three application priorities are supported. Software LLDP supports 32. This refers to the LLDP APP TLV - see `man lldptool-app` for more info.

# 2.0 Background

## 2.1 Ethernet Flow Control

By design, Ethernet is an unreliable protocol with no guarantee that packets arrive at their destination correctly and in order. Instead, Ethernet relies on upper-layer protocols (such as TCP) or applications to provide reliable service and error correction.

The 802.3x standard introduced flow control to the Ethernet protocol, defining a mechanism for throttling the flow of data between two directly connected full-duplex network devices. If the sender transmits data faster than the receiver can accept it, the overwhelmed receiver can send a pause signal (Xoff or *transmit off*) to the sender, requesting that the sender stop transmitting data for a specified period of time. The sender resumes transmission either after the timeout period expires or if the receiver indicates that it is ready to accept more data by sending an Xon (*transmit on*) signal.

Without flow control, data might be lost or need to be re-transmitted by a ULP or application, which can significantly affect performance.

## 2.2 Flow Control in RDMA Networks

The 800 Series supports both iWARP and RoCEv2 RDMA transports. Flow control is strongly recommended for RoCEv2, but iWARP also benefits.

| Base Transport | Flow Control Requirements |
|---|---|
| iWARP TCP | • iWARP runs over TCP, a reliable protocol that implements its own flow control.<br>• TCP's flow control might be relatively slow to respond in a high-performance, low-latency RDMA environment, especially under bursty traffic patterns.<br>• Ethernet flow control is **optional, but can be beneficial for iWARP**.<br>• iWARP mode requires VLAN to be configured fully to enable PFC. |
| RoCEv2 UDP | • RoCEv2 runs over UDP, an unreliable protocol with no built-in flow control.<br>• RoCEv2 therefore requires a lossless Ethernet network to ensure packet delivery.<br>• If the *irdma* driver is in RoCEv2 mode and detects no flow control, it automatically de-tunes, causing lower performance.<br>• Flow control is **always recommended for RoCEv2**. |

## 2.3 Types of Flow Control: LFC vs. PFC

Ethernet standards define two types of flow control:

- Link-level Flow Control (LFC)
- Priority Flow Control (PFC)

Both types use Xon/Xoff pause frames to control data transmission. The primary difference is that LFC pauses all traffic on a link, but PFC supports Quality-of-Service (QoS) by defining different traffic priorities that can be individually paused. PFC therefore offers greater flexibility when running multiple traffic streams

**NOTE**

Despite LFC being called *link-level* flow control, both LFC and PFC operate at the data link level (OSI Layer 2).

## Table 1. LFC vs. PFC Comparison

| | LFC | PFC |
|---|---|---|
| **Standard** | IEEE 802.3x (1997) | IEEE 802.1Qbb (2011) |
| **Pause Type** | Global pause - pauses the entire link, affecting all traffic on that link.<br>If a link carries multiple traffic streams, a high-flow stream can cause the link to pause, thereby blocking ALL streams. | Priority pause - defines eight priorities that can be individually paused.<br>High-bandwidth applications can be paused while allowing low-bandwidth applications to continue running. |
| **Traffic Shaping** | None. | Supports traffic classes, priorities, bandwidth allocation, and other QoS features. |
| **Ease of Setup** | Straightforward. Turn on Tx/Rx flow control on both the adapter and switch. | More complicated. Priorities, traffic classes, bandwidth allocations, and willing/non-willing mode must be configured on the adapter, switch, or both. |

PFC and LFC are mutually exclusive. Only one type at a time can be enabled on a device.

- PFC is generally recommended. It has greater flexibility to handle multiple traffic streams and enhanced QoS capabilities.
- LFC can be used in situations where there are no differentiated classes of traffic. It is usually used for testing purposes for RDMA.

# 3.0 Link-Level Flow Control

## 3.1 LFC Setup Instructions

Configuring LFC on an 800 Series network is relatively straightforward; enable flow control in both directions (Tx and Rx) on both sides of the link.

- If your hosts are connected through a switch, you must also enable flow control on the switch ports.
- If your hosts are connected back-to-back, enable LFC on both adapters.

The examples below use **eth0** as the 800 Series net device name (use **ip a** to find the device name on your system).

Switch settings vary by manufacturer. In your switch manual, look for syntax containing words like: flowcontrol, flow-control, tx-pause, or rx-pause.

To enable LFC on your network:

1. Disable firmware-based DCB on the adapter.

```
# ethtool --set-priv-flags <interface> fw-lldp-agent off
```

2. Verify that firmware DCB is disabled.

```
# ethtool --show-priv-flags <interface> | grep fw-lldp-agent
fw-lldp-agent : off
```

3. Ensure that**lldpad** is not running.

```
# ps -ef | grep lldpad
```

4. Disable PFC on your switch, if applicable (show PFC status per port).

```
switch>show priority-flow-control
```

For example, disable PFC on a given port (like port 31/1) on an Arista 7060CX:

```
switch>enable
switch#configure
switch(config)#interface Ethernet 31/1
switch(config-if-Et31/1)#no priority-flow-control
```

> **NOTE**
> Some switches allow for a range of ports to be specified, for example 1/1-32/1.

5. Enable link-level flow control on the adapter.

```
# ethtool -A eth0 rx on tx on
```

6. Check LFC settings on the adapter.

```
# ethtool -a eth0
Pause parameters for eth0:
Autonegotiate: on
RX: on
TX: on
RX negotiated: on
TX negotiated: on
```

7. Enable flow control on the switch ports.

   For example, enable Rx and Tx flow control on switch port 21 on an Arista 7060CX:

```
switch>enable
switch#configure
switch(config)#interface Ethernet 21/1
switch(config-if-Et31/1)#flowcontrol receive on
switch(config-if-Et31/1)#flowcontrol send on
```

8. Check LFC settings on the switch.

   For example, show flow control settings on ports 21-22 on an Arista 7060CX:

```
Switch(config-if-Et31/1)#show interfaces ethernet 21/1-22/1 flowcontrol
Port Send FlowControl Receive FlowControl RxPause TxPause
admin oper admin oper
---------- -------- -------- -------- -------- ------------- -------------
Et21/1 on on off off 170373384 0
Et22/1 on on off off 289143370 0
```

## 3.2 Symmetric vs. Asymmetric LFC

LFC operates in both the Tx (send) and Rx (receive) directions.

- Tx flow control means that the port generates and sends Ethernet pause frames as needed.
- Rx flow control means that the port accepts and responds to Ethernet pause frames received from the connected peer.

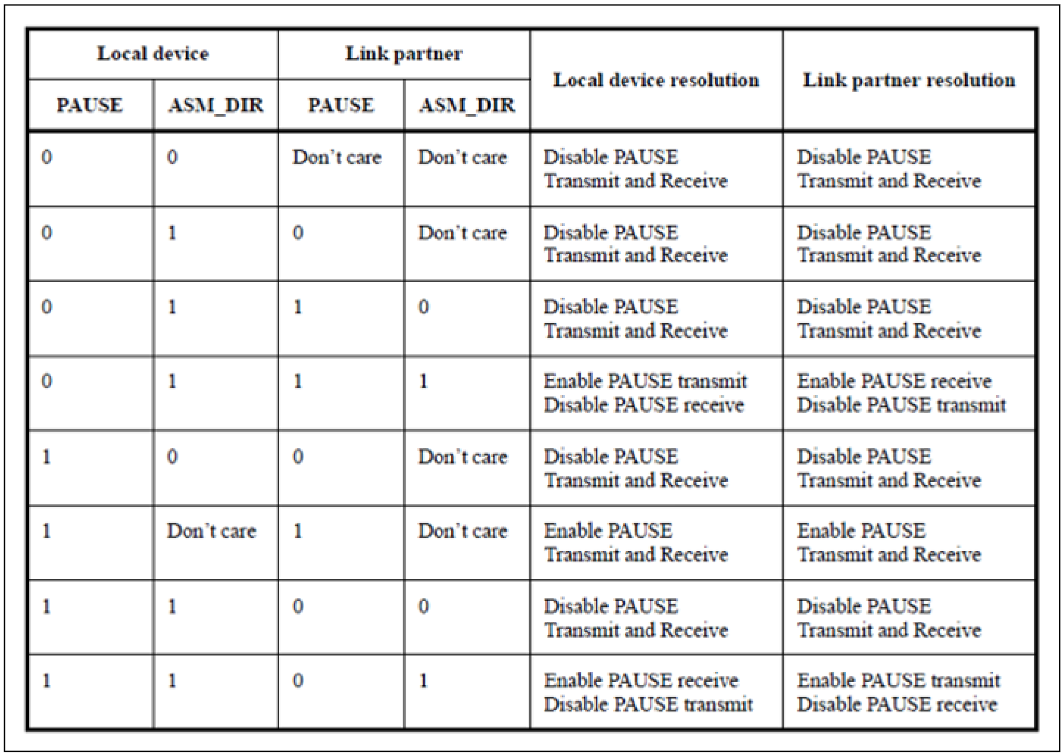
When using LFC on the 800 Series, Intel recommends enabling both Tx and Rx flow control on both sides of the link. Also, configuring asymmetric settings (different Tx or Rx settings on each side) might have non-intuitive results.

For expected behavior, see the pause resolution table of IEEE 802.3bz shown in Figure 1.

From the IEEE Standard:

> *"The PAUSE bit indicates that the device is capable of providing the symmetric PAUSE functions as defined in Annex 31B. The ASM_DIR bit indicates that asymmetric PAUSE operation is supported. The value of the PAUSE bit when the ASM_DIR bit is set indicates the direction PAUSE frames are supported for flow across the link. "*

### Figure 1. Pause Resolution Table



| Local device | | Link partner | | Local device resolution | Link partner resolution |
|---|---|---|---|---|---|
| PAUSE | ASM_DIR | PAUSE | ASM_DIR | | |
| 0 | 0 | Don't care | Don't care | Disable PAUSE<br>Transmit and Receive | Disable PAUSE<br>Transmit and Receive |
| 0 | 1 | 0 | Don't care | Disable PAUSE<br>Transmit and Receive | Disable PAUSE<br>Transmit and Receive |
| 0 | 1 | 1 | 0 | Disable PAUSE<br>Transmit and Receive | Disable PAUSE<br>Transmit and Receive |
| 0 | 1 | 1 | 1 | Enable PAUSE transmit<br>Disable PAUSE receive | Enable PAUSE receive<br>Disable PAUSE transmit |
| 1 | 0 | 0 | Don't care | Disable PAUSE<br>Transmit and Receive | Disable PAUSE<br>Transmit and Receive |
| 1 | Don't care | 1 | Don't care | Enable PAUSE<br>Transmit and Receive | Enable PAUSE<br>Transmit and Receive |
| 1 | 1 | 0 | 0 | Disable PAUSE<br>Transmit and Receive | Disable PAUSE<br>Transmit and Receive |
| 1 | 1 | 0 | 1 | Enable PAUSE receive<br>Disable PAUSE transmit | Enable PAUSE transmit<br>Disable PAUSE receive |

**NOTE**

Some switches might not support Tx flow control. In this case, enable Rx-only flow control on the switch, and either Tx/Rx or Tx-only on the adapter.

# 4.0 Priority Flow Control - Fundamentals

PFC is defined by IEEE Standard 802.1Qbb and is part of the DCB suite of enhancements designed to make Ethernet a more viable, competitive transport in compute and storage environments.

The following sections provide a brief overview of the DCB standards and the role of PFC.

## 4.1 DCB Standards

The goal of DCB is to create a completely loss-less Ethernet network that supports bandwidth allocation across links. The features of DCB are applicable to any high-performance Ethernet environment and have significant benefits for both LAN and RDMA traffic.

Several different parts work together to make this happen:

- **PFC:** IEEE 802.1Qbb — Defines eight different traffic priorities that can be paused independently.
- **Enhanced Transmission Selection (ETS):** IEEE 802.1Qaz — Assigns bandwidth percentages to each priority.
- **Congestion Notification:** IEEE 802.1Qau — End-to-end congestion management, further avoiding frame loss.
- **Data Center Bridging Capabilities Exchange Protocol (DCBX):** IEEE 802.1az (same standard as ETS) — Discover and exchange DCB capabilities between link neighbors. Based on functionality provided by Link Layer Discovery Protocol (LLDP) (IEEE 802.1AB).
- **Differentiated Services Code Point (DSCP):** RFC 2474 — Defines the IP header field called Differentiated Services (DS) that selects packets based on the value in this field for buffer management and packet scheduling.

### 4.1.1 DCB Willing vs. Non-willing Modes

DCB standards have a concept of willing vs. non-willing DCB configuration. This refers to whether the device is willing to receive its DCB settings from its link neighbor.

- In willing mode, a DCB-enabled device can query its neighbor's DCB settings, then apply the same settings to itself.
- In non-willing mode, DCB settings on the device must be explicitly configured.

A common strategy for using willing and non-willing modes in a cluster:

1. Set switches as non-willing.
2. Configure DCB (priority settings, traffic classes, bandwidth allocations, etc.) on the switch ports.
3. Set adapters as willing.
4. Adapters are automatically configured.

This helps simplify DCB cluster configuration by centralizing DCB settings on a switch and pushing the configuration to the adapters (rather than configuring each host individually).

Priority flow control (PFC) is supported on 800 Series in both willing and non-willing modes. 800 Series also has two DCB modes: software and firmware. For more background on software and firmware modes, refer to the Intel® Ethernet 800 Series ice driver README.

- For PFC willing mode, software DCB is recommended but firmware DCB is also supported.
- For PFC non-willing mode, software DCB must be used.

## 4.2 Determining PFC Priority Mode: PCP vs. DSCP

An Ethernet frame's priority can be determined by one of two distinct values: PCP (VLAN) or DSCP.

Priority Code Point (PCP) is used to classify and manage network traffic, and providing QoS in Layer 2 Ethernet networks. It uses the 3-bit PCP field in the VLAN header for packet classification.

Differentiated Services or DiffServ uses a 6-bit DSCP in the 8-bit DS field in the IP header for packet classification. The DS field replaces the outdated IPv4 TOS field. Of the 6 DSCP bits, 3 most significant bits represent priority value and the next 3 bits represent the drop precedence within each traffic class.

Intel's ice driver supports two PFC modes: Layer 3 DSCP-based Quality of Service (L3 QoS) and L2 VLAN based QoS in the PF driver. For RoCEv2 traffic, VLAN priority tags or DSCP values must be configured on the network. PFC mode is configured per port.

| Value | Reference | Layer | Field Description |
|---|---|---|---|
| PCP | IEEE 802.1Qbb | 2 | Priority determined by the 3-bit 802.1p Priority Code Point (PCP) field in a frame's VLAN tag.<br>Also sometimes called Class of Service (CoS). |
| DSCP | RFC 4594 | 3 | Priority determined by the 6-bit Differentiated Services Code Point (DSCP) value in the IPv4 or IPv6 header.<br>DSCP is the upper 6 bits of the Type of Service (ToS) field. |

Ethernet devices might choose to use either value when making QoS priority decisions. This setting is usually referred to as trust mode, with options like CoS Trust, DSCP Trust, or Untrusted.

**NOTE**

L3 QoS mode is not available when FW-LLDP is enabled. You also cannot enable FW-LLDP if L3 QoS mode is active. See the *L3 QoS mode* section in the ice README for more details.

**NOTE**

VLAN UP may be included in the L2 header even when the port is configured for DSCP PFC.

**NOTE**

Contrary to what its name might suggest, in RoCEv2 mode VLAN-based PFC can be used without explicitly creating new VLANs. Instead, when PFC is enabled, *VLAN 0 priority tagging* is automatically and transparently enabled on otherwise untagged interfaces. This creates a VLAN header so that the PCP field exists, but the VLAN ID is 0. Also, the VLAN priority value is influenced by setting the ToS value at the application level. While this is true for RoCEv2 mode, iWARP mode requires VLAN to be configured fully to enable PFC. Use a non-zero VLAN tag when setting up iWARP.

## 4.3 Assigning an Application to a Traffic Class

In Linux, there is a chain of mappings for VLAN PFC that starts with setting the ToS field in the IPv4/6 header that ultimately determines traffic class:

Type of Service (ToS) → Kernel Priority (sk_prio) → User Priority (UP) → Traffic Class (TC)

If using DSCP, the chain of mapping is:

Type of Service (ToS) → DSCP → User Priority (UP) → Traffic Class (TC)

QoS values, like bandwidth allocations or drop/no-drop characteristics, can then be applied to each TC. This applies whether you're using explicit VLANs or VLAN 0.

The rest of this section outlines the steps taken in designing and implementing this mapping.

### 4.3.1 Mapping Steps Details

#### 4.3.1.1 Step 1: Determine Flow Control Design Needed

Ensure that the QoS design is defined for the application per port. You will need to know the following parameters:

- Number of TCs and which TCs will use PFC
- DSCP mode or VLAN PFC/PCP mode to be used for PFC
- Priorities for the TCs
- Bandwidth allocation for the TCs
- Application ToS values

**NOTES**

- Common ToS values for VLAN PFC are 0, 8, 16, and 24. These correspond with priorities 0, 2, 6, and 4, respectively. See Step 2: Kernel Priority (sk_prio) Or DSCP to UP Mapping for details.
- DSCP is the upper six bits of the 8-bit ToS field. For historical reasons, DSCP and ToS are often used somewhat interchangeably, but they are different. Applications tend to set the entire ToS field, as this document references.

### 4.3.1.2 Step 2: Kernel Priority (sk_prio) Or DSCP to UP Mapping

#### Option A: PCP/VLAN PFC: ToS to Kernel Priority Mapping to UP Mapping

The ToS value automatically maps to Linux kernel priorities (hard coded in Linux based on **ip_tos2prio** in *net/ipv4/route.c*).

In the 800 Series, kernel priority to user priority (UP) is a hard-coded 1:1 mapping.

Mappings are:

```
ToS 0 --> sk_prio 0 --> UP 0
ToS 8 --> sk_prio 2 --> UP 2
ToS 24 --> sk_prio 4 --> UP 4
ToS 16 --> sk_prio 6 --> UP 6
```

For more details, refer to:

http://linux-tc-notes.sourceforge.net/tc/doc/priority.txt

Also, refer to `man tc-prio`. Although **tc-prio** is about Linux traffic control (note that the 800 Series cannot use this for RDMA traffic), the chart with TOS and Linux Priority lists the hard-coded values.

**Linux Source Notes:**

PCP Mappings are implemented in the Linux kernel and drivers using `rt_tos2priority`. For example, `prio = rt_tos2priority(tos)` (from *drivers/infiniband/core/cma.c*).

This is how 0, 2, 4, and 6 occurs.

**NOTES**

- The UP value is the priority referenced in the PFC.
- Competing technologies might use **mqprio qdisc** (see `man tc-mqprio`) to adjust this mapping, but the 800 Series implementation does not.
- The 800 Series ADQ feature uses **mqprio** to direct traffic. ADQ and PFC cannot be used at the same time.

#### Option B: DSCP to UP Mapping

DSCP mapping is implemented in the Linux kernel and drivers using ToS to DSCP direct mapping. Note that ToS is deprecated in favor of DSCP. The two low-order bits are used for ECN, while the upper six bits are used for the DSCP value (the *priority*).

DSCP marking uses 6 bits of the 8-bit TOS field in the IPv4 Header to provide up to 64 classes (or code points) for traffic, thus the value for the DSCP field is four times the value of PCP ToS. DSCP to UP translation table has 64 entries and provides a translation from every one of 64 DSCP values to a 3-bit UP value.

The following table shows the commonly used DSCP Values and their priority values.

| DSCP Value | Decimal Value | ToS (4 x DSCP) | UP |
|---|---|---|---|
| 000 000 | 0 | 0 | 0 |
| 001 000 | 8 | 32 | 1 |
| 010 000 | 16 | 64 | 2 |
| 010 100 | 20 | 80 | 2 |
| 011 000 | 24 | 96 | 3 |
| 011 010 | 26 | 104 | 3 |
| 100 000 | 32 | 128 | 4 |
| 100 110 | 38 | 152 | 4 |
| 101 000 | 40 | 160 | 5 |
| 101 110 | 46 | 184 | 5 |
| 110 000 | 48 | 192 | 6 |
| 111 000 | 56 | 224 | 7 |

### 4.3.1.3 Step 3: UP to TC

The ETS aspect of DCB is used to configure UP to TC mapping as a table per port.

ETS has a **up2tc** option, configurable on Linux using the **lldptool** utility.

For example, this command assigns packets marked with prio=2 to TC1, packets with prio=0 to TC2, and all other packet priorities to TC0.

```
# lldptool -T -i eth0 -V ETS-CFG up2tc=0:2,1:0,2:1,3:0,4:0,5:0,6:0,7:0
```

Linux traffic shaping utilities like **tc**, **cgroups**, or **egress-qos-map** do not work for RDMA applications because RDMA applications bypass the kernel. If running a combination of LAN and RDMA traffic, you can still use Linux traffic shaping for LAN traffic, but you must use the PCP or DSCP based mappings described in this section for RDMA traffic.

### 4.3.1.4 Step 4: Set ToS in the Application (or for All RoCEv2 Traffic)

ToS is an 8-bit field in the IPv4/6 header. It can be set in two different ways on the 800 Series:

- For an application: Use the application command line.

  For example:

  — LAN applications:

```
ping -Q 16; iperf3 -S 16; netperf -Y 16
```

  - RDMA applications:

    ```
    ucmatose -t 16; ib_write_bw -t 16
    ```

- For all RoCEv2 traffic on an interface: Use **default_roce_tos** in *configfs*.

  For example, set ToS 16 on device irdma0, port 1:

  ```
  # mkdir /sys/kernel/config/rdma_cm/irdma0
  # echo 16 > /sys/kernel/config/rdma_cm/irdma0/ports/1/default_roce_tos
  ```

# 5.0 Priority Flow Control - Planning and Guidelines

This section covers planning, considerations, and general configuration guidelines for enabling PFC on a network.

## 5.1 Steps

The steps for enabling PFC on your network include the following:

1. Set up your network hosts and switches. (Network Host and Switch Setup)
2. Decide whether to use willing or non-willing DCB mode on the800 Series adapter. (Willing vs. Non-willing DCB Mode)
3. Choose firmware or software DCB. (Firmware vs. Software DCB)
4. Decide how to separate and prioritize traffic streams. (Separating and Prioritizing Traffic Streams)
5. Configure ETS: Map priorities to traffic classes and allocate bandwidth. (Configuring ETS: Map Priorities to TCs/Allocate Bandwidth)
6. Configure PFC: Set priorities for drop or no-drop. (Configuring PFC)
7. Run your application with the right priority. (Run Applications with the Right Priority)

### 5.1.1 Network Host and Switch Setup

**NOTE**

PFC can be used with or without a switch in the network.

- If using a switch, you must configure PFC on both the adapter ports and the switch ports. Consult the appropriate switch manual for command syntax.
- If using adapters back-to-back, configure PFC on both hosts.

Host prerequisites for RDMA are outside the scope of this guide, but in general, you need at a minimum:

- Two Linux hosts with 800 Series adapters.
- Supporting 800 Series firmware and software (NVM with RDMA support, *ice* (Intel® Ethernet) driver, and *irdma* driver).

If using software in DCB mode, you also need **OpenLLDP**, which includes the **lldpad** daemon and **lldptool** configuration utility.

- In RHEL, install it with **yum** (**zypper** or **apt-get** might work as well in other operating systems.):

```
# yum install lldpad
```

- Alternatively, install from source from:
  https://github.com/intel/openlldp

### 5.1.2 Willing vs. Non-willing DCB Mode

DCB standards like PFC and ETS must be set to either willing or non-willing mode, which determines whether the port is willing to accept configuration settings from its link partner.

| Mode | When to Use |
|---|---|
| Willing | • If you want to configure DCB on their switch and let adapters accept settings from the switch ports.<br>This is the preferred, most common setup. |
| Non-willing | • For back-to-back configurations.<br>• For troubleshooting, testing, and manually tweaking the configuration.<br>• If preferred, configure DCB on all hosts and set the neighboring switch ports to willing (somewhat uncommon and might not be supported by all switches). |

### 5.1.3 Firmware vs. Software DCB

The 800 Series has two options for using DCB: firmware and software.

- Software DCB runs on the Linux host using **OpenLLDP**. It supports both willing and non-willing modes.
- Firmware DCB runs on the 800 Series adapter firmware. It only supports willing mode.

If you plan on using willing mode, software DCB is recommended but not required.

**NOTE**

Only one type of DCB might be active at a time. Enabling firmware DCB overrides the software DCB setting.

| DCB Type | When to Use | Willing Mode Setup | Non-willing Mode Setup |
|---|---|---|---|
| Firmware | Willing Mode | `# ethtool --set-priv-flags <iface> fw-lldp-agent on` | Not supported in firmware DCB. |
| Software | Willing Mode | Can be set up in IEEE or CEE modes.<br>Refer to Software DCB Willing Mode for details. | |
| Software | Non-willing Mode | `# ethtool --set-priv-flags <iface> fw-lldp-agent off`<br>`# lldptool -Ti <iface> -V PFC willing=yes`<br>`# lldptool -Ti <iface> -V ETS willing=yes` | `# ethtool --set-priv-flags <iface> fw-lldp-agent off`<br>`# lldptool -Ti <iface> -V PFC willing=no`<br>`# lldptool -Ti <iface> -V ETS willing=no` |

### 5.1.4 Software DCB Willing Mode

Software DCB can be configured in either IEEE or CEE mode.

#### For IEEE mode

1. Disable CEE transmission.

```
#lldptool -Ti $interface -V CEE-DCBX enableTx=no
```

2. Reset the DCBX mode to be *auto* (start in IEEE DCBX mode) after the next lldpad restart.

```
#lldptool -Ti $interface -V IEEE-DCBX mode=reset
```

3. Configure willing configuration for interface.

```
#lldptool -Ti $interface -V ETS-CFG enableTx=yes willing=yes
```

4. Configure willing recommendation for interface.

```
#lldptool -Ti $interface -V ETS-REC enableTx=yes
```

   Setting `willing=yes` for ETS-REC is not logical as it is by definition a recommendation for a willing link partner.

5. Configure willing PFC for interface.

```
#lldptool -Ti $interface -V PFC enable=yes willing=yes enableTx=yes
```

6. Terminate the first instance of lldpad that was started (e.g., from initrd). Once `lldpad -k` has been invoked and lldpad has been restarted, subsequent invocations of `lldpad -k` will not terminate lldpad.

```
#lldpad -k
```

7. Remove lldpad state records from shared memory.

```
#lldpad -s
```

8. Restart service lldpad.

```
#systemctl restart lldpad.service
```

9. Ensure **CEE mode** enableTx is set to **no**.

```
#lldptool -ti $interface -V CEE-DCBX -c
```

   Output:

```
enableTx=no
```

10. Ensure **DCBX mode** is set to **auto**.

```
#lldptool -ti $interface -V IEEE-DCBX -c
```

Output:

```
mode=auto
```

### For CEE mode

In CEE, successful negotiation requires the link partner also to be in CEE mode.

1. Enable CEE transmission.

```
#lldptool -T -i $interface -V CEE-DCBX enableTx=yes
```

2. Reset the DCBX mode to be *auto* (start in IEEE DCBX mode) after the next lldpad restart.

```
#lldptool -Ti $interface -V IEEE-DCBX mode=reset
```

3. To clean configuration of interface, set willing to off, disable priority group features, and set advertise to off.

```
#dcbtool sc $interface pg w:0 e:0 a:0
```

4. To clean configuration of interface, set willing to off, disable PFC features, and set advertise to off.

```
#dcbtool sc $interface pfc w:0 e:0 a:0
```

5. Configure willing, enable, and advertise configuration for priority group for interface.

```
#dcbtool sc $interface pg w:1 e:1 a:1
```

6. Configure willing, enable, and advertise configuration for PFC for interface.

```
#dcbtool sc $interface pfc w:1 e:1 a:1
```

7. Terminate the first instance of lldpad that was started (for example, from initrd).

   Once `lldpad -k` has been invoked and lldpad has been restarted, subsequent invocations of `lldpad -k` will not terminate lldpad.

```
#lldpad -k
```

8. Remove lldpad state records from shared memory.

```
#lldpad -s
```

9. Restart service lldpad.

```
#systemctl restart lldpad.service
```

10. Ensure **CEE mode** enableTx is set to **yes**.

```
#lldptool -ti $interface -V CEE-DCBX -c
```

Output:

```
enableTx=yes
```

11. Ensure **DCBX mode** is set to **cee**.

```
#lldptool -ti $interface -V IEEE-DCBX -c
```

Output

```
mode=cee
```

### 5.1.5 Separating and Prioritizing Traffic Streams

For networks carrying multiple traffic types, you typically want:

- One loss-less (no-drop) TC for RDMA traffic.
- One or more lossy (drop) TCs for LAN traffic.

This can change depending on specific applications.

Example configuration:

| Traffic Stream | Loss-less | TC | Priority | Bandwidth |
|---|---|---|---|---|
| RDMA Traffic | Yes | 0 | 0 | 50% |
| LAN Application #1 | No | 1 | 2 | 25% |
| LAN Application #2 | No | 2 | 4 | 25% |
| Unused | No | Any[1] | All Others[1] | None |

*Note:* 1. Unused priorities can be mapped to any TC (no traffic is being steered to specific priorities). Leaving them mapped to TC 0 is acceptable.

**NOTES**

- The 800 Series supports a maximum of four TCs per port, one of which can have PFC enabled.
- Traffic classes must start at zero and **must be contiguous (0, 1, 2, 3, … )**.
- ETS bandwidth allocations must total 100%.
- Multiple priorities might map to the same TC. For example, TC0 can contain prio=0,1,2,3,4,5,6,7. However, a given priority might map to only a single TC (like prio 0 cannot be in both TC0 and TC1).
- Linux PFC defines eight TCs, but if you are steering traffic using PCP PFC, there are only four priorities available: 0, 2, 4, and 6, which correspond to ToS 0, 8, 24, and 16, respectively.

### 5.1.6 Configuring ETS: Map Priorities to TCs/Allocate Bandwidth

The ETS component of DCB is responsible for mapping priorities to TCs and configuring bandwidth allocations.

**NOTE**

If your adapter is in non-willing mode, use **lldptool -V ETS-CFG**. Otherwise, configure these mappings on your switch.

| Mode | When to Use |
|---|---|
| Non-willing | Use ETS-CFG to set priority mappings (up2tc), transmission algorithm (tsa), and bandwidth (tcbw) for each traffic class. Continuing the previous example above, set mappings with:<br>`# lldptool -Ti <interface> -V ETS-CFG willing=no \`<br>`up2tc=0:0,1:0,2:1,3:0,4:2,5:0,6:0,7:0 \`<br>`tsa=0:ets,1:ets,2:ets,3:strict,4:strict,5:strict,6:strict,7:strict \`<br>`tcbw=50,25,25,0,0,0,0,0`<br>*Notes:* • If setting tcbw for a particular TC, also set tsa=ets for that TC.<br>• Configure the up2tc, tsa and tcbw options for ETS together. Although it's valid syntax to do them individually, the resulting configuration might not be valid.<br>• The other priorities (UPs 1, 3, 5, 6, and 7), which are unused in this example, also map to TC0. Although they are unused, they need to map to somewhere; TC0 is a conventional choice. |
| Willing | Consult the appropriate switch manual for QoS configurations. |

### 5.1.7 Configuring PFC

PFC can be configured either by setting the 3-bit VLAN based PCP or the 6-bit DCSP field.

#### 5.1.7.1 Option A: PCP PFC - Set Priorities for Drop or No-drop

If setting PCP PFC, the PFC component of DCB enables drop or no-drop settings for each priority.

- When PFC is enabled, the priority is considered no-drop, or loss-less.
- When PFC is disabled, the priority is considered drop, or lossy.

If an adapter is in non-willing mode, use **lldptool-V PFC**. Otherwise, configure these mappings on the switch.

| Mode | When to Use |
|---|---|
| Non-willing | Enable PFC on your lossless priorities (priority 0 in this example):<br>`# lldptool -Ti <interface> -V PFC willing=no enabled=0`<br>*Note:* On syntax<br>• enabled=0 means that PFC is enabled on Priority 0.<br>• enabled=0,1,2,3,4,5,6,7 enables PFC on all priorities.<br>• enabled=none disables PFC on all priorities |
| Willing | Consult the appropriate switch manual for QoS configurations. |

### 5.1.7.2 Option B: DSCP PFC

When configuring PFC using DSCP, set the traffic class to be used for the application using DSCP.

| Mode | When to Use |
|---|---|
| Non-willing | Enable DSCP (map DSCP value 24 to traffic class 3 in this example):<br>`# lldptool -Ti <interface> -V APP app=<prio>,<sel>,<pid>`<br>`# lldptool -Ti <interface> -V APP app=3,5,24`<br>Note on syntax:<br>• <prio>: TC assigned to the DSCP/ToS code point<br>• <sel>: is 5 for DSCP to TC mapping (see IEEE_8021QAZ_APP_SEL_DSCP symbol in Linux kernel)<br>• <pid>: DSCP/ToS code point |
| Willing | In DSCP mode, DCBx is disabled. |

## 5.1.8 Run Applications with the Right Priority

Set the ToS value in the application to steer it to the right traffic class.

**NOTE**

Command line options differ based on application.

Alternatively, for RoCEv2, you can set ToS for all traffic using the **default_roce_tos** parameter in *configfs*.

Following are some examples for setting ToS=24 (corresponding to prio 4):

LAN applications:

```
# ping -Q 24
# iperf3 -S 24
# netperf -Y 24
```

RDMA applications:

```
# ucmatose -t 24
# ib_write_bw -t 24
```

Set ToS for all RoCEv2 traffic: (device irdma0, port 1):

```
# mkdir /sys/kernel/config/rdma_cm/irdma0
# echo 24 > /sys/kernel/config/rdma_cm/irdma0/ports/1/default_roce_tos
```

**NOTE**

default_roce_tos in the case of DSCP is an 8-bit value derived from 6-bit DSCP value left-shifted by 2-bits. Hence default_roce_tos is 4 x DSCP value.

For example, a DSCP value of 24 (011 000) left-shifted by 2 bits gives an 8-bit ToS value of 96 (011 000 00) with a priority value of 3.

## 5.2 Example Configurations

### 5.2.1 Example 1 - 800 Series-800 Series Back-to-Back – PCP PFC with Single TC

A common benchmarking setup uses 800 Series adapters back-to-back, running a single traffic type.

**NOTE**

When using a single TC, that TC is always TC0. Application traffic runs on TC0 by default, unless explicitly configured otherwise.

Settings in this example:

- Non-willing mode — Configure both hosts in the same way to be compatible.
- Software DCB — Required to use non-willing mode.
- All priorities mapped to TC0.
- 100% bandwidth allocated to TC0.
- PFC enabled on priority 0 — in this configuration, enabling PFC on prio 0 essentially enables it for all traffic.

Perform the following steps on both servers:

1. Disable LFC (LFC and PFC cannot co-exist).

```
# ethtool -A <interface> rx off tx off
```

2. Verify that LFC is disabled.

```
# ethtool -a <interface>
Pause parameters for <interface>:
Autonegotiate:    on
RX:            off
TX:            off
RX negotiated: off
TX negotiated: off
```

3. Configure the adapter for software DCB mode by disabling firmware DCB mode.

   ```
   # ethtool --set-priv-flags <interface> fw-lldp-agent off
   ```

4. Verify that firmware DCB is disabled.

   ```
   # ethtool --show-priv-flags <interface> | grep fw-lldp-agent
   fw-lldp-agent         : off
   ```

5. Install **OpenLLDP** (the software that controls PFC and other DCB settings), if not already installed.
   - RHEL:

     ```
     # yum install lldpad
     ```

   - SLES or Ubuntu:

     ```
     zypper or apt-get might work (untested)
     ```

   - All operating systems:

     Download and build from source from https://github.com/intel/openlldp.

6. Start the LLDP daemon.

   ```
   # lldpad -d
   ```

7. Verify LLDP is active by showing current LLDP settings on the interface.

   These are the default **llpad** settings (some outputs might be different).

   ```
   Chassis ID TLV
           MAC: 68:05:ca:a3:89:78
   Port ID TLV
           MAC: 68:05:ca:a3:89:78
   Time to Live TLV
           120
   IEEE 8021QAZ ETS Configuration TLV
           Willing: yes
           CBS: not supported
           MAX_TCS: 8
           PRIO_MAP: 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
           TC Bandwidth: 0% 0% 0% 0% 0% 0% 0% 0%
           TSA_MAP: 0:strict 1:strict 2:strict 3:strict 4:strict 5:strict
   6:strict 7:strict
   IEEE 8021QAZ PFC TLV
           Willing: yes
           MACsec Bypass Capable: no
           PFC capable traffic classes: 8
           PFC enabled: none
   End of LLDPDU TLV
   ```

8. Enable PFC on priority 0 in non-willing mode.

   Note that `enabled=0` means that PFC is enabled on priority 0, not that PFC is not enabled.

   ```
   # lldptool -Ti <interface> -V PFC willing=no enabled=0
   ```

Output:

```
willing = no
prio = 0
```

9. Enable ETS to map all priorities to TC0 and allocate 100% bandwidth to TC0.

**NOTE**

The following is a single long command line:

```
# lldptool -Ti <interface> -V ETS-CFG willing=no \
up2tc=0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0 \
tsa=0:ets,1:strict,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict \
tcbw=100,0,0,0,0,0,0,0
```

Output:

```
willing = no
up2tc = 0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0
TSA = 0:ets 1:strict 2:strict 3:strict 4:strict 5:strict 6:strict 7:strict
tcbw = 100% 0% 0% 0% 0% 0% 0% 0%
```

10. Verify new settings.

```
# lldptool -ti <interface>
Chassis ID TLV
        MAC: 68:05:ca:a3:89:78
Port ID TLV
        MAC: 68:05:ca:a3:89:78
Time to Live TLV
        120
IEEE 8021QAZ ETS Configuration TLV
        Willing: no
        CBS: not supported
        MAX_TCS: 8
        PRIO_MAP: 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
        TC Bandwidth: 100% 0% 0% 0% 0% 0% 0% 0%
        TSA_MAP: 0:ets 1:strict 2:strict 3:strict 4:strict 5:strict 6:strict
7:strict
IEEE 8021QAZ PFC TLV
        Willing: no
        MACsec Bypass Capable: no
        PFC capable traffic classes: 8
        PFC enabled: 0
End of LLDPDU TLV
```

11. Repeat on other host.

    Alternatively, you could configure the other host for willing mode.

12. Run the application.

    Run the application normally on the parent interface.

    You do not need to specify any ToS or other priority values. The application runs on TC 0, with PFC enabled.

### 5.2.2 Example 2 - Adapters Connected Through a Switch – Willing Mode on Adapters, DCB Configured on Switch

A common DCB strategy in a cluster is to configure DCB on the switches, then configure the adapters for willing mode. When the willing adapters are connected to a switch with DCB configured, the adapters automatically apply the same DCB configuration.

**NOTE**

This example shows how to enable firmware willing mode on an 800 Series adapter. Consult the appropriate switch manual for DCB configuration steps.

Settings in this example:

- Willing mode — Adapters use DCB settings from the link partner.
- Software LLDP — Recommended for willing mode.

Perform the following steps on all servers:

1. Disable LFC (LFC and PFC cannot co-exist).

```
# ethtool -A <interface> rx off tx off
```

2. Verify that LFC is disabled.

```
# ethtool -a <interface>
Pause parameters for <interface>:
Autonegotiate:    on
RX:            off
TX:            off
RX negotiated: off
TX negotiated: off
```

3. Configure the adapter for firmware DCB mode (as opposed to software DCB mode.

```
# ethtool --set-priv-flags <interface> fw-lldp-agent on
```

4. Verify that firmware DCB is enabled.

```
# ethtool --show-priv-flags <interface> | grep fw-lldp-agent
fw-lldp-agent : on
```

5. Configure DCB on the switch.

   Consult the appropriate switch manual for DCB configuration. You can configure PFC, ETS, or any combination.

6. Optional. Verify that the adapter is using the correct settings.

   Syntax varies by switch. In general, enable DCBX on the switch, then show the current DCBX information. The switch can then report how the attached adapter is configured.

   For example, from an Arista 7060CX:

a. Enable DCBX in IEEE mode on port 21.

```
switch#enable
switch#configure
switch(config)#interface Ethernet 21/1
switch(config-if-Et21/1)#dcbx mode ieee
```

b. Show DCBX settings for port 21.

```
(config-if-Et21/1)#show dcbx Ethernet 21/1
Ethernet21/1:
IEEE DCBX is enabled and active
Last LLDPDU received on Fri Feb 14 15:42:09 2020
- PFC configuration: willing
capable of bypassing MACsec
supports PFC on up to 8 traffic classes
PFC not enabled on any priorities
- Application priority configuration:
1 application priorities configured:
ether 35078 priority 3
```

**NOTE**

This output shows a combination of the switch port's own settings and the link partner's settings. It indicates that:

- IEEE DCBX is active on the switch.
- At the given timestamp, the switch port received an LLDPDU message from the link partner describing the link partner's DCB settings.
- Everything after the Last LLDPU received line describes the contents of the received LLDPDU. These are the adapter’s settings.

### 5.2.3 Example 3 - DSCP PFC with Non-Default TCs - Non-Willing Mode on Adapters, ECN Configured

This benchmarking setup using DSCP based PFC to configure two priorities: one for data and the other for ECN packets

Settings in this example:

- Non-willing mode
- Software DCB — Required to use non-willing mode.
- Priority 3 with DSCP set to 24 mapped to TC1 for data
- Priority 6 with DSCP set to 48 mapped to TC1 for ECN

Perform the following steps on both servers:

1. Disable LFC (LFC and PFC cannot co-exist).

```
# ethtool -A <interface> rx off tx off
```

2. Verify that LFC is disabled.

```
# ethtool -a <interface>
Pause parameters for <interface>:
Autonegotiate:    on
```

```
RX:             off
TX:             off
RX negotiated: off
TX negotiated: off
```

3. Configure the adapter for software DCB mode by disabling firmware DCB mode.

```
# ethtool --set-priv-flags <interface> fw-lldp-agent off
```

4. Verify that firmware DCB is disabled.

```
# ethtool --show-priv-flags <interface> | grep fw-lldp-agent
fw-lldp-agent : off
```

5. Install **OpenLLDP** (the software that controls PFC and other DCB settings), if not already installed.

   - RHEL:

     ```
     # yum install lldpad
     ```

   - SLES or Ubuntu:

     ```
     zypper or apt-get might work (untested)
     ```

   - All operating systems:

     Download and build from source from https://github.com/intel/openlldp.

6. Start the LLDP daemon.

```
# lldpad -d
```

7. Verify LLDP is active by showing current LLDP settings on the interface.

   These are the default **llpad** settings (some outputs might be different).

```
# lldptool -ti <interface>
Chassis ID TLV
        MAC: 68:05:ca:a3:89:78
Port ID TLV
        MAC: 68:05:ca:a3:89:78
Time to Live TLV
        120
IEEE 8021QAZ ETS Configuration TLV
        Willing: yes
        CBS: not supported
        MAX_TCS: 8
        PRIO_MAP: 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
        TC Bandwidth: 0% 0% 0% 0% 0% 0% 0% 0%
        TSA_MAP: 0:strict 1:strict 2:strict 3:strict 4:strict 5:strict
6:strict 7:strict IEEE 8021QAZ PFC TLV
        Willing: yes
        MACsec Bypass Capable: no
        PFC capable traffic classes: 8
        PFC enabled: none
End of LLDPDU TLV
```

8. Plan your DSCP configuration.

| Traffic Stream | Lossless | DSCP | Prio | ToS | TC | Bandwidth |
|---|---|---|---|---|---|---|
| RDMA Application | Yes | 24 | 3 | 96 | 1 | 80% |
| ECN Congestion Notification | Yes | 48 | 6 | 192 | 1 | -- |
| General Traffic | Yes |  | Any |  | 0 | 20% |

Unused priorities can be mapped to any TC (no traffic is being steered to specific priorities). Leaving them mapped to TC 0 is acceptable.

9. Enable DSCP of 24 (Priority 3) on TC1 in non-willing mode.

```
# lldptool -Ti <interface> -V PFC willing=no enabled=3
```

Output:

```
willing = no
prio = 3
```

```
# lldptool -Ti <interface> -V APP app=1,5,24
```

Output:

```
0:(1,5,24) local hw (set)
```

**NOTE**

'5' is a fixed value indicating that this is setting DSCP to TC mapping.

10. Add DSCP map for ECN 48 on TC1 in non-willing mode.

```
# lldptool -Ti <interface> -V APP app=1,5,48
```

Output:

```
0:(1,5,48) local hw (set)
1:(1,5,24) local hw (set)
```

11. Enable ETS to map DSCP priority 3 and ECN to priority 6 on TC1 and allocate 20% bandwidth to TC0 and 80% to TC1.

**NOTE**

The following is a single long command line:

```
# lldptool -Ti <interface> -V ETS-CFG willing=no \
up2tc=0:0,1:0,2:0,3:1,4:0,5:0,6:1,7:0 \
tsa=0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict
tcbw=20,80,0,0,0,0,0,0
```

Output:

```
willing = no
up2tc = 0:0,1:0,2:0,3:1,4:0,5:0,6:1,7:0
TSA = 0:ets 1:ets 2:strict 3:strict 4:strict 5:strict 6:strict 7:strict
tcbw = 20% 80% 0% 0% 0% 0% 0% 0%
```

12. Verify new settings.

```
# lldptool -ti <interface>
Chassis ID TLV
        MAC: 68:05:ca:a3:89:78
Port ID TLV
        MAC: 68:05:ca:a3:89:78
Time to Live TLV
        120
IEEE 8021QAZ ETS Configuration TLV
        Willing: no
        CBS: not supported
        MAX_TCS: 8
        PRIO_MAP: 0:0 1:0 2:0 3:1 4:0 5:0 6:1 7:0
        TC Bandwidth: 20% 80% 0% 0% 0% 0% 0% 0%
        TSA_MAP:
0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict
IEEE 8021QAZ PFC TLV
        Willing: no
        MACsec Bypass Capable: no
        PFC capable traffic classes: 8
        PFC enabled: 3
End of LLDPDU TLV
```

13. Set the default_roce_tos.

    In this example, DSCP is set to 24 (0x18) and priority 3 so default_roce_tos is 96 (4 x DSCP value).

```
# mkdir /sys/kernel/config/rdma_cm/<interface>
# echo 96 > /sys/kernel/config/rdma_cm/<interface>/ports/1/default_roce_tos
# echo 0x706050418020100 > up_up_map
```

14. Configure ECN and add DSCP mapping for ECN.

    In this example, ECN has a DSCP value of 48.

```
# echo 1 > roce_dcqcn_enable
# echo 48 > cnp_up_override
# echo 1 > up_map_enable
```

15. Repeat on other node.
16. Run the application on the configured ports.

### 5.2.4 Example 4 - PCP PFC with Multiple TCs (1 for RDMA, 1 for LAN) – No VLANs

This example describes how to run both RDMA and LAN traffic on the same link using the parent interface (no explicit VLANs, although VLAN 0 are used transparently).

These steps can be used in a back-to-back configuration, or if you are using a switch, be sure to configure the neighboring switch ports for the same configuration (consult the appropriate switch manual for more detail).

Settings in this example:

- Non-willing mode — In this example, adapter settings are configured explicitly using **lldptool** (vs. configuring DCB on a switch and using willing mode on adapters).
- Software DCB — Required to use non-willing mode.
- Two traffic classes:
  - One loss-less TC for RDMA, with 80% bandwidth allocated.
  - One lossy TC for LAN, with 20% bandwidth allocated.
- PFC enabled for only the RDMA traffic class (this makes it loss-less).

Perform the following steps on both servers:

1. Disable LFC (LFC and PFC cannot co-exist).

```
# ethtool -A <interface> rx off tx off
```

2. Verify that LFC is disabled.

```
# ethtool -a <interface>
Pause parameters for <interface>:
Autonegotiate:    on
RX:            off
TX:            off
RX negotiated: off
TX negotiated: off
```

3. Configure the adapter for software DCB mode by disabling firmware DCB mode.

```
# ethtool --set-priv-flags <interface> fw-lldp-agent off
```

4. Verify that firmware DCB is disabled:

```
# ethtool --show-priv-flags <interface> | grep fw-lldp-agent
fw-lldp-agent : off
```

5. Install **OpenLLDP** (the software that controls PFC and other DCB settings), if not already installed:
   - RHEL:

```
# yum install lldpad
```

- SLES or Ubuntu:

  ```
  zypper or apt-get might work (untested)
  ```

- All operating systems:

  Download and build from source from https://github.com/intel/openlldp.

6. Start the LLDP daemon.

   ```
   # lldpad -d
   ```

7. Verify LLDP is active by showing current LLDP settings on the interface.

   The following example shows the **OpenLLDP** default:

   ```
   # lldptool -ti <interface>
   Chassis ID TLV
           MAC: 68:05:ca:a3:89:78
   Port ID TLV
           MAC: 68:05:ca:a3:89:78
   Time to Live TLV
           120
   IEEE 8021QAZ ETS Configuration TLV
           Willing: yes
           CBS: not supported
           MAX_TCS: 8
           PRIO_MAP: 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
           TC Bandwidth: 0% 0% 0% 0% 0% 0% 0% 0%
           TSA_MAP: 0:strict 1:strict 2:strict 3:strict 4:strict 5:strict
   6:strict 7:strict
   IEEE 8021QAZ PFC TLV
           Willing: yes
           MACsec Bypass Capable: no
           PFC capable traffic classes: 8
           PFC enabled: none
   End of LLDPDU TLV
   ```

8. Plan your DCB configuration.

| Traffic Stream | Loss-less | TC | Priority | ToS | Bandwidth |
|---|---|---|---|---|---|
| RDMA Application | Yes | 0 | 0 | 0 | 80% |
| LAN Application | No | 1 | 4 | 24 | 20% |
| Unused | No | Any[8.a] | All Others | N/A | 0% |

*Note:* a. Unused priorities can be mapped to any TC (no traffic is being steered to specific priorities). Leaving them mapped to TC 0 is acceptable.

9. Configure ETS.
   - Map priorities to traffic classes.
   - Allocate bandwidths.

**NOTE**

The following is a single long command line:

```
# lldptool -Ti <interface> -V ETS-CFG willing=no \
up2tc=0:0,1:0,2:0,3:0,4:1,5:0,6:0,7:0 \
tsa=0:ets,1:ets,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict \
tcbw=80,20,0,0,0,0,0,0
```

Output:

```
willing = no
up2tc = 0:0,1:0,2:0,3:0,4:1,5:0,6:0,7:0
TSA = 0:ets 1:ets 2:strict 3:strict 4:strict 5:strict 6:strict 7:strict
tcbw = 80% 20% 0% 0% 0% 0% 0% 0%
```

10. Enable PFC on priority 0 (non-willing).

```
# lldptool -Ti <interface> -V PFC willing=no enabled=0
```

Output:

```
willing = no
prio = 0
```

11. Verify new settings.

```
# lldptool -ti <interface>
Chassis ID TLV
        MAC: 68:05:ca:a3:89:78
Port ID TLV
        MAC: 68:05:ca:a3:89:78
Time to Live TLV
        120
IEEE 8021QAZ ETS Configuration TLV
        Willing: no
        CBS: not supported
        MAX_TCS: 8
        PRIO_MAP: 0:0 1:0 2:0 3:0 4:1 5:0 6:0 7:0
        TC Bandwidth: 80% 20% 0% 0% 0% 0% 0% 0%
        TSA_MAP: 0:ets 1:ets 2:strict 3:strict 4:strict 5:strict 6:strict
7:strict
IEEE 8021QAZ PFC TLV
        Willing: no
        MACsec Bypass Capable: no
        PFC capable traffic classes: 8
        PFC enabled: 0 ← This means PFC is enabled on prio 0 (not that PFC is
disabled)
End of LLDPDU TLV
```

12. Repeat on neighbor node:
    - If using a back-to-back configuration, either repeat the configuration on the other host or enable willing mode on that host.
    - If using a switch, configure the same DCB scheme on the switch port. Consult the appropriate switch manual for details.
13. Run the application.
    - RDMA:

Run the RDMA application normally. Since RDMA is running on the default TC0 with prio 0 in this example, you do not need any command line options to set ToS. ToS (and therefore priority) is 0 by default.

- LAN:

  Run the LAN application with a command line option to set ToS=24. In Linux, this maps to prio=4.

### 5.2.5 Example 5 - PCP PFC with Multiple TCs (1 for RDMA, 1 for LAN) – with VLANs

This example describes how to run both RDMA and LAN traffic on the same link using VLANs.

These steps can be used in a back-to-back configuration, or if you are using a switch, be sure to configure the neighboring switch ports for the same configuration (consult the appropriate switch manual for more detail).

Settings in this example:

- Non-willing mode — In this example, adapter settings are configured explicitly using **lldptool** (vs. configuring DCB on a switch and using willing mode on adapters).
- Software DCB — Required to use non-willing mode.
- Three traffic classes:
  - 1 lossy TC for general LAN traffic on the parent interface.
  - 1 loss-less TC for RDMA traffic on VLAN 100.
  - 1 lossy TC for LAN traffic on VLAN 200.
- PFC enabled for only the RDMA traffic class (this makes it loss-less).

Perform the following steps on both servers:

1. Disable LFC (LFC and PFC cannot co-exist).

```
# ethtool -A <interface> rx off tx off
```

2. Verify that LFC is disabled.

```
# ethtool -a <interface>
Pause parameters for <interface>:
Autonegotiate:    on
RX:             off
TX:             off
RX negotiated: off
TX negotiated: off
```

3. Configure the adapter for software DCB mode by disabling firmware DCB mode.

```
# ethtool --set-priv-flags <interface> fw-lldp-agent off
```

4. Verify that firmware DCB is disabled.

```
# ethtool --show-priv-flags <interface> | grep fw-lldp-agent
fw-lldp-agent : off
```

5. Install **OpenLLDP** (the software that controls PFC and other DCB settings), if not already installed.
    - RHEL:

    ```
    # yum install lldpad
    ```

    - SLES or Ubuntu:

    ```
    zypper or apt-get might work (untested)
    ```

    - All operating systems:

      Download and build from source from https://github.com/intel/openlldp.
6. Start the LLDP daemon.

    ```
    # lldpad -d
    ```

7. Verify LLDP is active by showing current LLDP settings on the interface.

   The following example shows the **OpenLLDP** default:

    ```
    # lldptool -ti <interface>
    Chassis ID TLV
            MAC: 68:05:ca:a3:89:78
    Port ID TLV
            MAC: 68:05:ca:a3:89:78
    Time to Live TLV
            120
    IEEE 8021QAZ ETS Configuration TLV
            Willing: yes
            CBS: not supported
            MAX_TCS: 8
            PRIO_MAP: 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
            TC Bandwidth: 0% 0% 0% 0% 0% 0% 0% 0%
            TSA_MAP: 0:strict 1:strict 2:strict 3:strict 4:strict 5:strict
    6:strict 7:strict
    IEEE 8021QAZ PFC TLV
            Willing: yes
            MACsec Bypass Capable: no
            PFC capable traffic classes: 8
            PFC enabled: none
    End of LLDPDU TLV
    ```

8. Plan your DCB configuration
    - RDMA is loss-less (PFC enabled).
    - LAN traffic is lossy (PFC disabled).

| Traffic Stream | Loss-less | TC | Priority | ToS | Bandwidth | Interface |
|---|---|---|---|---|---|---|
| General Traffic | No | 0 | 0 | 0 | 10% | Parent |
| RDMA Application | Yes | 1 | 2 | 8 | 50% | VLAN 100 |
| LAN Application | No | 2 | 3 | 0[8.a] | 40% | VLAN 200 |
| Unused | No | Any[8.b] | All Others | N/A | 0% | N/A |

*Notes:* a. LAN traffic can set VLAN priority directly using egress-qos-map when configuring the interface, so ToS mappings are not required. RDMA traffic cannot use egress-qos-map because egress-qos-map is controlled by the kernel, and RDMA traffic bypasses the kernel.

b. Unused priorities can be mapped to any TC (no traffic is being steered to specific priorities). Leaving them mapped to TC 0 is acceptable.

9. Configure ETS.
   - Map priorities to traffic classes.
   - Allocate bandwidth.

**NOTE**

The following is a single long command line:

```
# lldptool -Ti <interface> -V ETS-CFG willing=no \
up2tc=0:0,1:0,2:1,3:2,4:0,5:0,6:0,7:0 \
tsa=0:ets,1:ets,2:ets,3:strict,4:strict,5:strict,6:strict,7:strict \
tcbw=10,50,40,0,0,0,0,0
```

Output:

```
willing = no
up2tc = 0:0,1:0,2:1,3:2,4:0,5:0,6:0,7:0
TSA = 0:ets 1:ets 2:ets 3:strict 4:strict 5:strict 6:strict 7:strict
tcbw = 10% 50% 40% 0% 0% 0% 0%
```

10. Enable PFC on Priority 2 (non-willing).

```
# lldptool -Ti <interface> -V PFC willing=no enabled=2
```

Output:

```
willing = no
prio = 2
```

11. Verify new settings.

```
# lldptool -ti <interface>
Chassis ID TLV
        MAC: 68:05:ca:a3:89:78
Port ID TLV
        MAC: 68:05:ca:a3:89:78
Time to Live TLV
        120
IEEE 8021QAZ ETS Configuration TLV
        Willing: no
        CBS: not supported
        MAX_TCS: 8
        PRIO_MAP: 0:0 1:0 2:1 3:2 4:0 5:0 6:0 7:0
        TC Bandwidth: 10% 50% 40% 0% 0% 0% 0% 0%
        TSA_MAP: 0:ets 1:ets 2:ets 3:strict 4:strict 5:strict 6:strict
7:strict
IEEE 8021QAZ PFC TLV
        Willing: no
        MACsec Bypass Capable: no
        PFC capable traffic classes: 8
        PFC enabled: 0x4 ← This is a mask. 0x4 = 0b0000_0100, meaning PFC is
enabled on TC 2.
End of LLDPDU TLV
```

12. Repeat DCB settings on the neighbor node:
    - If using a back-to-back configuration, either repeat the DCB configuration on the other host or enable willing mode on that host.

- If using a switch, configure the same DCB scheme on the switch port. Consult the appropriate switch manual for details.

13. Create VLAN 100 for RDMA traffic:
    a. Create VLAN 100 as a part of the parent interface (like parent interface eth0, with an IP Address of 192.168.0.3).

```
# ip link add eth0.100 link eth0 type vlan id 100
```

    b. Bring the new interface up.

```
# ip link set eth0.100 up
```

    c. Set the IP Address on the new interface. In the example address, the third octet is 100, same as the VLAN ID, but the values do not need to match. However, the VLAN IP Address does need to be in a different subnet than the parent address.

```
# ip address add dev eth0.100 192.168.100.3/24
```

14. Create VLAN 200 (for LAN traffic) withqos-egress-map set:
    a. Create VLAN 200 as a part of the same parent interface (still using eth0 in the example).
    b. Use egress-qos-map to map all VLAN 200 LAN traffic to priority 3 in the VLAN header (see man ip-link for documentation).

**NOTE**

The egress-qos-map method works only for LAN traffic and not RDMA traffic.

```
# ip link add eth0.200 link eth0 type vlan id 200 egress-qos-map 0:3 1:3
2:3 3:3 4:3 5:3 6:3 7:3
```

    c. Bring the new interface up.

```
# ip link set eth0.200 up
```

    d. Set the IP Address on the new interface. In the example address, the third octet is 200, like the VLAN ID, but it does not need to match.

```
# ip address add dev eth0.100 192.168.200.3/24
```

15. Verify new interfaces:
    a. Examine the output of ip link show and verify both new VLANs are up and have the right IP Address.

```
10: enp175s0f0.100@enp175s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc noqueue state UP group default qlen 1000
link/ether 68:05:ca:a3:89:78 brd ff:ff:ff:ff:ff:ff
inet 192.168.100.1/24 scope global enp175s0f0.100
valid_lft forever preferred_lft forever
inet6 fe80::6a05:caff:fea3:8978/64 scope link
valid_lft forever preferred_lft forever
12: enp175s0f0.200@enp175s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
```

```
qdisc noqueue state UP group default qlen 1000
link/ether 68:05:ca:a3:89:78 brd ff:ff:ff:ff:ff:ff
inet 192.168.200.3/24 scope global enp175s0f0.200
valid_lft forever preferred_lft forever
inet6 fe80::6a05:caff:fea3:8978/64 scope link
valid_lft forever preferred_lft forever
```

b. Verify egress mappings for VLAN 200 here.

```
# cat /proc/net/vlan/eth0.200 | grep EGRESS
EGRESS priority mappings: 0:3 1:3 2:3 3:3 4:3 5:3 6:3 7:3
```

16. Repeat VLAN settings on the neighbor node:

- If using a back-to-back configuration, configure the same VLANs on the other host.
- If using a switch, consult the appropriate switch manual for details.

Sample commands from an Arista 7060CX:

a. Create VLANs 100 and 200.

```
switch>enable
switch>config
switch(config)>vlan 100
switch(config)>vlan 200
```

b. Set the switch ports where adapters are connected to trunk mode. Example, for port 21/1.

```
switch(config)>#interface Et21/1
switch(config-if-Et21/1)#switchport mode trunk
```

c. Add the VLANs to the switch ports where adapters are connected.

```
switch(config-if-Et21/1)#switchport trunk allowed vlan 1,100,200
```

d. Show current VLANs (VLAN 1 always exists by default).

```
switch> show vlan
VLAN Name Status Ports
----- --------- ------ --------------
1 default active Et21/1, Et23/1
100 VLAN0100 active Et21/1, Et23/1
200 VLAN0200 active Et21/1, Et23/1
```

**NOTE**

If needed, undo settings, preface them with “no”.

- To delete a VLAN: switch(config)>no vlan 100
- To remove trunk mode: switch(config-if-Et23/1)#no switchport mode trunk

17. Run the applications.

| Traffic Stream | Interface | Example IP Address | TC | Priority | ToS | Set Application Priority |
|---|---|---|---|---|---|---|
| General Traffic | Parent | 192.168.0.1 | 0 | 0 | 0 | Run normally on 192.168.0.1, no ToS options needed. prio 0 and TC 0 are the defaults. |
| RDMA Application | VLAN100 | 192.168.100.1 | 1 | 2 | 8 | Run on 192.168.100.1 and set ToS=8 on the application command line.<br>Alternatively, if using RoCEv2:<br>Set default_roce_tos=8 (Ctrl-F this article for syntax). This sets ToS=8 for all RoCEv2 traffic, so you do not need the application command line option. |
| LAN Application | VLAN 200 | 192.168.200.1 | 2 | 3 | 0 | Run on 192.168.200.1 normally. No command line ToS options needed because egress-qos-map is set to use priority 3. |

# 6.0 Priority Flow Control - Verification

## 6.1 Priority Counters

Priority flow control counters for each interface are available in **ethtool**. They measure the number of Xon and Xoff (transmit on and off) frames sent and received by that interface.

To view them:

```
# ethtool -S <interface> | grep prio
```

Counters are named with tx/rx, priority number, and either xon or xoff.

For example:

```
# ethtool -S enp175s0f0 | grep prio tx_priority_0_xon.nic: 1
tx_priority_0_xoff.nic: 6434
tx_priority_1_xon.nic: 1
tx_priority_1_xoff.nic: 6434
tx_priority_2_xon.nic: 2
tx_priority_2_xoff.nic: 14864
tx_priority_3_xon.nic: 1
tx_priority_3_xoff.nic: 6434
tx_priority_4_xon.nic: 0
tx_priority_4_xoff.nic: 0
tx_priority_5_xon.nic: 1
tx_priority_5_xoff.nic: 6434
tx_priority_6_xon.nic: 1
tx_priority_6_xoff.nic: 6434
tx_priority_7_xon.nic: 1
tx_priority_7_xoff.nic: 6434
rx_priority_0_xon.nic: 0
rx_priority_0_xoff.nic: 0
rx_priority_1_xon.nic: 0
rx_priority_1_xoff.nic: 0
rx_priority_2_xon.nic: 0
rx_priority_2_xoff.nic: 0
rx_priority_3_xon.nic: 0
rx_priority_3_xoff.nic: 0
rx_priority_4_xon.nic: 0
rx_priority_4_xoff.nic: 0
rx_priority_5_xon.nic: 0
rx_priority_5_xoff.nic: 0
rx_priority_6_xon.nic: 0
rx_priority_6_xoff.nic: 0
rx_priority_7_xon.nic: 0
rx_priority_7_xoff.nic: 0
```

Note that the Rx counters all 0.

When adapters are connected through a switch, the rx_priority_* counters might be 0, indicating that the adapter has not received any pause frames from the switch. Depending on the level of stress in the network, this is acceptable if the switch has enough buffering to keep up with the host demand. However, for high stress traffic such as HPC applications at larger scale, often the switch sends pause frames to the host. In general, it is expected to see both tx and rx_priority counters.

Note that some of the Tx counters have the same value.

In the 800 Series QoS implementation, if PFC is enabled for any priority in a traffic class, all priorities in that traffic class get pause frames. This means that the counters for all priorities in the same TC are incremented in unison, regardless of the particular single priority that is causing PFC to trigger. If all priorities are mapped to the same TC, they all increment in unison.

This implementation is in line with 802.1Q recommendations.

- 802.1Q Section 37.3: NOTE 2 — All priorities within a traffic class typically share similar traffic handling requirements (e.g., loss and bandwidth).
- 802.1Q Section 8.6.8: NOTE 1 — Two or more priorities can be combined in a single queue. In this case if one or more of the priorities in the queue are paused, it is possible for frames in that queue not belonging to the paused priority to not be scheduled for transmission.
- 802.1Q Section 8.6.8: NOTE 2 — Mixing PFC and non-PFC priorities in the same queue results in non-PFC traffic being paused causing congestion spreading, and therefore is not recommended.

**TIP**

Run a **watch** command in a separate terminal window to see priority counters moving in real time:

```
# watch -d -n 1 "ethtool -S <interface> | grep prio"
```

## 6.2 Discard Counters

Enabling flow control should eliminate drops and discards in the network.

### 6.2.1 LAN Packet Drops

```
# ethtool -S enp175s0f0 | grep drop
     rx_dropped: 0
     tx_dropped_link_down.nic: 0
     rx_dropped.nic: 0
```

### 6.2.2 RDMA Discards

```
# cd /sys/class/infiniband/irdma-enp175s0f0/hw_counters
# for f in *Discards; do echo -n "$f: "; cat "$f"; done
ip4InDiscards: 0
ip6InDiscards: 0
```

If you see non-zero counters for packet discards, double check that both tx and rx_priority counters are being sent and received by the NIC as described in Priority Counters. If any are zero, PFC might not be fully enabled.

## 6.3 tpcdump

**tcpdump** can be used to verify ToS, DSCP, PCP, or VLAN ID values.

RDMA traffic requires switch port mirroring or an inline protocol analyzer to capture RDMA traffic.

To find each value in tcpdump:

| Value | tcpdump Option | tcpdump Output |
|---|---|---|
| ToS | -v | Run **ping** with -Q 24 to set ToS=24 (0x18):<br>`# ping -I enp175s0f0 -Q 24 192.168.0.3`<br>Use **tcpdump** with **-v** and look for tos 0x18 in the IP header:<br>`# tcpdump -nXX -v -i enp175s0f0`<br>`15:05:53.197406 IP (tos 0x18, ttl 64, id 20320, offset 0, flags [DF], proto ICMP (1), length 84)`<br>`192.168.0.1 > 192.168.0.3: ICMP echo request, id 14718, seq 3, length 64`<br>`0x0000: 6805 caa3 8778 6805 caa3 8978 0800 4518 h....xh....x..E.`<br>``0x0010: 0054 4f60 4000 4001 69dc c0a8 0001 c0a8 .TO`@.@.i.......``<br>`0x0020: 0003 0800 d918 397e 0003 312f 585e 0000 ......9~..1/X^..`<br>`0x0030: 0000 9a05 0300 0000 0000 1011 1213 1415 ................`<br>`0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425 ..........!"#$%`<br>`0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 &'()*+,-./012345`<br>`0x0060: 3637` |
| DSCP | -v | DSCP is not shown explicitly in tcpdump. DSCP is based on ToS value.<br>Example: ToS 0x18 maps to DSCP 0x6.<br>1. Start with ToS 0x18.<br>2. Convert 0x18 from hex to decimal.<br>0x18 = 0b0001_1000<br>3. Look at the upper 6 bits of the ToS field to find DSCP.<br>0x18 = 0b**000110**00<br>0b**000110** = 0x06<br>4. DSCP = 0x6 |
| VLAN ID and VLAN priority (PCP) | -e vlan | Run ping on VLAN 200 with -Q 24 to set ToS=24 (0x18):<br>`# ping -I enp175s0f0.200 -Q 24 192.168.200.3`<br>Run **tcpdump** with **-e vlan** and look for vlan 200 and p 4 in the VLAN header:<br>`# tcpdump -nXX -v -i enp175s0f0 -e vlan`<br>`15:23:37.554911 68:05:ca:a3:89:78 > 68:05:ca:a3:87:78, ethertype 802.1Q (0x8100), length 102: vlan 200, p 4, ethertype IPv4, (tos 0x18, ttl 64, id 53320, offset 0, flags [DF], proto ICMP (1), length 84)`<br>`192.168.200.1 > 192.168.200.3: ICMP echo request, id 15739, seq 3, length 64`<br>`0x0000: 6805 caa3 8778 6805 caa3 8978 8100 80c8 h....xh....x.... # PCP is 3 bits: 0x8 → 0b1000 → 4`<br>`0x0010: 0800 4518 0054 d048 4000 4001 58ea c0a8 ..E..T.H@.@.X...`<br>`0x0020: c801 c0a8 c803 0800 82a4 3d7b 0003 5933 ..........={..Y3`<br>`0x0030: 585e 0000 0000 bf78 0800 0000 0000 1011 X^.....x........`<br>`0x0040: 1213 1415 1617 1819 1a1b 1c1d 1e1f 2021 ...............!`<br>`0x0050: 2223 2425 2627 2829 2a2b 2c2d 2e2f 3031 "#$%&'()*+,-./01`<br>`0x0060: 3233 3435 3637` |

# 7.0 Troubleshooting

| Problem | Solution |
|---|---|
| "Agent instance for device not found 00007f" when running **lldptool**. | "ifup" your netdev interface. |
| "Failed to connect to lldpad - clif_open: Connection refused" when running **lldptool**. | Make sure **lldpad** is running.<br>`ps -ef \| grep lldpad)` |
| "Invalid command 00007f" when running **lldptool** | Check your command syntax and arguments. Possible missing argument or bad values. |
| "kernel: ice 0000:af:00.0: Set DCB Config failed" in the system log or **dmesg**. | Likely an invalid DCB configuration:<br>• Verify in ETS that TCs are contiguous and start at TC0.<br>• Verify in ETS that bandwidth allocations total 100%.<br>• Verify that you are not trying to allocate bandwidth to TCs that do not exist in up2tc.<br>If necessary, reset the **OpenLLDP** configuration to the default by removing the configuration file.<br>`# killall lldpad && rm /var/lib/lldpad/lldpad.conf`<br>When you restart **lldpad**, it regenerated the default configuration file. |
| "[4353.872101] ice 0000:af:00.0: application TOS[0] and remote client TOS[24] mismatch" in the system log or **dmesg**. | This happens because an RDMA application with, for example,16 QP with ToS 24 might still create a 17th QP for application setup using ToS 0. This is a quirk of the application and not controlled by the 800 Series *ice* or *irdma* drivers. |
| **lldpad** or **llpdtool** problems. | Enable verbose lldpad debug logging:<br>1. Edit */usr/lib/systemd/system/lldpad.service*.<br>`ExecStart=/usr/sbin/lldpad -t -V9`<br>2. Save and reload**lldpad** service.<br>`systemctl daemon-reload && systemctl restart lldpad`<br>3. Save log to a file.<br>`journalctl -u lldpad -f \| tee lldpad.log` |