SOFTWARE MANUAL

Mellanox OFED for FreeBSD for ConnectX-4 and above Adapter Cards

User Manual

Rev 3.5.2

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT (“PRODUCT(S)”) AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES “AS-IS” WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Table of Contents

List of Tables

Document Revision History

Table 1 - Document Revision History

Revision	Date	Description

3.5.2	September 29, 2019	• Updated the following sections:
		  • Section 2.2, "Installing Mellanox Driver for FreeBSD", on page 12
		• Added the following sections:
		  • Section 3.6.9, "Priority Flow Control (PFC) Hardware Buffer Configuration", on page 34
		  • Section 3.9, "Forward Error Correction (FEC)", on page 36
		  • Section 3.10, "IRQ Labeling", on page 36
		  • Section 3.11, "Port Module Events", on page 37

3.5.1	May 29, 2019	Updated instructions under Section 2.3.3.3, "Updating Firmware Using Kernel Module", on page 13.

	May 2, 2019	• Updated the following sections:
		  • Section 3.6.5, "Mapping User Priority to Traffic Class", on page 32
		  • Section 2.3.3, "Updating Firmware", on page 13
		  • Section 4.1, "Receive Queue Interrupt Moderation", on page 39
		• Added the following sections:
		  • Section 2.3.3.2, "Updating Firmware Using mlx5tool", on page 13
		  • Section 2.3.3.3, "Updating Firmware Using Kernel Module", on page 13
		  • Section A.1.1, "General", on page 43

3.5.0	January 7, 2019	• Added the following missing Ethernet counters to Table 8, "Configuration Options - Ethernet," on page 43:
		  • dev.<ifnet_name>.vstats.tx_jumbo_packets
		  • dev.<ifnet_name>.rxstat0.bytes
		  • dev.<ifnet_name>.txstat0tc0.bytes

	November 29, 2018	• Added Section 3.6.8, "Enhanced Transmission Selection (ETS)", on page 34. • Updated Section 3.6.7, "Maximum Rate Limiting", on page 33.

Table 1 - Document Revision History

Revision	Date	Description

3.4.2	July 18, 2018	• Added the following:
		  • Section 3.4.3, "RoCE Packet Sniffing", on page 28
		  • Section 3.4.4, "Perftest", on page 28
		  • Section 3.5.1, "Global Pause", on page 29
		  • Section 3.6.1.1, "VLAN 0 Priority Tagging", on page 31
		  • Section 3.6.2, "Differentiated Service Code Point (DSCP)", on page 32
		  • Section 3.6.3, "Trust State", on page 32
		  • Section 3.6.4, "QoS with RDMA", on page 32
		  • Section 3.6.6, "Mapping DSCP to Priority Mapping", on page 33

3.4.1	March 1, 2018	• Added the following:
		  • Appendix A: "Sysctl Configuration and Counters," on page 43
		  • Section 3.5.4, "Explicit Congestion Notification (ECN)", on page 30
		  • Section 3.5.2, "Priority Flow Control (PFC)", on page 29
		  • Section 3.6, "Quality of Service", on page 31
		  • Section 3.7, "Rx Hardware Time-Stamping", on page 35
		  • Section 3.8, "Firmware Dump", on page 35
		• Updated the following:
		  • Section 3.4, "RDMA over Converged Ethernet (RoCE)", on page 25

3.4.0	July 2017	Added the following section: • Section 3.4, "RDMA over Converged Ethernet (RoCE)", on page 25
3.3.0	November 2016	Added the following section: • Section 3.3, "Packet Pacing", on page 21
3.0.0	November 2015	Initial release

About this Manual

This Preface provides general information concerning the scope and organization of this User's Manual.

Intended Audience

This manual is intended for system administrators responsible for the installation, configuration, management and maintenance of the software and hardware of VPI (Infiniband, Ethernet) in ETH mode adapter cards.

Common Abbreviations and Acronyms

Table 2 - Abbreviations and Acronyms

Abbreviation / Acronym	Whole Word / Description
B	(Capital) 'B' is used to indicate size in bytes or multiples of bytes (e.g., 1KB = 1024 bytes, and 1MB = 1048576 bytes)
b	(Small) 'b' is used to indicate size in bits or multiples of bits (e.g., 1Kb = 1024 bits)
FW	Firmware
HCA	Host Channel Adapter
HW	Hardware
IB	InfiniBand
LSB	Least significant *byte*
lsb	Least significant *bit*
MSB	Most significant *byte*
msb	Most significant *bit*
NIC	Network Interface Card
SW	Software
VPI	Virtual Protocol Interconnect
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet
SL	Service Level
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane

Support and Updates Webpage

Please visit http://www.mellanox.com > Products > Software > Ethernet Drivers > FreeBSD Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.

1 Overview

This document provides information on the Mellanox driver for FreeBSD and instructions for installing the driver on Mellanox ConnectX® adapter cards supporting the following uplinks to servers:

Table 3 - Supported Uplinks to Servers

HCA	Uplink Speed
ConnectX®-4	• Ethernet: 10GigE, 25GigE, 40GigE, 50GigE and 100GigE • InfiniBand: SDR, QDR, FDR, FDR10, EDR
ConnectX®-4 Lx	• Ethernet: 10GigE, 25GigE, 40GigE and 50GigE
ConnectX®-5/ ConnectX®-5 Ex	• Ethernet: 10GigE, 25GigE, 40GigE, 50GigE and 100GigE • InfiniBand: SDR, QDR, FDR, FDR10, EDR
ConnectX®-6	• InfiniBand: SDR, EDR, HDR • Ethernet: 10GigE, 25GigE, 40GigE, 50GigE, 100GigE[a] and 200GigE (**alpha**)

a. ConnectX-6 Ethernet adapter cards currently support Force mode only. Auto-Negotiation mode is not supported.

The driver release introduces the following capabilities:

- Single/Dual port
- Number of RX queues per port - according to number of CPUs
- Number of TX queues per port - according to number of CPUs
- MSI-X or INTx
- Hardware Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- VLAN Tx/Rx acceleration (Hardware VLAN stripping/insertion)
- ifnet statistics

1.1 Mellanox OFED for FreeBSD Package Contents

1.1.1 Tarball Package

Mellanox OFED for FreeBSD package includes the following directories:

- sys - kernel space
- contrib - user space
- user.sbin - mlx5tool

1.1.2 mlx5 driver

mlx5 is the low level driver implementation for the ConnectX-4 and above adapter cards designed by Mellanox Technologies.

1.1.2.1 Software Components

Mellanox OFED for FreeBSD contains the following software components:

Table 4 - Mellanox OFED for FreeBSD Software Components

Components	Description
mlx5ib	Implementation of ibcore interface to support RoCE and InfiniBand in ConnectX-4/ConnectX-5 adapter cards.
mlx5	Acts as a library of common functions required by ConnectX®-4/ConnectX-4 Lx adapter cards. For example: initializing the device after reset.
mlx5en	Handles Ethernet specific functions and plugs into the ifnet mid-layer.
Documentation	Release Notes, User Manual

2 Installation

This chapter describes how to install and test the Mellanox driver for FreeBSD package on a single host machine with Mellanox adapter hardware installed.

2.1 Software Dependencies

- To install the driver software, kernel sources must be installed on the machine.
- To load mlx5, linuxkpi must be loaded as well.
  - Compile and install linuxkpi module under /sys/modules/linuxkpi.

2.2 Installing Mellanox Driver for FreeBSD

2.2.1 Installing Mellanox EN Driver for FreeBSD

FreeBSD driver is no longer provided as a tarball. The EN driver can be obtained as part of the Kernel used. In order to load it, please run the following command:

```
#kldload mlx5en
```

The dependent mlx5 and linuxkpi modules will be loaded automatically.

2.2.2 Installing Mellanox IB/RDMA Driver for FreeBSD

➢ ***To load the IB/RDMA driver, follow the steps below.***

Get into the FreeBSD SRC tree (/usr/src).

```
# make buildworld WITH_OFED=YES
# make kernel WITH_OFED=YES
# make installworld WITH_OFED=YES
```

- Driver related code located in:

```
sys/compat/linuxkpi/
sys/dev/mlx5/
```

- Userspace related code located in

```
contrib/ofed/
usr.sbin/mlx5tool/
```

2.3 Firmware Programming

The adapter card was shipped with the most current firmware available. This section is intended for future firmware upgrades, and provides instructions for (1) installing Mellanox firmware update tools (MFT), (2) downloading FW, and (3) updating adapter card firmware.

2.3.1 Installing Firmware Tools

Step 1. Download the current Mellanox Firmware Tools package (MFT) from www.mellanox.com > Products > Adapter IB/VPI SW > Firmware Tools.
The tools package to download is "MFT_SW for FreeBSD" (tarball name is mft-X.X.X.tgz). For ConnectX®-4, you will need at least MFT-4.1.X.X.X.

Step 2. Extract the tarball and run the installation script.

2.3.2 Downloading Firmware

1. Retrieve device's PCI slot (i.e. pci0:x:0:0). Run:

```
#> mst status
```

2. Verify your card's PSID.

```
#> flint -d pci0:<x>:0:0 q
```

3. Download the desired firmware from the Mellanox website.
   http://www.mellanox.com/page/firmware_download

2.3.3 Updating Firmware

2.3.3.1 Updating Firmware Using flint

1. Before burning a new firmware, make sure the modules are unloaded. To unload the modules, run:

```
#> kldunload mlx5en
#> kldunload mlx5ib
#> kldunload mlx5
#> kldstat | grep mlx5
```

2. Unzip the firmware binary file.
3. Burn the firmware on your server:

```
$flint -d pci0:<x>:0:0 -i <img.bin> b
```

4. Reboot the server.

2.3.3.2 Updating Firmware Using mlx5tool

Please note that mlx5tool can only use images of MFA2 format.

Run the following command:

```
mlx5tool -d domain:bus:slot:func -f file.mfa2
```

where:

-f Flashes the firmware image file.mfa2 to the specified adapter.
Image must be in MFA2 format and must contain a component suitable for the adapter hardware.

2.3.3.3 Updating Firmware Using Kernel Module

1. Create a Makefile in sys/modules/mlx5mfa/Makefile directory structure as follows:

```
$ cat sys/modules/mlx5mfa/Makefile
# $FreeBSD$
KMOD=   mlx5fw_mfa
FIRMWS= mlx5fw.mfa2:${KMOD}
.include <bsd.kmod.mk>
```

2. Place the MFA2 file in the same directory.
3. Build and install the Kernel module.
4. Enable auto firmware update:

```
kenv hw.mlx5.auto_fw_update=1
```

Make sure to disable auto firmware update before rebooting/resetting the firmware.

2.3.4 Setting the Ports to ETH

If you have a VPI HCA, you will need to set the ports to ETH. This is done by using the mlxconfig tool (part of the MFT).

1. If you have a card with two ports, run:

```
#> mlxconfig -d pci0:<x>:0:0 set  LINK_TYPE_P1=2 (For the first port)
#> mlxconfig -d pci0:<x>:0:0 set  LINK_TYPE_P2=2 (For the second port)
```

2. Reboot the server.

2.4 Driver Usage and Configuration

Interface name has changed from mlx5en to mce. Note that ifconfig and sysctl commands were updated accordingly.

➢ ***To assign an IP address to the interface:***

```
#> ifconfig mce<N> <ip>
```

Note: <N> is the OS assigned interface number

➢ ***To check driver and device information:***

```
#> pciconf -lv | grep mlx
#> flint -d pci0:<x>:0:0 q
```

Example:

```
#> flint -d pci0:6:0:0 dc | grep Description
```

Example:

```
#> pciconf -lv  | grep mlx -C 3
mlx5_core0@pci0:33:0:0:     class=0x020000 card=0x001415b3 chip=0x101315b3 rev=0x00 hdr=0x00
    vendor     = 'Mellanox Technologies'
    device     = 'MT27620 Family'
    class      = network
    subclass   = ethernet
mlx5_core1@pci0:33:0:1:     class=0x020000 card=0x001415b3 chip=0x101315b3 rev=0x00 hdr=0x00
    vendor     = 'Mellanox Technologies'
    device     = 'MT27620 Family'
    class      = network
#> flint -d pci0:33:0:0: q
Image type:      FS3
FW Version:      12.12.0610
FW Release Date: 3.9.2015
Description:     UID                GuidsNumber
Base GUID:       e41d2d03006094ec        20
Base MAC:        0000e41d2d6094ec        20
Image VSD:
Device VSD:
PSID:             MT_2190110032
#> flint -d pci0:6:0:0 dc | grep Description
;;Description = ConnectX-4 VPI adapter card; EDR IB (100Gb/s) and 100GbE; dual-port QSFP28;
PCIe3.0 x16; ROHS R6
```

➢ ***To check driver version:***

```
#>sysctl -a
```

Example:

```
sysctl -a | grep Mellanox
dev.mlx5_core.1.%desc: Mellanox Ethernet driver (3.0.0-RC2)
dev.mlx5_core.0.%desc: Mellanox Ethernet driver (3.0.0-RC2)
```

➢ ***To check firmware version:***

- dmesg

  ```
  #> dmesg
  ```

  Example:

  ```
  Mlx5_core0: INFO: firmware version: 12.12.2008
  ```

- sysctl

  ```
  #> sysctl -a
  ```

  Example:

  ```
  dev.mlx5_core.0.hw.fw_version: 12.12.2008
  ```

➢ ***To query stateless offload status:***

```
#> ifconfig mce<x>
```

Note: <x> is the OS assigned interface number

➢ ***To set stateless offload status:***

```
#> ifconfig mce<x> [rxcsum|-rxcsum] [txcsum|-txcsum] [tso|-tso] [lro|-lro]
```

Note: <x> is the OS assigned interface number

➢ ***To query and set interrupt coalescing modes:***

```
#> sysctl -a | grep coalesce_mode
```

Example:

```
#> sysctl -a | grep coalesce_mode
dev.mce.0.conf.rx_coalesce_mode: 1
dev.mce.1.conf.rx_coalesce_mode: 1
```

- coalesce mode '0' indicates interrupt timer is resetting with each interrupt event.
- coalesce mode '1' indicates interrupt timer is resetting with each received packet.

➢ ***To query and modify values for timer initialization between interrupts:***

```
#> sysctl -a |  grep tx_coalesce_usecs
#> sysctl -a |  grep rx_coalesce_usecs
```

➢ ***To query and modify values for number of received packets between interrupts:***

```
#> sysctl -a | grep tx_coalesce_pkts
#> sysctl -a | grep rx_coalesce_pkts
```

Example:

```
#> sysctl -a | grep rx_coalesce_usecs
dev.mce.1.conf.rx_coalesce_usecs: 3
dev.mce.0.conf.rx_coalesce_usecs: 3
#> sysctl -a | grep rx_coalesce_pkts
dev.mce.1.conf.rx_coalesce_pkts: 32
dev.mce.0.conf.rx_coalesce_pkts: 32
```

➢ ***To query ring size values:***

```
#> sysctl -a | grep mce| grep _size
```

Example:

```
#> sysctl -a | grep mlx | grep _size
dev.mce.1.conf.rx_queue_size: 1024
dev.mce.1.conf.tx_queue_size: 1024
dev.mce.1.conf.rx_queue_size_max: 8192
dev.mce.1.conf.tx_queue_size_max: 8192
```

➢ ***To modify rings size:***

```
#> sysctl dev.mce.<N>.conf.rx_queue_size=[N]
#> sysctl dev.mce.<N>.conf.tx_queue_size=[N]
```

Note: <x> is the OS assigned interface number

➢ ***To obtain device statistics:***

```
#> sysctl -a | grep mce | grep stat
```

- ***To obtain additional device statistics:***

```
#> sysctl dev.mce.<N>.conf.debug_stats=1
#> sysctl -a | grep mce | grep stats
```

- ***To show out of receive buffers counter:***

```
sysctl dev.mce.<N>.vstats.rx_out_of_buffer
dev.mce.<N>.vstats.rx_out_of_buffer: 0
```

- ***To verify support for Rx/Tx pause frames:***
  - ifconfig

```
#> ifconfig
media: Ethernet autoselect (100GBase-CR4 <full-duplex,rxpause,txpause>)
```

  - sysctl

```
#> sysctl dev.mce.<N>.rx_pauseframe_control
dev.mce.<N>.rx_pauseframe_control: 1
#> sysctl dev.mce.<N>.tx_pauseframe_control
dev.mce.<N>.tx_pauseframe_control: 1
```

- ***To enable/disable Rx/Tx pause frames:***

```
sysctl dev.mce.<N>.rx_pauseframe_control=1
sysctl dev.mce.<N>.tx_pauseframe_control=1
```

Note: 0 = disable, 1 = enable

- ***To show all supported media:***

```
#> ifconfig -m mce<x>
supported media:
 media autoselect
 media 50GBase-CR2 mediaopt full-duplex
 media 25GBase-SR mediaopt full-duplex
 media 25GBase-CR mediaopt full-duplex
 media 100GBase-LR4 mediaopt full-duplex
 media 100GBase-SR4 mediaopt full-duplex
 media 100GBase-CR4 mediaopt full-duplex
 media 40Gbase-LR4 mediaopt full-duplex
```

Note: <x> is the OS assigned interface number

The list of supported media is different in ConnectX-4 and ConnectX-4 Lx.

- ***To set new media:***

```
#> ifconfig -m mce<x> media <y> mediaopt full-duplex
```

Note: <x> is the OS assigned interface number. <y> is the relevant media

When updating the media, make sure to choose the right cable type.

Once the driver is loaded, both ports will be activated, meaning that an ifnet will be created for each port.

3 Features Overview and Configuration

3.1 Hardware Large Receive Offload (HW LRO)

Large Receive Offload (LRO) increases inbound throughput of high-bandwidth network connections by reducing CPU overhead. It works by aggregating multiple incoming packets from a single stream into a larger buffer before they are passed higher up the networking stack, thus reducing the number of packets that have to be processed.

- ***In order to turn on the LRO device, run:***

```
#> ifconfig mce<x> lro
```

- ***In order to turn off the LRO device, run:***

```
#> ifconfig mce<x> -lro
```

When the LRO device is on, HW LRO can be turned on. HW LRO is off by default.

- ***In order to turn on HW LRO run:***

```
#> sysctl dev.mce.<N>.conf.hw_lro=1
```

- ***In order to turn off HW LRO run:***

```
#> sysctl dev.mce.<N>.conf.hw_lro=0
```

3.2 EEPROM Cable Information Reader

EEPROM cable reading feature allows reading important information about the plugged cable, such as cable type, cable speed, vendor and more.

In order to read the cable EEPROM info:

1. Read the cable information by enabling the following sysctl parameter. Output will be printed in dmesg:

```
#> sysctl dev.mce.<X>.conf.eeprom_info=1
```

Example:

```
#>sysctl dev.mce.1.conf.eeprom_info=1
dev.mce.1.conf.eeprom_info: 0 -> 0

 #>dmesg
   Offset          Values
   ------          ------
   0x0000          0d 05 06 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0010          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0020          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0030          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0040          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0050          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   0x0060          00 00 00 00 00 00 00 00 00 00 00 00 01 00 04 00
```

```
    0x0070          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    0x0080          0d 00 23 88 00 00 00 00 00 00 00 00 ff 00 00 00
    0x0090          00 00 01 a0 4d 65 6c 6c 61 6e 6f 78 20 20 20 20
    0x00a0          20 20 20 20 1f 00 02 c9 4d 43 50 31 36 30 30 2d
    0x00b0          45 30 30 41 20 20 20 20 41 32 02 03 04 07 00 3f
    0x00c0          0b 00 00 00 4d 54 31 35 32 31 56 53 30 36 34 38
    0x00d0          34 20 20 20 31 35 30 35 32 36 20 20 00 00 67 5e
    0x00e0          31 32 38 38 35 35 32 33 38 44 33 33 00 00 00 00
    0x00f0          00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 00
```

2. Another option for reading cable information is by using the ifconfig:

```
#>ifconfig -v mce<X>
#>ifconfig -vv mce<X>
#>ifconfig -vvv mce<X>
```

Example:

```
#> ifconfig -vvv mce1
plugged: QSFP+ 40GBASE-CR4 (No separate connector)
vendor: Mellanox PN: MCP1600-E00A SN: MT1521VS06484 DATE: 2015-05-26
compliance level: SFF-8636 rev <=1.5
nominal bitrate: 25750 Mbps

 SFF8436 DUMP (0xA0 128..255 range):
 0D 00 23 88 00 00 00 00 00 00 00 00 FF 00 00 00
 00 00 01 A0 4D 65 6C 6C 61 6E 6F 78 20 20 20 20
 20 20 20 20 1F 00 02 C9 4D 43 50 31 36 30 30 2D
 45 30 30 41 20 20 20 20 41 32 02 03 04 07 00 3F
 0B 00 00 00 4D 54 31 35 32 31 56 53 30 36 34 38
 34 20 20 20 31 35 30 35 32 36 20 20 00 00 67 5E
 31 32 38 38 35 35 32 33 38 44 33 33 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 30 00 00

 SFF8436 DUMP (0xA0 0..81 range):
 0D 05 06 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00
```

3.3 Packet Pacing

This feature is supported in firmware v12.17.1016 and above.

Packet pacing, also known as “rate limit,” defines a maximum bandwidth allowed for a TCP connection. Limitation is done by hardware where each QP (transmit queue) has a rate limit value from which it calculates the delay between each packet sent.

➢ ***To enable Packet Pacing in firmware:***

1. Create a file with the following content:

```
# vim /tmp/enable_packet_pacing.txt
MLNX_RAW_TLV_FILE
0x00000004 0x0000010c 0x00000000 0x00000001
```

2. Update firmware configuration to enable Packing Pacing:

```
mlxconfig -d pci0:<x>:0:0 -f /tmp/enable_packet_pacing.txt set_raw
```

3. Reset the firmware:

```
mlxfwreset -d pci0:<x>:0:0 reset
```

Packet Pacing and Quality of Service (QoS) features do not co-exist.

3.3.1 Setting Rates for Packet Pacing

Rates that are being used with packet pacing must be defined in advance.

New Rates Configuration

- Newly configured rates must be within a certain range, determined by the firmware, and they can be read through sysctl.
  - For a minimum value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_limit_min
```

  - For a maximum value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_limit_max
```

- The number of configured rates is also determined by the firmware. In order to check how many rates can be defined, run:

```
sysctl dev.mce.<N>.rate_limit.tx_rates_max
```

- To add a new rate:

```
sysctl dev.mce.<N>.rate_limit.tx_limit_add=800000
```

This will add the defined rate to the next available index. If all rates were already defined with an index, the new rate will not be added.

Rates are determined and then saved in bits per second.
Rates requested for a new socket are added in bytes per second.

- To remove a rate limit, run:

```
sysctl dev.mce.<N>.rate_limit.tx_limit_clr=80000
```

Deviation:

The user can specify a maximum deviation of the rate via sysctl. If the rate limit table cannot satisfy the requirement, rate limiting will be disabled.

- For minimum value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_allowed_deviation_min
```

- For maximum value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_allowed_deviation_max
```

- For changing the deviation value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_allowed_deviation=10000
```

- For reading the current deviation value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_allowed_deviation
```

Limitation: Rate values must be multiples of 1000.

- Burst size is determined by the hardware, and can be configured via sysctl:
- For a minimum value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_burst_size_min
```

- For a maximum value, run:

```
sysctl dev.mce.<N>.rate_limit.tx_burst_size_max
```

- For changing burst level, run:

```
sysctl dev.mce.<N>.rate_limit.tx_burst_size=150
```

- To read which burst level was defined, run:

```
sysctl dev.mce.<N>.rate_limit.tx_burst_size
```

- For displaying the packet pacing configuration, run:

```
sysctl dev.mce.<N>.rate_limit.tx_rate_show
                ENTRYBURST      RATE [bit/s]
                -----------------------------------
                  0150          800000
                  1150          40000
                  2150          1000000
                  3150          25000000000

                ENTRYBURST      RATE [bit/s]
                -----------------------------------
                  03            800000
                  13            40000
                  23            1000000
                  33            25000000000
```

where:

Entry	Rate limit table entry
Burst	Burst size configured for rate limit traffic
Rate	Rate configured for the relevant index

All rates are shown in bits per second.

3.3.2 Using Packet Pacing Sockets

1. Create a rate-limited socket according to the desired rate using the setsockopt() interface **based on the previous section:**

```
setsockopt(s, SOL_SOCKET, SO_MAX_PACING_RATE, pacing_rate, sizeof(pacing_rate))
```

SO_MAX_PACING_RATE	Marks the socket as a rate limited socket
pacing_rate	Defined rate in bytes/sec. The type is unsigned int. **Note:** The same value entered via sysctl in bytes instead of bits.

   - A rate-limited ring corresponding to the requested rate will be created and associated to the relevant socket.
   - Rate-limited traffic will be transmitted when data is sent via the socket.
2. Modify the rate-limited value using the same socket.
3. Destroy the relevant ring upon TCP socket completion.

3.3.2.1 Error Detection

Detecting failures can be done using the getsockopt() interface to query a specific socket.

3.3.3 Feature Characteristics

- MLNX_OFED for FreeBSD supports up to 100,000 rate limited TCP connections.
- Each TCP connection is mapped to a specific SQ

3.3.4 Limitations

- Max rate limited rings is 100,000
- Min rate: 1 Kbps
- Max rate: 100 Gbps

```
#> sysctl -a | grep rate_limit
sysctl dev.mce.<N>.rate_limit.tx_limit_min: 1000
sysctl dev.mce.<N>.rate_limit.tx_limit_max: 100000000000
```

3.3.5 Performance Tuning

The following settings are recommended for a large number of connections to reduce the amount of overhead related to connection processing, as well as to handle the increased use of network buffers.

- Increase size of rate limit send queue:

```
# sysctl dev.mce.<N>.rate_limit.tx_queue_size=1024
```

- Reduce number of completion events per rate limit send queue:

```
# sysctl dev.mce.<N>.rate_limit.tx_completion_fact=-1
```

- Increase non-rate-limit send queue size:

```
# sysctl dev.mce.<N>.conf.tx_queue_size=16384
```

- Reduce number of completion events per send queue:

```
# sysctl dev.mce.<N>.conf.tx_completion_fact=-1
```

- Increase receive queue size and allow many packets to be accumulated.

  This gives better TX burst performance:

```
# sysctl dev.mce.<N>.conf.rx_queue_size=16384
# sysctl dev.mce.<N>.conf.rx_coalesce_usecs=250
# sysctl dev.mce.<N>.conf.rx_coalesce_pkts=4096
```

- Note for production. Allow high number of connections to terminate simultaneously:

```
# sysctl net.inet.icmp.icmplim=-1
```

- Increase memory pool for network buffers:

```
# sysctl kern.ipc.nmbufs=100000000
```

3.4 RDMA over Converged Ethernet (RoCE)

RoCE v2 is currently supported in FreeBSD v12-CURRENT only.

Remote Direct Memory Access (RDMA) is the remote memory management capability that allows server-to-server data movement directly between application memory without any CPU involvement. RDMA over Converged Ethernet (RoCE) is a mechanism to provide this efficient data transfer with very low latencies on lossless Ethernet networks. With advances in data center convergence over reliable Ethernet, ConnectX® Ethernet adapter cards family with RoCE uses the proven and efficient RDMA transport to provide the platform for deploying RDMA technology in mainstream data center application at 10GigE, 25GigE, 40GigE, 50GigE, and 100 GigE link-speed. ConnectX® Ethernet adapter cards family with its hardware offload support takes advantage of this efficient RDMA transport (InfiniBand) services over Ethernet to deliver ultra-low latency for performance-critical and transaction intensive applications such as financial, database, storage, and content delivery networks.

When working with RDMA applications over Ethernet link layer the following points should be noted:

- The presence of a Subnet Manager (SM) is not required in the fabric. Thus, operations that require communication with the SM are managed in a different way in RoCE. This does not affect the API but only the actions such as joining multicast group, that need to be taken when using the API
- Since LID is a layer 2 attribute of the InfiniBand protocol stack, it is not set for a port and is displayed as zero when querying the port
- With RoCE, the alternate path is not set for RC QP. Therefore, APM (another type of High Availability and part of the InfiniBand protocol) is not supported
- Since the SM is not present, querying a path is impossible. Therefore, the path record structure must be filled with the relevant values before establishing a connection. Hence, it is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure
- VLAN tagged Ethernet frames carry a 3-bit priority field. The value of this field is derived from the IB SL field by taking the 3 least significant bits of the SL field
- RoCE traffic is not shown in the associated Ethernet device's counters since it is offloaded by the hardware and does not go through Ethernet network driver. RoCE traffic is counted in the same place where InfiniBand traffic is counted;
  sysctl sys.class.infiniband.<device>.ports.<port number>.counters

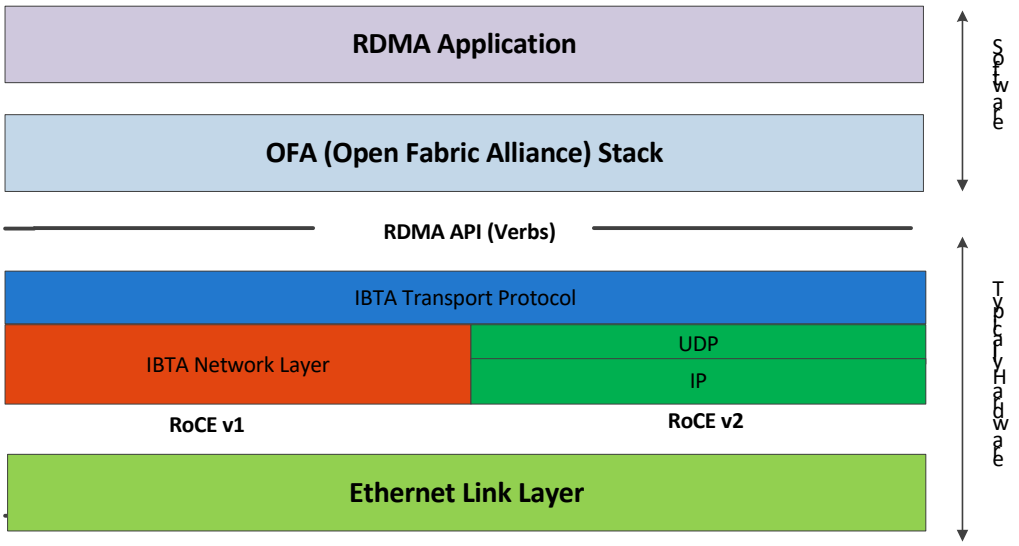
3.4.1 RoCE Modes

RoCE encapsulates IB transport in one of the following Ethernet packet

- RoCE v1 - dedicated ether type (0x8915)

- RoCE v2 - UDP and dedicated UDP port (4791)

Figure 1: RoCE v1 and RoCE v2 Protocol Stack



3.4.1.1 RoCE v1

RoCE v1 protocol is defined as RDMA over Ethernet header (as shown in the figure above). It uses ethertype 0x8915 and can be used with or without the VLAN tag. The regular Ethernet MTU applies on the RoCE frame.

3.4.1.2 RoCE v2

RoCE v2 is supported **ONLY** in ConnectX®-3 Pro adapter cards and above.

A straightforward extension of the RoCE protocol enables traffic to operate in IP layer 3 environments. This capability is obtained via a simple modification of the RoCE packet format. Instead of the GRH used in RoCE, IP routable RoCE packets carry an IP header which allows traversal of IP L3 Routers and a UDP header (RoCE v2 only) that serves as a stateless encapsulation layer for the RDMA Transport Protocol Packets over IP.

The proposed RoCE v2 packets use a well-known UDP destination port value that unequivocally distinguishes the datagram. Similar to other protocols that use UDP encapsulation, the UDP source port field is used to carry an opaque flow-identifier that allows network devices to implement packet forwarding optimizations (e.g. ECMP) while staying agnostic to the specifics of the protocol header format.

Furthermore, since this change exclusively affects the packet format on the wire, and due to the fact that with RDMA semantics packets are generated and consumed below the AP, applications can seamlessly operate over any form of RDMA service, in a completely transparent way.

3.4.1.3 RoCE Modes Parameters

While ConnectX®-3 supports only RoCE v1, ConnectX®-3 Pro supports both RoCEv1 and RoCE v2. The RoCE mode can be set using the `'sys.class.infini-band.<device>.default_roce_mode_port<N>'` parameter in the `boot/loader.conf` file or by using the sysctl utility.

The following are the possible RoCE mode values:

- If set to 'IB/RoCE v1', the driver will use RoCE v1 by default
- If set to 'RoCE v2', the driver will use RoCEv 2 by default

ConnectX-4 adapter cards family and above supports both RoCE v1 and RoCE v2. By default, the driver associates all GID indexes to RoCE v1 and RoCE v2, thus, a single entry for each RoCE version.

3.4.2 GID Table Population

GID table entries are created whenever an IP address is configured on one of the Ethernet devices of the NIC's ports. Each entry in the GID table for RoCE ports has the following fields:

- GID value
- GID type
- Network device

For ports on devices that support two RoCE modes (ConnectX®-3 Pro and above) the table will be occupied with two GID entries, both with the same GID value but with different types. The Network device in an entry is the Ethernet device with the IP address that GID is associated with. The GID format can be of 2 types, IPv4 and IPv6. IPv4 GID is an IPv4-mapped IPv6 address while IPv6 GID is the IPv6 address itself. Layer 3 header for packets associated with IPv4 GIDs will be IPv4 (for RoCE v2) and IPv6/GRH for packets associated with IPv6 GIDs and IPv4 GIDs for RoCE v1.

The number of entries in the GID table is equal to $N^{1,2}(K+1)$ where N is the number of IP addresses that are assigned to all network devices associated with the port including VLAN devices and alias devices, but excluding bonding masters, since RoCE LAG is not supported in FreeBSD. Link local IPv6 addresses are excluded from this count since the GID for them is always preset (the default GIDs) at the beginning of each table. K is the number of the supported RoCE types. Since the number of entries in the hardware is limited to 128 for each port, it is important to understand the limitations on N. MLNX_OFED provides a script called show_gids to view the GID table conveniently.

3.4.2.1 GID Table in sysctl Tree

GID table is exposed to user space via the sysctrl tree.

1. When the mode of the device is RoCE v1/RoCE v2, each entry in the GID table occupies 2 entries in the hardware. In other modes, each entry in the GID table occupies a single entry in the hardware.
2. In multifunction configuration, the PF gets 16 entries in the hardware while each VF gets 112/F where F is the number of virtual functions on the port. If 112/F is not an integer, some functions will have 1 less entries than others. **Note** that when F is larger than 56, some VFs will get only one entry in the GID table.

- GID values can be read from:

```
sysctl sys.class.infiniband.{device}.ports.{port}.gids.{index}
```

- GID type can be read from:

```
sysctl sys.class.infiniband.{device}.ports.{port}.gid_attrs.types.{index}
```

- GID net_device can be read from:

```
sysctl sys.class.infiniband.{device}.ports.{port}.gid_attrs.ndevs.{index}
```

3.4.2.2 GID Table Example

The following is an example of the GID table.

```
DEV      PORT   INDEX   GID                                     IPv4           VER       DEV
----     ----   -----   ----                                    -----          -----     -------
mlx5_0   1      0       fe80:0000:0000:0000:0202:c9ff:feb6:7c70                V2        mce1
mlx5_0   1      1       fe80:0000:0000:0000:0202:c9ff:feb6:7c70                V1        mce1
mlx5_0   1      2       0000:0000:0000:0000:0000:ffff:c0a8:0146 192.168.1.70   V2        mce1
mlx5_0   1      3       0000:0000:0000:0000:0000:ffff:c0a8:0146 192.168.1.70   V1        mce1
mlx5_0   1      4       0000:0000:0000:0000:0000:ffff:c1a8:0146 193.168.1.70   V2        mce1.100
mlx5_0   1      5       0000:0000:0000:0000:0000:ffff:c1a8:0146 193.168.1.70   V1        mce1.100
mlx5_0   1      6       1234:0000:0000:0000:0000:0000:0000:0070                V2        mce1
mlx5_0   1      7       1234:0000:0000:0000:0000:0000:0000:0070                V1        mce1
mlx5_0   2      0       fe80:0000:0000:0000:0202:c9ff:feb6:7c71                V2        mce1
mlx5_0   2      1       fe80:0000:0000:0000:0202:c9ff:feb6:7c71                V1        mce1
```

Where:

- Entries on port 1 index 0/1 are the default GIDs, one for each supported RoCE type
- Entries on port 1 index 2/3 belong to IP address 192.168.1.70 on eth1.
- Entries on port 1 index 4/5 belong to IP address 193.168.1.70 on eth1.100.
- Packets from a QP that is associated with these GID indexes will have a VLAN header (VID=100)
- Entries on port 1 index 6/7 are IPv6 GID. Packets from a QP that is associated with these GID indexes will have an IPv6 header

3.4.3 RoCE Packet Sniffing

RoCE packets can be sniffed using tcpdump tool using `tcpdump -i <RDMA device>`.

Example:

```
# tcpdump -i mlx5_0
```

3.4.4 Perftest

Perftest is a collection of tests for RDMA micro-benchmarking that can be installed via ports: benchmarks/perftest.

3.5 Flow Control

3.5.1 Global Pause

Global Pause is a control frame sent by the receiver to alert the sender that the receiver buffer is about to overflow.

Both Tx and Rx global pause frames are enabled by default.

3.5.2 Priority Flow Control (PFC)

PFC, IEEE 802.1Qbb, applies pause functionality to specific classes of traffic on the Ethernet link. For example, PFC can provide lossless service for the RoCE traffic and best-effort service for the standard Ethernet traffic. PFC can provide different levels of service to specific classes of Ethernet traffic (using IEEE 802.1p traffic classes).

3.5.2.1 PFC Local Configuration on ConnectX-4/ConnectX-5

Step 1. Disable global pause frames. Example:

```
# ifconfig mce<N> media autoselect mediaopt full-duplex
```

Step 2. Enable PFC on the desired priority by using the sysctl utility:

```
dev.mce.<N>.rx_priority_flow_control_<prio>: <enabled:1 disabled:0>
dev.mce.<N>.tx_priority_flow_control_<prio>: <enabled:1 disabled:0>
```

➢ ***To check PFC statistics:***

```
# sysctl -a dev.mce.<N>.pstats | grep prio<prio>
```

3.5.3 PFC Hardware Buffer Configuration

- **dev.mce.X.conf.qos.buffers_size** - the hardware allows to configure up to eight buffer sizes. Total sum of all buffers must not exceed the hardware memory size. The limitation is enforced automatically. The sysctl allows to set each buffer size. Buffer space exhaustion causes the card to send xoff to the other side of the link.
- **dev.mce.X.conf.qos.buffers_prio** - maps buffer index into the hardware-defined priority. Note that the priority is the internal number after translation from the external QoS parameters.
- **dev.mce.X.conf.qos.cable_length** - for more precise determination of the moment when xoff should be issued, user might specify the cable length in meters, which is used to calculate the signal propagation delay.

3.5.4 Explicit Congestion Notification (ECN)

ECN in ConnectX-4 and ConnectX-5 HCAs enables end-to-end congestion notifications between two end-points when a congestion occurs, and works over Layer 3. ECN must be enabled on all nodes in the path (nodes, routers, etc.) between the two end points and the intermediate devices (switches) between them to ensure reliable communication. ECN handling is supported only for RoCEv2 packets.

➢ ***To enable ECN on the hosts:***

Step 1. Load mlx5ib(4) and mlx5en(4):

```
# kldload mlx5ib mlx5en
```

Step 2. Query the relevant attributes:

```
# sysctl -a sys.class.infiniband.mlx5_<devno>.cong.conf
```

Step 3. Modify the attributes:

```
# sysctl sys.class.infiniband.mlx5_<devno>.cong.conf.<attr>=<value>
```

ECN supports the following algorithms:

- `r_roce_ecn_rp - Reaction point`
- `r_roce_ecn_np - Notification point`

Each algorithm has a set of relevant parameters and statistics, which are defined per device. ECN and QCN are not compatible.

3.6 Quality of Service

Quality of Service (QoS) is a mechanism of assigning a priority to a network flow and manage its guarantees, limitations and its priority over other flows. This is accomplished by mapping the User Priority (UP) to a hardware Traffic Class (TC). TC is assigned with the QoS attributes and the different flows behave accordingly.

Note: Packet Pacing and Quality of Service (QoS) features do not co-exist.

➢ ***To be able to work with QoS, make sure to disable Packet Pacing in firmware:***

1. Create a file with the following content:

```
# vim /tmp/disable_packet_pacing.txt
MLNX_RAW_TLV_FILE
0x00000004 0x0000010c 0x00000000 0x00000000
```

2. Update firmware configuration to disable Packing Pacing:

```
mlxconfig -d pci0:<x>:0:0 -f /tmp/disable_packet_pacing.txt set_raw
```

3. Reset the firmware:

```
mlxfwreset -d pci0:<x>:0:0 reset
```

3.6.1 Priority Code Point (PCP)

PCP is used as a means for classifying and managing network traffic, and providing QoS in Layer 2 Ethernet networks. It uses the 3-bit PCP field in the VLAN header for the purpose of packet classification.

➢ ***To create a VLAN interface and assign the desired priority to it:***

```
# ifconfig mce<N>.<vlan> create
# ifconfig mce<N>.<vlan> vlanpcp <prio>
```

3.6.1.1 VLAN 0 Priority Tagging

The VLAN 0 Priority Tagging feature enables 802.1Q Ethernet frames to be transmitted with VLAN ID set to zero.

Setting the VLAN ID tag to zero allows its tag to be ignored, and the Ethernet frame to be processed according to the priority configured in the 802.1P bits of the 802.1Q Ethernet frame header.

➢ ***To enable VLAN 0 priority tagging on a specific interface:***

```
# ifconfig mce<N> pcp <prio>
```

➢ ***To disable VLAN 0 priority tagging on a specific interface:***

```
# ifconfig mce<N> -pcp
```

Switch port must be configured to accept VLAN 0 priority tagged packets. Otherwise, these packets may be dropped.

3.6.2 Differentiated Service Code Point (DSCP)

Differentiated services or DiffServ is a computer networking architecture that specifies a simple and scalable mechanism for classifying and managing network traffic and providing quality of service (QoS) on IP networks.

DiffServ uses a 6-bit DSCP in the 8-bit DS field in the IP header for packet classification purposes. The DS field replaces the outdated IPv4 TOS field.

3.6.3 Trust State

Trust state enables prioritizing sent/received packets based on packet fields.

The default trust state is PCP. Ethernet packets are prioritized based on the value of the field (PCP/DSCP/BOTH).

To configure Trust State, use the following sysctl node:

```
# sysctl -d dev.mce.<N>.conf.qos.trust_state
dev.mce.<N>.conf.qos.trust_state: Set trust state, 1:PCP 2:DSCP 3:BOTH
```

3.6.4 QoS with RDMA

RDMA application is responsible for setting QoS values.

- In RDMA CM mode, QoS is set in the rdma_id_private struct in the tos field.

  Incoming RDMA CM connections always take precedence setting the current priority.
- In non-RDMA CM mode, priority values are set using a modify_qp command with ibv_qp_attr parameter. IPv4 type of service (“ToS”) and IPv6 traffic class are set using the attr.ah_attr.grh.traffic_class field. VLAN PCP is set using the attr.ah_attr.sl field.

3.6.5 Mapping User Priority to Traffic Class

This feature allows users to map a specific User Priority (UP) to a specific TC.

Note that this configuration is permanent and will not be reset to default unless manually changed.

Example

➢ ***To map UP 5 to TC 4 on device mce0:***

```
# sysctl dev.mce.0.conf.qos.prio_0_7_tc=1,0,2,3,4,4,6,7
dev.mce.0.conf.qos.prio_0_7_tc: 1 0 2 3 4 5 6 7 -> 1 0 2 3 4 4 6 7
```

Note: By default, UP 0 is mapped to TC 1, and UP 1 is mapped to TC 0:

```
# sysctl dev.mce.0.conf.qos.prio_0_7_tc
dev.mce.0.conf.qos.prio_0_7_tc: 1 0 2 3 4 5 6 7
```

3.6.6 Mapping DSCP to Priority Mapping

Each DSCP value can be mapped to a priority using the following sysctl nodes:

```
dev.mce.<N>.conf.qos.dscp_56_63_prio: 7 7 7 7 7 7 7 7
dev.mce.<N>.conf.qos.dscp_48_55_prio: 6 6 6 6 6 6 6 6
dev.mce.<N>.conf.qos.dscp_40_47_prio: 5 5 5 5 5 5 5 5
dev.mce.<N>.conf.qos.dscp_32_39_prio: 4 4 4 4 4 4 4 4
dev.mce.<N>.conf.qos.dscp_24_31_prio: 3 3 3 3 3 3 3 3
dev.mce.<N>.conf.qos.dscp_16_23_prio: 2 2 2 2 2 2 2 2
dev.mce.<N>.conf.qos.dscp_8_15_prio: 1 1 1 1 1 1 1 1
dev.mce.<N>.conf.qos.dscp_0_7_prio: 0 0 0 0 0 0 0 0
```

Example:

```
# sysctl dev.mce.0.conf.qos.dscp_0_7_prio=1,1,1,1,1,1,1,1
dev.mce.0.conf.qos.dscp_0_7_prio: 0 0 0 0 0 0 0 0 -> 1 1 1 1 1 1 1 1
```

3.6.7 Maximum Rate Limiting

This feature allows users to rate limit a specific TC. Rate limit defines a maximum bandwidth allowed for a TC. Please note that 10% deviation from the requested values is considered acceptable.

Note that instead of setting the maximum rate for a single priority, you should pass the maximum rates for all relevant priorities as a single input.

Notes:

- This configuration is permanent and will not be set to default unless manually changed
- Rate is specified in kilobits, where kilo=1000.
- Rate must be divisible by 100,000, meaning that values must be in 100Mbs units.
  Examples for valid values:
  - 200000 - 200Mbs
  - 1000000 - 1Gbs
  - 3400000 - 3.4Gbs
- 0 value = unlimited rate

Example

➢ ***To "rate limit" TC 4 on device mce1 to 2.4Gbits:***

```
# sysctl dev.mce.0.conf.qos.tc_max_rate=0,0,0,0,2400000,0,0,0
dev.mce.0.conf.qos.tc_max_rate: 0 0 0 0 0 0 0 0 -> 0 0 0 0 2400000 0 0 0
```

3.6.8 Enhanced Transmission Selection (ETS)

To be able to fully utilize this feature, make sure Priority Flow Control (PFC) feature is enabled.

Enhanced Transmission Selection standard (ETS) exploits the time periods in which the offered load of a particular Traffic Class (TC) is less than its minimum allocated bandwidth by allowing the difference to be available to other traffic classes.

After servicing the strict priority TCs, the amount of bandwidth (BW) left on the wire may be split among other TCs according to a minimal guarantee policy.

If, for instance, TC0 is set to 80% guarantee and TC1 to 20% (the TCs sum must be 100), then the BW left after servicing all strict priority TCs will be split according to this ratio.

Since this is a minimal guarantee, there is no maximum enforcement. This means, in the same example, that if TC1 did not use its share of 20%, the reminder will be used by TC0.

Example

```
sysctl dev.mce.0.conf.qos.tc_rate_share=20,10,10,10,10,10,10,20
```

In this example, Priority 7 and Priority 0 are guaranteed for 20% of the bandwidth, and all the rest are guaranteed for 10% of the bandwidth.

3.6.9 Priority Flow Control (PFC) Hardware Buffer Configuration

Support for this feature is currently at beta level.

Hardware buffers configuration can be tuned for priority flow control (PFC).

Table 5 - Buffer Parameters

Parameter	Description
dev.mce.X.conf.qos.buffers_size	This parameter is used to set the buffer size. The hardware allows to configure up to eight buffers sizes. The total sum of all buffers must not exceed the hardware memory size. The limitation is enforced automatically. Sysctl allows to set each buffer size. Buffer space exhaustion causes the adapter card to send xoff to the other side of the link.
dev.mce.X.conf.qos.buffers_prio	This parameter shows the mapping between priority to buffer. Maps buffer index into the hardware-defined priority. Note that the priority is the internal number after translation from the external QoS parameters.
dev.mce.X.conf.qos.cable_length	For more precise determination of the moment when xoff should be issued, users may specify the cable length in meters to calculate the signal propagation delay.

3.7 Rx Hardware Time-Stamping

This feature is supported in FreeBSD v12-CURRENT and onwards.

Time-stamping is the process of keeping track of the creation of a packet. A time-stamping service supports assertions of proof that a datum existed before a particular time. Incoming packets are time-stamped before they are distributed on the PCI, depending on the congestion in the PCI buffers. Outgoing packets are time-stamped very close to placing them on the wire.

ConnectX-4 and above adapter cards support high quality internal timer that can provide time-stamps on each received packet. Resulting quality of time-stamp is much higher than software can provide, and can be transparently utilized by applications that use standard BSD socket features such as SO_TIMESTAMP.

To verify that your HCA and operating system support time-stamping, check the presence of HWRXTSTMP option in the ifconfig(8) output of the interface:

```
# ifconfig mce4
mce4: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=2ed07bb<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,JUMBO_MTU,VLAN_HWCSUM,TSO4,TSO6,LRO,VLAN_H
WFILTER,VLAN_HWTSO,LINKSTATE,RXCSUM_IPV6,TXCSUM_IPV6,HWRXTSTMP>
```

- ***To disable the feature, run:***

```
# ifconfig mce<N> -hwrxtsmp
```

- ***To re-enable the feature, run:***

```
# ifconfig mce<N> hwrxtsmp
```

This feature is only supported using the firmware version compatible with your FreeBSD driver version.

3.8 Firmware Dump

This feature introduces the ability to dump hardware registered data upon demand. Data that is dumped is stored in a kernel buffer, and can be later manually copied to userspace using mlx-5tool. It can be maintained until explicitly cleared.

mlx5tool is a simple program which utilizes the mlx5io interface and allows executing the following firmware dump commands:

- `MLX5_FWDUMP_FORCE` - forces dump unless one was already stored in the kernel buffer
- `MLX5_FWDUMP_GET` - copies the recorded dump from kernel into user mode
- `MLX5_FWDUMP_RESET` - clears the kernel buffer, in preparation for storing another dump

Commands are communicated with the driver using ioctl interface over the devfs device /dev/mlx5ctl.

➢ ***To build the mlx5tool, run the following command from the driver's top-level directory:***

```
cd usr.sbin/mlx5tool && make all install clean
```

mlx5tool Usage

`mlx5tool -d pci0:<x>:0:0 -e`	Force dump, storing it into the kernel buffer
`mlx5tool -d pci0:<x>:0:0 -w [-o dump.file]`	Store the recorded dump into file dump.file. If -o is omitted, the dump is streamed into standard output.
`mlx5tool -d pci0:<x>:0:0 -r`	Reset dump

3.9 Forward Error Correction (FEC)

Forward Error Correction, or FEC, is configurable per MLX5EN(4) device. The following set of sysctls define the API for FEC:

```
dev.mce.<N>.conf.fec.avail_50x: 0 0 0 0
dev.mce.<N>.conf.fec.mask_50x: 0 0 0 0
dev.mce.<N>.conf.fec.avail_10x_25x: 3 7 3 4
dev.mce.<N>.conf.fec.mask_10x_25x: 0 0 0 0
dev.mce.<N>.conf.fec.mode_active: 4
```

Note: The avail_xxx sysctl provides the available bits which you can set in the associated mask/masks. To get a description of the bit values, refer to the sysctl description available per variable. The mode_active sysctl gives the value of the currently active FEC method. A FEC mask of zero means that the default values should be used.

3.10 IRQ Labeling

IRQ IDs are now labeled according to their Mellanox device functionality.

➢ ***To obtain the current Mellanox devices IRQ mapping:***

1. Enable the debug status:

   ```
   sysctl dev.mce.<N>.conf.debug_stats=1
   ```

2. Dump the IRQ vectors by reading the hw_ctx_debug sysctl:

   ```
   sysctl dev.mce.<N>.hw_ctx_debug
   ```

Output Example

```
sysctl -n dev.mce.0.hw_ctx_debug
pages irq 99
command irq 100
async irq 101
channel 0 rq 131 cq 17 irq 102
channel 0 tc 0 sq 130 cq 16 irq 102
channel 1 rq 134 cq 19 irq 103
channel 1 tc 0 sq 133 cq 18 irq 103
channel 2 rq 137 cq 21 irq 104
channel 2 tc 0 sq 136 cq 20 irq 104
channel 3 rq 140 cq 23 irq 105
channel 3 tc 0 sq 139 cq 22 irq 105
channel 4 rq 143 cq 25 irq 106
channel 4 tc 0 sq 142 cq 24 irq 106
channel 5 rq 146 cq 27 irq 107
channel 5 tc 0 sq 145 cq 26 irq 107
channel 6 rq 149 cq 29 irq 108
channel 6 tc 0 sq 148 cq 28 irq 108
channel 7 rq 152 cq 31 irq 109
channel 7 tc 0 sq 151 cq 30 irq 109
channel 8 rq 155 cq 49 irq 110
channel 8 tc 0 sq 154 cq 48 irq 110
channel 9 rq 158 cq 51 irq 111
channel 9 tc 0 sq 157 cq 50 irq 111
channel 10 rq 161 cq 53 irq 112
channel 10 tc 0 sq 160 cq 52 irq 112
channel 11 rq 164 cq 55 irq 113
channel 11 tc 0 sq 163 cq 54 irq 113
```

The value after the “irq” keyword indicates which system IRQ vector is used for the given resource. This IRQ value is accepted by utilities such as cpuset(1). At the moment, there are five different resources that use IRQ vector:

1. Page allocation or free interrupts
2. Command interrupts
3. Asynchronous event interrupts
4. Transmit ring completion interrupts
5. Receive ring completion interrupt

3.11 Port Module Events

The following statistics indicate the number of various port module events.

Table 6 - Port Module Events

Event	Description
dev.mlx5_core.<device_number>.pme_stats.module_unplug	Number of time module unplugged
dev.mlx5_core.<device_number>.pme_stats.module_plug	Number of time module plugged

Table 6 - Port Module Events

Event	Description
dev.mlx5_core.<device_number>.pme_stats.errors.cable_shorted	Module Cable is shorted
dev.mlx5_core.<device_number>.pme_stats.errors.high_temp	Module High Temperature
dev.mlx5_core.<device_number>.pme_stats.errors.unknown_id	Module Unknown identifier
dev.mlx5_core.<device_number>.pme_stats.errors.enforce_part_number	Module Enforce part number list
dev.mlx5_core.<device_number>.pme_stats.errors.no_eeprom	No EEPROM/retry timeout
dev.mlx5_core.<device_number>.pme_stats.errors.bus_stuck	Module Bus stuck(I2C or data shorted)
dev.mlx5_core.<device_number>.pme_stats.errors.long_range	Module Long Range for non MLNX cable/module
dev.mlx5_core.<device_number>.pme_stats.errors.power_budget	Module Power Budget Exceeded

4 Performance Tuning

In order to improve performance, please make sure the HW LRO is enabled.

4.1 Receive Queue Interrupt Moderation

An armed CQ will generate an event when either of the following conditions is met:

- The number of completions generated since the one which trigged the last event generation reached a set in advance number.
- The timer has expired and an event is pending.

The timer can be set to be restarted either upon event generation or upon completion generation.

Setting the timer to be restarted upon completion generation affects the interrupt receiving rate. When receiving a burst of incoming packets, the timer will not reach its limit, therefore, the interrupt rate will be associated to the size of the packets.

➢ ***In order to modify the timer restart mode, run:***

```
#> sysctl dev.mce.<N>.conf.rx_coalesce_mode=[0/1/2/3]
```

0: For timer restart upon event generation.

1: For timer restart upon completion generation.

2: For timer restart upon event generation where usecs and pkts values are adaptive/dynamic, depending on the traffic type and network usage.

3: For timer restart upon completion generation where usecs and pkts values are adaptive/dynamic, depending on the traffic type and network usage.

➢ ***In order to modify the number of completions generated between interrupts, run:***

```
#> sysctl dev.mce.<N>.conf.rx_coalesce_pkts=<x>
```

➢ ***In order to modify the time for the timer to finish, run:***

```
#> sysctl dev.mce.<N>.conf.rx_coalesce_usecs=<x>
```

Note: The default values are:

- dev.mce.1.conf.rx_coalesce_mode: 1 - Timer restarts upon completion generation.
- dev.mce.1.conf.rx_coalesce_pkts: 32 - 32 completions generate interrupts.
- dev.mce.1.conf.rx_coalesce_usecs: 3 - Timer count down 3 micro sec.

4.2 Tuning for NUMA Architecture

4.2.1 Single NUMA Architecture

When using a server with single NUMA, no tuning is required. Also, make sure to avoid using core number 0 for interrupts and applications.

1. Find a CPU list:

```
#> sysctl -a | grep "group level=\"2\"" -A 1
<group level="2" cache-level="2">
<cpu count="12" mask="fff">0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</cpu>
```

2. Tune Mellanox NICs to work on desirable cores
   a. Find the device that matches the interface:

```
#> sysctl -a | grep mce | grep mlx
dev.mce.<N>.conf.device_name: mlx5_core1
dev.mce.<N>.conf.device_name: mlx5_core0
```

   b. Find the device interrupts.

```
vmstat -ia | grep mlx5_core0 | awk '{print $1}' | sed s/irq// | sed s/://
269
270
271
…
```

   c. Bind each interrupt to a desirable core.

```
cpuset -x 269 -l  1
cpuset -x 270 -l  2
cpuset -x 271 -l  3
…
```

   d. Bind the application to the desirable core.

```
cpuset -l 1-11 <app name> <sever flag>
cpuset -l 1-11 <app name> <client flag> <IP>
```

Specifying a range of CPUs when using the cpuset command will allow the application to choose any of them. This is important for applications that execute on multiple threads.
The range argument is not supported for interrupt binding.

4.2.2 Dual NUMA Architecture

1. Find the CPU list closest to the NIC
   a. Find the device that matches the interface:

```
#> sysctl -a | grep mce | grep mlx
dev.mce.3.conf.device_name: mlx5_core3
dev.mce.2.conf.device_name: mlx5_core2
dev.mce.1.conf.device_name: mlx5_core1
dev.mce.0.conf.device_name: mlx5_core0
```

   b. Find the NIC's PCI location:

```
#> sysctl -a | grep mlx5_core.0 | grep parent
dev.mlx5_core.0.%parent: pci3
```

   Usually, low PCI locations are closest to NUMA number 0, and high PCI locations are closest to NUMA number 1. Here is how to verify the locations:

c. Find the NIC's pcib by PCI location:

```
#> sysctl -a | grep pci.3.%
parent dev.pci.3.%parent: pcib3
```

d. Find the NIC's pcib location:

```
#> sysctl -a | grep pcib.3.%location

dev.pcib.3.%location: pci0:0:2:0 handle=\_SB_.PCI0.PEX2
```

In "handle", PCI0 is the value for locations near NUMA0, and PCI1 is the value for locations near NUMA1.

e. Find the cores list of the closest NUMA:

```
#> sysctl -a | grep "group level=\"2\"" -A 1
<group level="2" cache-level="2">
<cpu count="12" mask="fff">0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11</cpu>
--
<group level="2" cache-level="2">
<cpu count="12" mask="fff000">12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23</cpu>
```

Note: Each list of cores refers to a different NUMA.

2. Tune Mellanox NICs to work on desirable cores.

a. Pin both interrupts and application processes to the relevant cores.

b. Find the closest NUMA to the NIC

c. Find the device interrupts.

```
vmstat -ia | grep mlx5_core0 | awk '{print $1}' | sed s/irq// | sed s/://
304
305
306
…
```

d. Bind each interrupt to a core from the closest NUMA cores list
Note: It is best to avoid core number 0.

```
cpuset -x 304 -l 1
cpuset -x 305 -l 2
cpuset -x 306 -l 3
...
```

e. Bind the application to the closest NUMA cores list.
Note: It is best to avoid core number 0

```
cpuset -l 1-11 <app name> <sever flag>
cpuset -l 1-11 <app name> <client flag> <IP>
```

For best performance, change CPU's BIOS configuration to performance mode.

Due to FreeBSD internal card memory allocation mechanism on boot, it is preferred to insert the NIC to a NUMA-0 slot for max performance.

Appendix A: Sysctl Configuration and Counters

This appendix provides detailed information about configuration options and statistics exposed in Mellanox FreeBSD driver.

A.1 Configuration Options

A.1.1 General

Table 7 - Configuration Options - General Options

sysctrl Name	Description
hw.mlx5.calibr.normal	Recalibration interval during normal operations
hw.mlx5.calibr.fast	Recalibration interval during initial calibration
hw.mlx5.calibr.duration	Duration of initial calibration
hw.mlx5.relaxed_ordering_write	Set to enable relaxed ordering for PCIe writes
hw.mlx5.fast_unload_enabled	Set to enable fast unload. Clear to disable.
hw.mlx5.prof_sel	profile selector. Valid range 0 - 2
hw.mlx5.debug_mask	debug mask: 1 = dump cmd data, 2 = dump cmd exec time, 3 = both. Default=0
hw.mlx5.sw_reset_timeout	Minimum timeout in seconds between two firmware resets
hw.mlx5.fw_reset_enable	Enable firmware reset

A.1.2 Ethernet

Table 8 - Configuration Options - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name><ifnet_name>.conf.eeprom_info	Enable the dump of the inserted modules' EEPROM info (in hex)	RW
dev.<ifnet_name>.conf.device_name	IB device name associated with the ethernet device	RO
dev.<ifnet_name>.conf.uc_local_lb	Enable/Disable local loopback support for unicast	RW
dev.<ifnet_name>.conf.mc_local_lb	Enable/Disable local loopback support for multicast	RW

Table 8 - Configuration Options - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name>.conf.hw_mtu	The current MTU value set for the adapter (maximum MTU size is 9000)	RW
dev.<ifnet_name>.conf.modify_rx_dma	Enable/Disable the ability to receive packets	RW
dev.<ifnet_name>.conf.modify_tx_dma	Enable/Disable the ability to transmit packets	RW
dev.<ifnet_name>.conf.cqe_zipping	Enables/Disables CQE compression. Compressing reduces PCI overhead by coalescing and compressing multiple CQEs into a single merged CQE. Successful compressing improves message rate especially for small packet traffic Disabled by default	RW
dev.<ifnet_name>.conf.hw_lro	Enables/Disables hardware LRO support	RW
dev.<ifnet_name>.conf.tx_completion_fact_max	The max completion factor that can be used	RW
dev.<ifnet_name>.conf.tx_completion_fact	The number TX packets to send before receiving a completion interrupt	RW
dev.<ifnet_name>.conf.tx_bufring_disable	Enables/Disables the allocation of the TX DRBR	RW
dev.<ifnet_name>.conf.tx_coalesce_mode	Sets the TX Coalesce mode to EQE or CQE. EQE mode causes the coalesce timer to restart on event generation. CQE mode causes the coalesce timer to restart on CQE generation	RW
dev.<ifnet_name>.conf.tx_coalesce_pkts	The maximum number of TX packets to coalesce before transmitting the data	RW
dev.<ifnet_name>.conf.tx_coalesce_usecs	The maximum number of usecs to wait prior to transmitting the current coalesced TX packets	RW
dev.<ifnet_name>.conf.rx_coalesce_mode	Sets the RX Coalesce mode to EQE or CQE. EQE mode causes the coalesce timer to restart on event generation. CQE mode causes the coalesce timer to restart on CQE generation	RW
dev.<ifnet_name>.conf.rx_coalesce_pkts	The maximum number of RX packets to coalesce before passing the data up the stack	RW
dev.<ifnet_name>.conf.rx_coalesce_usecs	The maximum number of usecs to wait prior to sending the current coalesced RX packets up the stack	RW
dev.<ifnet_name>.conf.coalesce_pkts_max	Maximum number of packets to coalese before processing	RW

Table 8 - Configuration Options - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name>.conf.coalesce_usecs_max	Maximum number of usecs to wait before processing the coalesced packets	RW
dev.<ifnet_name>.conf.channels	The numer of RX and TX channels (rings) to create for driver utilization	RW
dev.<ifnet_name>.conf.rx_queue_size	The default number of RX buffers created per channel	RW
dev.<ifnet_name>.conf.tx_queue_size	The default number of TX buffers created per channel	RW
dev.<ifnet_name>.conf.rx_queue_size_max	The maximum value possible for rx_queue_size	RW
dev.<ifnet_name>.conf.tx_queue_size_max	The maximum value possible for tx_queue_size	RW
dev.<ifnet_name>.rx_pauseframe_control	Enables/Disables receive pause frames	RW
dev.<ifnet_name>.tx_pauseframe_control	Enables/Disables transmit pause frames	RW
hw.mlx5.calibr.normal	Recalibration interval during normal operations	RW
hw.mlx5.calibr.fast	Recalibration interval during initial calibration	RW
hw.mlx5.calibr.duration	Duration of initial calibration	RW
hw.mlx5.relaxed_ordering_write	Set to enable relaxed ordering for PCIe writes	RW
hw.mlx5.fast_unload_enabled	Set to enable fast unload. Clear to disable.	RW
hw.mlx5.prof_sel	profile selector. Valid range 0 - 2	RW
hw.mlx5.debug_mask	debug mask: 1 = dump cmd data, 2 = dump cmd exec time, 3 = both. Default=0	RW
hw.mlx5.sw_reset_timeout	Minimum timeout in seconds between two firmware resets	RW
hw.mlx5.fw_reset_enable	Enable firmware reset	RW

A.1.3 InfiniBand/RDMA

Table 9 - Configuration Options - InfiniBand/RDMA

sysctrl Name	Description	Read/ Write
sys.class.infiniband.<device_name>.default_roce_mode_port1	The default RoCE mode when attempting new connections.	RW

A.2 Statistical Counters

A.2.1 Ethernet

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name><ifnet_name>.rxstat0.wqe_err	Total number of RX WQE errors encountered across all RX channels	RO
dev.<ifnet_name>.rxstat0.sw_lro_flushed	Total number of LRO packets sent to the OS networking stack across all RX channels	RO
dev.<ifnet_name>.rxstat0.sw_lro_queued	Total number of LRO packets aggregated by the driver across all RX channels.	RO
dev.<ifnet_name>.rxstat0.lro_bytes	Total number of LRO bytes successfully received across all RX channels	RO
dev.<ifnet_name>.rxstat0.lro_packets	Total number of LRO packets successfully received across all RX channels	RO
dev.<ifnet_name>.rxstat0.csum_none	Total number of RX packets with no checksum offload across all RX channels.	RO
dev.<ifnet_name>.rxstat0.packets	Total number of RX packets across all RX channels.	RO
dev.<ifnet_name>.rxstat0.bytes	Total number of received bytes across all RX channels.	RO
dev.<ifnet_name>.txstat0tc0.nop	Total number of TX NOPs processed across all TX channels.	RO
dev.<ifnet_name>.txstat0tc0.dropped	Total number of TX packets dropped across all TX channels.	RO
dev.<ifnet_name>.txstat0tc0.defragged	Total number of defraggmented buffers across all TX channels.	RO
dev.<ifnet_name>.txstat0tc0.csum_offload_-none	Total number of TX packets processed without checksum offload across all TX channels.	RO
dev.<ifnet_name>.txstat0tc0.tso_bytes	Total number of TSO bytes across all TX channels.	RO
dev.<ifnet_name>.txstat0tc0.tso_packets	Total number of TSO packets across all TX channels	RO
dev.<ifnet_name>.txstat0tc0.packets	Total number of TX packets across all TX channels.	RO
dev.<ifnet_name>.txstat0tc0.bytes	Total number of transmitted bytes across all TX channels.	RO
dev.<ifnet_name>.pstats.collisions	Total number of collisions.	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name>.pstats.jabbers	Total number of packets received that were longer than 1518 octets and had either a bad FCS or alignment error.	RO
dev.<ifnet_name>.pstats.fragments	Total number of packets received that were less than 64 octets and had either a bad FCS or alignment error.	RO
dev.<ifnet_name>.pstats.oversize_pkts	Total number of packets received that were longer than 1518 octets long	RO
dev.<ifnet_name>.pstats.undersize_pkts	Total number of packets received that were less than 64 octets long	RO
dev.<ifnet_name>.pstats.crc_align_errors	Total number of packets received that had either a bad FCS or alignment error.	RO
dev.<ifnet_name>.pstats.multicast_pkts	Total number of good multicast packets received.	RO
dev.<ifnet_name>.pstats.broadcast_pkts	Total number of good broadcast packets received.	RO
dev.<ifnet_name>.pstats.pkts	Total number of packets (including bad, broadcast, and multicast) received on the network	RO
dev.<ifnet_name>.pstats.octets	Total number of octets of data (including errors) received on the network	RO
dev.<ifnet_name>.pstats.drop_events	Total number of events in which packets were dropped by the probe due to lack of resources	RO
dev.<ifnet_name>.pstats.pause_ctrl_tx	Total number of pause control frames sent.	RO
dev.<ifnet_name>.pstats.pause_ctrl_rx	Total number of pause control frames received.	RO
dev.<ifnet_name>.pstats.unsupported_op_rx	The numbrer of mac control frames received that contain an opcode that is not supported.	RO
dev.<ifnet_name>.pstats.mac_control_rx	The number of mac control frames received	RO
dev.<ifnet_name>.pstats.mac_control_tx	The number of mac control frames transmitted.	RO
dev.<ifnet_name>.pstats.symbol_err	Total number of symbol errors	RO
dev.<ifnet_name>.pstats.too_long_errors	The number of frames received that exceeded the MTU size.	RO
dev.<ifnet_name>.pstats.out_of_range_len	The number of frames received with a length field value greater than the maximum allowed logical length control (LLC) data size.	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name>.pstats.in_range_len_errors	A count of frames with a length/type field value between the minimum unpadded MAC client data size and the maximum allowed MAC client data size, inclusive, that does not match the number of MAC client data octets received.	RO
dev.<ifnet_name>.pstats.broadcast_rx	Total number of broadcast packets successfully received	RO
dev.<ifnet_name>.pstats.multicast_rx	Total number of multicast packets successfully received	RO
dev.<ifnet_name>.pstats.broadcast_xmitted	Total number of broadcast bytes successfully transmitted	RO
dev.<ifnet_name>.pstats.multicast_xmitted	Total number of multicast packets successfully transmitted	RO
dev.<ifnet_name>.pstats.octets_received	Total number of octets received	RO
dev.<ifnet_name>.pstats.octets_tx	Total number of octets transmitted	RO
dev.<ifnet_name>.pstats.alignment_err	Total number of packet alignment errors. i.e. The number of bits received is an uneven byte count, or there is a Frame Check Sequence (FCS) Error.	RO
dev.<ifnet_name>.pstats.check_seq_err	Total number of FCS errors	RO
dev.<ifnet_name>.pstats.frames_rx	Total number of frames received	RO
dev.<ifnet_name>.pstats.frames_tx	Total number of frames transmitted.	RO
dev.<ifnet_name>.vstats.rx_wqe_err	Total number of RX WQEs that completed in error.	RO
dev.<ifnet_name>.vstats.tx_defragged	The number of times the TX mbufs were unable to be loaded via bus_d-mamap_load due to an excessive amount of mbufs in the chain.	RO
dev.<ifnet_name>.vstats.tx_queue_dropped	Total number of TX packets dropped by the driver prior transmission due to an error.	RO
dev.<ifnet_name>.vstats.tx_csum_offload	Total number of transmit packets that successfully used checksum offloading	RO
dev.<ifnet_name>.vstats.rx_csum_none	Total number of received packets that did not have any checksum offload indication	RO
dev.<ifnet_name>.vstats.rx_csum_good	Total number of received packets that had the checksum successfully off-loaded.	RO
dev.<ifnet_name>.vstats.sw_lro_flushed	Total number of LRO packets sent to the OS networking stack.	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name>.vstats.sw_lro_queued	Total number of LRO packets aggregated by the driver.	RO
dev.<ifnet_name>.vstats.lro_bytes	Total number of LRO bytes successfully received	RO
dev.<ifnet_name>.vstats.lro_packets	Total number of LRO packets successfully received	RO
dev.<ifnet_name>.vstats.tso_bytes	Total number of TSO bytes successfully transmitted	RO
dev.<ifnet_name>.vstats.tso_packets	Total number of TSO packets successfully transmitted	RO
dev.<ifnet_name>.vstats.rx_out_of_buffer	Total number of packets dropped due to lack of network buffers	RO
dev.<ifnet_name>.vstats.tx_broadcast_bytes	Total number of broadcast bytes successfully transmitted	RO
dev.<ifnet_name>.vstats.tx_broadcast_packets	Total number of broadcast packets successfully transmitted	RO
dev.<ifnet_name>.vstats.rx_broadcast_bytes	Total number of broadcast bytes successfully received	RO
dev.<ifnet_name>.vstats.rx_broadcast_packets	Total number of broadcast packets successfully received	RO
dev.<ifnet_name>.vstats.tx_multicast_bytes	Total number of multicast bytes successfully transmitted	RO
dev.<ifnet_name>.vstats.tx_multicast_packets	Total number of multicat packets successfully transmitted	RO
dev.<ifnet_name>.vstats.rx_multicast_bytes	Total number of multicast bytes successfully received	RO
dev.<ifnet_name>.vstats.rx_multicast_packets	Total number of multicast packets successfully received	RO
dev.<ifnet_name>.vstats.tx_unicast_bytes	Total number of unicast bytes successfully transmitted	RO
dev.<ifnet_name>.vstats.tx_unicast_packets	Total number of unicast packets successfully transmitted	RO
dev.<ifnet_name>.vstats.rx_unicast_bytes	Total number of unicast bytes successfully received	RO
dev.<ifnet_name>.vstats.rx_unicast_packets	Total number of unicast packets successfully received	RO
dev.<ifnet_name>.vstats.tx_error_bytes	Total number of transmit error bytes	RO
dev.<ifnet_name>.vstats.tx_error_packets	Total number of transmit error packets	RO
dev.<ifnet_name>.vstats.rx_error_bytes	Total number of received error bytes	RO
dev.<ifnet_name>.vstats.rx_error_packets	Total number of received error packets	RO
dev.<ifnet_name>.vstats.tx_bytes	Total number of bytes successfully transmitted	RO
dev.<ifnet_name>.vstats.tx_packets	Total number of packets successfully transmitted	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
dev.<ifnet_name>.vstats.rx_bytes	Total number of bytes successfully received	RO
dev.<ifnet_name>.vstats.rx_packets	Total number of packets successfully received	RO
dev.<ifnet_name>.conf.debug_stats	Enables the additional debug_stats found below:	RW
	dev.mce.0.debug_stats.error_counter_lane[0-15]: 0	RO
	dev.mce.0.debug_stats.fatal_err_msg_sent: 0	RO
	dev.mce.0.debug_stats.non_fatal_err_msg_sent: 0	RO
	dev.mce.0.debug_stats.correctable_err_msg_sent: 0	RO
	dev.mce.0.debug_stats.config_cycle64usec: 0	RO
	dev.mce.0.debug_stats.config_cycle16to63usec: 376618	RO
	dev.mce.0.debug_stats.config_cycle8to15usec: 139635	RO
	dev.mce.0.debug_stats.config_cycle2to7usec: 26968634	RO
	dev.mce.0.debug_stats.config_cycle1usec: 14311	RO
	dev.mce.0.debug_stats.dl_down: 17	RO
	dev.mce.0.debug_stats.times_in_l23: 17	RO
	dev.mce.0.debug_stats.times_in_l1: 0	RO
	dev.mce.0.debug_stats.perst_handler: 18	RO
	dev.mce.0.debug_stats.time_to_iron_image_start: 105773	RO
	dev.mce.0.debug_stats.time_to_plastic_image_start: 120126	RO
	dev.mce.0.debug_stats.time_to_crs_en: 105825	RO
	dev.mce.0.debug_stats.time_to_l0: 102883	RO
	dev.mce.0.debug_stats.time_to_detect_state: 102871	RO
	dev.mce.0.debug_stats.time_to_first_perst: 102533	RO
	dev.mce.0.debug_stats.calibration_time: 64815	RO
	dev.mce.0.debug_stats.time_to_link_image: 15560	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
	dev.mce.0.debug_stats.time_to_boot_image_start: 5735	RO
	dev.mce.0.debug_stats.outbound_stalled_writes_events: 0	RO
	dev.mce.0.debug_stats.outbound_stalled_reads_events: 0	RO
	dev.mce.0.debug_stats.outbound_stalled_writes: 0	RO
	dev.mce.0.debug_stats.outbound_stalled_reads: 0	RO
	dev.mce.0.debug_stats.crc_error_tlp: 0	RO
	dev.mce.0.debug_stats.crc_error_dllp: 0	RO
	dev.mce.0.debug_stats.l0_to_recovery_retrain: 0	RO
	dev.mce.0.debug_stats.l0_to_recovery_framing: 0	RO
	dev.mce.0.debug_stats.l0_to_recovery_ts: 36	RO
	dev.mce.0.debug_stats.l0_to_recovery_eieos: 53	RO
	dev.mce.0.debug_stats.tx_errors: 89	RO
	dev.mce.0.debug_stats.rx_errors: 0	RO
	dev.mce.0.debug_stats.tx_overflow_buffer_marked_pkt: 0	RO
	dev.mce.0.debug_stats.tx_overflow_buffer_pkt: 0	RO
	dev.mce.0.debug_stats.life_time_counter: 591311368723210	RO
	dev.mce.0.debug_stats.phy_corrected_bits_lane[0-3]: 0	RO
	dev.mce.0.debug_stats.phy_corrected_bits: 0	RO
	dev.mce.0.debug_stats.phy_symbol_errors: 0	RO
	dev.mce.0.debug_stats.phy_received_bits: 5417449000000000	RO
	dev.mce.0.debug_stats.phy_time_since_last_clear: 54174490	RO
	dev.mce.0.debug_stats.tx_stat_p8192to10239octets: 0	RO
	dev.mce.0.debug_stats.tx_stat_p4096to8191octets: 0	RO
	dev.mce.0.debug_stats.tx_stat_p2048to4095octets: 0	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
	dev.mce.0.debug_stats.tx_stat_p1519to2047octets: 0	RO
	dev.mce.0.debug_stats.tx_stat_p1024to1518octets: 0	RO
	dev.mce.0.debug_stats.tx_stat_p512to1023octets: 0	RO
	dev.mce.0.debug_stats.tx_stat_p256to511octets: 0	RO
	dev.mce.0.debug_stats.tx_stat_p128to255octets: 1776	RO
	dev.mce.0.debug_stats.tx_stat_p65to127octets: 1	RO
	dev.mce.0.debug_stats.tx_stat_p64octets: 0	RO
	dev.mce.0.debug_stats.ex_reserved_1: 0	RO
	dev.mce.0.debug_stats.ex_reserved_0: 0	RO
	dev.mce.0.debug_stats.rx_icrc_encapsulated: 0	RO
	dev.mce.0.debug_stats.rx_buffer_full: 18	RO
	dev.mce.0.debug_stats.rx_buffer_almost_full: 0	RO
	dev.mce.0.debug_stats.tx_ebp: 0	RO
	dev.mce.0.debug_stats.rx_ebp: 0	RO
	dev.mce.0.debug_stats.no_buffer_discard_mc: 0	RO
	dev.mce.0.debug_stats.ecn_marked: 0	RO
	dev.mce.0.debug_stats.port_transmit_wait: 214	RO
	dev.mce.0.debug_stats.rs_corrected_symbols_lane[0-3]: 0	RO
	dev.mce.0.debug_stats.rs_corrected_symbols_total: 0	RO
	dev.mce.0.debug_stats.rs_single_error_blocks: 0	RO
	dev.mce.0.debug_stats.rs_no_errors_blocks: 0	RO
	dev.mce.0.debug_stats.rs_uncorrectable_blocks: 0	RO
	dev.mce.0.debug_stats.rs_corrected_blocks: 0	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
	dev.mce.0.debug_stats.fc_corrected_blocks_lane[0-3]: 0	RO
	dev.mce.0.debug_stats.edpl_bip_errors_lane[0-3]: 0	RO
	dev.mce.0.debug_stats.sync_headers_errors: 0	RO
	dev.mce.0.debug_stats.symbol_errors: 0	RO
	dev.mce.0.debug_stats.time_since_last_clear: 54174490	RO
	dev.mce.0.debug_stats.out_broadcast_pkts: 0	RO
	dev.mce.0.debug_stats.out_multicast_pkts: 1777	RO
	dev.mce.0.debug_stats.in_broadcast_pkts: 0	RO
	dev.mce.0.debug_stats.in_multicast_pkts: 1812	RO
	dev.mce.0.debug_stats.out_errors: 0	RO
	dev.mce.0.debug_stats.out_discards: 0	RO
	dev.mce.0.debug_stats.out_ucast_pkts: 0	RO
	dev.mce.0.debug_stats.out_octets: 227942	RO
	dev.mce.0.debug_stats.in_unknown_protos: 0	RO
	dev.mce.0.debug_stats.in_errors: 0	RO
	dev.mce.0.debug_stats.in_discards: 0	RO
	dev.mce.0.debug_stats.in_ucast_pkts: 0	RO
	dev.mce.0.debug_stats.in_octets: 231872	RO
	dev.mce.0.debug_stats.p8192to10239octets: 0	RO
	dev.mce.0.debug_stats.p4096to8191octets: 0	RO
	dev.mce.0.debug_stats.p2048to4095octets: 0	RO
	dev.mce.0.debug_stats.p1519to2047octets: 0	RO
	dev.mce.0.debug_stats.p1024to1518octets: 0	RO
	dev.mce.0.debug_stats.p512to1023octets: 0	RO

Table 10 - Statistical Counters - Ethernet

sysctrl Name	Description	Read/ Write
	dev.mce.0.debug_stats.p256to511octets: 0	RO
	dev.mce.0.debug_stats.p128to255octets: 1811	RO
	dev.mce.0.debug_stats.p65to127octets: 0	RO
	dev.mce.0.debug_stats.p64octets: 1	RO
dev.mce.0.vstats.tx_jumbo_packets	TX packets greater than 1518 octets	RO

A.2.2 InfiniBand/RDMA

Table 11 - Statistical Counters - InfiniBand/RDMA

sysctrl Name	Description	Read/ Write
sys.class.infiniband.<device_name>.reg_pages	Total number of OS pages registered for IB/RoCE use	RO
sys.class.infiniband.<device_name><device_name>.fw_pages	Total number of OS pages used by firmware	RO
sys.class.infiniband.<device_name>.board_id	The PSID of the board	RO
sys.class.infiniband.<device_name>.hca_type	The device family of the adapter	RO
sys.class.infiniband.<device_name>.hw_rev	The hardware revision of the adapter	RO
sys.class.infiniband.<device_name>.ports.<port_num>..<port_-num>.hw_counters.lifespan		RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.local_ack_timeout_err	Total number of No ACK responses within the timer interval. Supported only for the XRC, DC and RC transports	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.implied_nak_seq_err	Total number of times the requester detected an ACK with a PSN larger expected for an RDMA READ or ATOMIC response. Supported only for the XRC, DC and RC transports	RO

Table 11 - Statistical Counters - InfiniBand/RDMA

sysctrl Name	Description	Read/ Write
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.packet_seq_err	Total number of received NAK-Sequence error packets. Supported only for the XRC, DC and RC transports	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.rnr_nak_retry_err	Total number of received RNR NAK packets on this port. Supported only for the XRC, DC and RC transports	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.duplicate_request	Total number of duplicate request packets received on this port. Supported only for XRC, DC, and RC transports	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.out_of_sequence	Total number of out of sequence packets received on this port. Supported only for XRC, DC, and RC transports	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.out_of_buffer	Total number of packet drops that occurred due to lack of WQEs	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.rx_atomic_requests	Total number of Atomic operation requests on this port	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.rx_read_requests	Total number of RDMA Read requests on this port	RO
sys.class.infiniband.<device_name>.ports.<port_-num>.hw_counters.rx_write_requests	Total number of RDMA Write requests on this port	RO
sys.class.infiniband.<device_name>.ports.<port_num>.pkeys.<pkey index>	The partion key defined at location <pkey index>	RO
sys.class.infiniband.<device_name>.ports.<port_num>.gids.<gid index>	The gid index defined at location <gid index>	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.multicast_xmit_packets	Total number of IB/RoCE multicast packets transmitted by the port.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.multicast_rcv_packets	Total number of IB/RoCE multicast packets received by the port.	RO

Table 11 - Statistical Counters - InfiniBand/RDMA

sysctrl Name	Description	Read/ Write
sys.class.infiniband.<device_name>.ports.<port_num>.counters.unicast_xmit_packets	Total number of IB/RoCE unicast packets transmitted by the port.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.unicast_rcv_packets	Total number of IB/RoCE unicast packets received by the port.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_rcv_packets	Total number of ticks the port is unable to send data due to insufficient credits or lack of arbitration.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_xmit_wait	Total number of IB/RoCE packets received by the port. Includes packets with errors.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_xmit_packets	Total number of IB/RoCE packets transmitted by the port. Includes packets with errors.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_rcv_data	Total number of IB/RoCE data octets received by the port.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_xmit_data	Total number of IB/RoCE data octets transmitted from the port.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.VL15_dropped	Total number of incoming VL15 packets discarded due to resource limitations.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.excessive_buffer_overrun_errors	Total number of times overrun errors occurred during consecutive flow control update periods.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.local_link_integrity_errors	Total number of times port physical errors exceeded the defined threshold	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_rcv_constraint_errors	Total number of IB/RoCE receive packets discarded by the port due to raw filter or partition enforcement	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_xmit_constraint_errors	Total number of IB/RoCE packets not sent from the port due to raw filter or partition enforcement	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_xmit_discards	Total number of IB/RoCE transmit packets discarded because the port is down or congested	RO

Table 11 - Statistical Counters - InfiniBand/RDMA

sysctrl Name	Description	Read/ Write
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_rcv_switch_relay_errors	Total number of IB/RoCE receive packets disarded because they could not be forwarded by the switch relay	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_rcv_remote_physical_errors	Total number of IB/RoCE packets received with the EBP delimiter marked.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.port_rcv_errors	Total number of IB/RoCE packets received that contained errors	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.link_downed	Total number of times the port training state machine failed link recovery and set the link down.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.link_error_recovery	Total number of times the port training state machine successfully completed link recovery	RO
sys.class.infiniband.<device_name>.ports.<port_num>.counters.symbol_error	Total number of physical link errors detected on the port.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.gid_attrs.types.<gid index>	The RoCE version associated with gid index <gid index>	RO
sys.class.infiniband.<device_name>.ports.<port_num>.gid_attrs.ndevs.<gid index>	The FreeBSD network device associated with gid index <gid index>	RO
sys.class.infiniband.<device_name>.ports.<port_num>.link_layer	The link layer mode configured for the specified port. Will be Ethernet or Infiniband.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.phys_state	Indicates the physical port state as defined in the IB spec.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.rate	Indicates the Infiniband port rate and speed.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.cap_mask	The supported IB capabilities available for this port as defined by the IB spec.	RO
sys.class.infiniband.<device_name>.ports.<port_num>.sm_sl	Indicates the SL used to communicate with the subnet manager (IB only)	RO
sys.class.infiniband.<device_name>.ports.<port_num>.sm_lid	Indicates the LID of the subnet manager (IB only)	RO

Table 11 - Statistical Counters - InfiniBand/RDMA

sysctrl Name	Description	Read/ Write
sys.class.infiniband.<device_name>.ports.<port_-num>.lid_mask_count	The value assigned by the subnet manager that specifies the number of path bits in the LID (IB only)	RO
sys.class.infiniband.<device_name>.ports.<port_num>.lid	The lid assigned to the local adapter port by the subnet manager (IB only)	RO
sys.class.infiniband.<device_name>.ports.<port_num>.state	The current port state as defined by the IB spec.	RO
sys.class.infiniband.<device_name>.fw_ver	The current version of firmware running on the network adapter	RO
sys.class.infiniband.<device_name>.node_desc	The IB node description	RO
sys.class.infiniband.<device_name>.node_guid	The IB node GUID	RO
sys.class.infiniband.<device_name>.sys_image_guid	The IB system image GUID	RO
sys.class.infiniband.<device_name>.node_type	Indicates the node type as defined by the IB spec. Valid values are Channel Adapter (CA), Switch, or Router	RO