EMULEX®
An Avago Technologies Company

Emulex Poll Mode Driver User Manual

10.7 Release

Revision 1.4
June 17, 2015

Emulex, 3333 Susan Street

Costa Mesa, CA 92626

Table of Contents

1 Overview

1.1 Purpose

Emulex has partnered with 6WIND® to provide an Emulex poll mode driver (PMD) for the OneConnect® OCe14000-series 10Gb and 40Gb Ethernet Network Adapters and Converged Network Adapters, delivering the power of the data plane development kit (DPDK) to next generation network functions virtualization (NFV) workloads, and enabling customers to choose the best networking connectivity for their server deployments.

The Emulex PMD enables the OneConnect Ethernet (oce) adapters to interface with the DPDK, giving customers the flexibility to choose their Ethernet connectivity for standard Intel© x86 platforms.

This document provides instructions for installing and configuring the DPDK, the Emulex PMD, and the Emulex SLI User Ready Framework (SURF™) application programming interface (API). It also lists limitations and provides operational considerations for the Emulex PMD.

1.2 Features

The Emulex PMD includes the following features:

- Concurrent iSCSI support
- Universal multi-channel (UMC) configuration support
  - PMD installation available on all four channels per port provided that the PMD is installed on three or four channels.

    Note: If you install the Emulex PMD on only one or two channels on the port, it must be installed on the third and/or fourth channel. This is necessary because freeing up the resources for the first or second channel does not release the required RSS resources.

  - Four physical functions (PFs) per port
- Single root I/O virtualization (SR-IOV) support
  - With receive side scaling (RSS) enabled, up to six virtual functions (VFs) are supported with one PF in a SR-IOV configuration. In this configuration, the PMD running on the VFs has access to two RSS rings each.
  - With RSS disabled, up to twelve VFs are supported with one PF in a SR-IOV configuration.

- In a 4-port adapter, up to six VFs are supported on each of the first three ports. The fourth port does not support any VFs because of limited resources.

Note: SR-IOV VFs are not supported if multichannel is enabled (that is, if UMC, Flex10, or NIC Partitioning (NPAR) are used).

1.3 References

Emulex Drivers for Linux User Manual

This manual includes SR-IOV configuration information. This manual is available under the various "Emulex Standard Drivers for Linux" links at emulex.com:

`http://www.emulex.com/downloads/emulex/drivers/linux/`

Emulex PMD for DPDK program information:

`http://www.emulex.com/solutions/network-connectivity-solutions/industries/telco/`

Open source DPDK project information:

`http://www.dpdk.org/`

1.4 Abbreviations

API	application programming interface
ARI	alternative requester ID interpretation
CPU	central processing unit
DPDK	data plane development kit
LLDP	Link Layer Discovery Protocol
NFV	network functions virtualization
NIC	network interface card
NPAR	NIC partitioning
oce	OneConnect Ethernet
OS	operating system
PF	physical function
PMD	poll mode driver
RHEL	Red Hat Enterprise Linux
RSS	receive side scaling
RX	receive
SR-IOV	single root I/O virtualization
SURF	SLI User Ready Framework
TX	transmit

UMC	universal multi-channel
VF	virtual function
VLAN	virtual local area network
VM	virtual machine

2 Configuring the Emulex PMD

2.1 Requirements

Table 2-1 lists the requirements for configuring the Emulex PMD for DPDK.

Note: The Emulex filenames in the following table and instructions are examples and may not correspond to the latest filenames that may be in your OEM package or on the Emulex website. For the latest Emulex product numbers, versions, and files, please contact your Emulex systems engineer.

Table 2-1 Emulex PMD for DPDK Configuration Requirements

Requirements
RHEL 6.5 64-bit
Emulex OCe14000-series (Skyhawk) adapter
Emulex PMD for DPDK package (such as, Palau_10.4.218.0_DPDK_PMD_Internal.zip) • PMD files • SURF source files
Emulex Linux Ethernet (be2net) Source Driver files (format of filename is "be2net-<build version>.src.rpm", such as "be2net-10.2.470.14-1.src.rpm")
Emulex OneConnect Firmware (format of filename is "OneConnect-Flash-<build version>.iso", such as "OneConnect-Flash-10.4.240.0-x64.iso")
DPDK Version 1.7.1 or 1.8.0

2.2 Basic Set-Up

2.2.1 Download the Required Files

Download the following files:

- dpdk-1.7.1.tar.gz or dpdk-1.8.0.tar.gz

  You can download these files directly from the dpdk.org site, for example:

  `http://www.dpdk.org/browse/dpdk/snapshot/dpdk-1.8.0.tar.gz`

  Alternatively, you can access the files by going to http://www.dpdk.org/download, and clicking on the "browsing interface" link in:

  "Other versions and formats are available from the browsing interface."

- Emulex Linux Ethernet (be2net) Source Driver files (be2net-<build version>.src.rpm)

  These files are available at one of the following locations:

  - In your OEM package

- Emulex website: Downloads>Standard>Linux>RHEL 6/CentOS 6: Ethernet driver: Source driver kit (be2net-10.2.470.14-1.src.rpm):

  `http://www.emulex.com/downloads/emulex/drivers/linux/rhel-6-centos-6/drivers`

- Emulex OneConnect Firmware (OneConnect-Flash-10.4.240.0-x64.iso)
- Emulex PMD for DPDK Package (Palau_10.4.218.0_DPDK_PMD_Internal.zip)

2.2.2 Flash the Firmware

1. Flash the Emulex firmware to your OCe14000-series adapter:
   a. Locate the OneConnect-Flash-10.4.240.0-x64.iso file.
   b. Burn the OneConnect-Flash-10.4.240.0-x64.iso file to a bootable CD of DVD.
   c. Boot the newly-created CD/DVD.
   d. Flash the firmware onto your OCe14000-series adapter by answering 'y' to the prompts seen after booting the CD/DVD.
2. Reboot the system under test.
3. Configure the OCe14000-series adapter to use the "NIC-Only" profile by using the PXE Boot BIOS.
4. Install RHEL 6.5, 64-bit, with the "Software Development Workstation" option.
5. Log in as root and remove the in-box be2net.ko driver:

   ```
   rm /lib/modules/2.6.32-431.el6.x86_64/kernel/drivers/net/benet/be2net.ko
   ```

6. Reboot, log in as root, and verify that be2net.ko is not loaded:

   ```
   lsmod | grep be2net
   ```

2.2.3 Install Linux Ethernet (be2net) Source Driver Files

On the system under test, with the appropriate file (be2net-<build version>.src.rpm) downloaded from step 1 in section 2.2, "Basic Set-Up", install the source driver files using the rpm command to a directory of your choice. This directory corresponds to the "<be2net driver directory>" variable in the export command line in section 2.3.2, "Build the Emulex SURF and PMD Binaries," on page 10 and the cp command line in section 2.3.3, "Copy the Binaries," on page 10.

2.3 Build and Copy the Binaries

2.3.1 Build the DPDK Binaries

On the system under test, with the dpdk-1.8.0.tar.gz file downloaded from step 1 in section 2.2, "Basic Set-Up", build the DPDK binaries:

```
cd /
cp dpdk-1.8.0.tar.gz .
gunzip dpdk-1.8.0.tar.gz
tar –xvf dpdk-1.8.0.tar
```

```
cd dpdk-1.8.0
make config T=x86_64-native-linuxapp-gcc
export RTE_SDK=`pwd`
export RTE_TARGET=build
make
```

2.3.2 Build the Emulex SURF and PMD Binaries

Use the Palau_10.4.218.0_DPDK_PMD_Internal.zip file downloaded from step 1 in section 2.2, "Basic Set-Up", to build the SURF and PMD binaries:

```
cd /
mkdir 10.4.218.0
cd 10.4.218.0
cp Palau_10.4.218.0_DPDK_PMD_Internal.zip .
unzip Palau_10.4.218.0_DPDK_PMD_Internal.zip
cd packages
cd DPDK_PMD
cd surf_linux
export ELX_SURF_HUB=`pwd`/surf_hub
export BE2NET=<be2net driver directory>
chmod 777 compile_elx_pmd_dpdk
./compile_elx_pmd_dpdk
```

2.3.3 Copy the Binaries

```
cd /
mkdir DPDK_RH65.218.0
cd /10.4.218.0
cd packages
cd DPDK_PMD
cp surf_linux/surf_hub/surf_hub.ko                    /DPDK_RH65.218.0
cp surf_linux/surf_provider/surf_provider.ko          /DPDK_RH65.218.0
cp surf_linux/PMD_DRIVER/librte_pmd_oce.so            /DPDK_RH65.218.0
cp surf_linux/PMD_DRIVER/dpdk_oce_surf/dpdk_surf.ko   /DPDK_RH65.218.0
cp /<be2net driver directory>/be2net.ko               /DPDK_RH65.218.0
cp /dpdk-1.8.0/build/app/testpmd                      /DPDK_RH65.218.0
```

2.4 Parameters for dpdk_surf.ko Module

When loading the dpdk_surf.ko module, the following command line parameters may be specified.

2.4.1 dpdk_oce_trace <int>

The dpdk_oce_trace parameter specifies the display of debug traces by the dpdk_surf.ko module:

- 0x0 – module tracing disabled (default value).
- 0x1 – displays debug messages.
- 0x2 – displays error messages.
- 0x3 – displays all messages.

2.4.2 lldp_mode <int>

The lldp_mode parameter specifies the behavior of the Link Layer Discovery protocol (LLDP) on the OCe14000-series adapter ports:

- 0x0 – LLDP disabled (default value).
- 0x1 – enables periodic transmission of LLDP frames.
- 0x2 – enables processing of input LLDP frames.

2.4.3 max_dpdk_oce_port <int>

The max_dpdk_oce_port parameter is the maximum number of OCe14000-series adapter ports to support. The default is 8.

2.4.4 max_rxq_per_port <int>

The max_rxq_per_port parameter specifies the maximum number of RX queues per port provided to the Emulex PMD. The default value is 0.

Note: Because of the limited resources on an OCe14000-series adapter, creating a single RSS-enabled RX queue may fail in SR-IOV with more than six VFs enabled per port. To support seven to 12 VFs per port, you must request a single non-RSS RX queue by setting max_rxq_per_port to 1. This forces the Emulex PMD to create only one non-RSS RX queue per OCe14000-series adapter port.

2.5 Create the load_drivers and run_testpmd Scripts

Create the following load_drivers and run_testpmd scripts to simplify the loading and execution processes.

2.5.1 Create the load_drivers Script

Use the following steps to create the load_drivers script.

1. First gather the Linux Ethernet interface names for your adapter ports:

```
cd /DPDK_RH65.218.0
insmod be2net.ko
ifconfig | grep Ethernet
rmmod be2net.ko
```

2. Create the /DPDK_RH65.218.0/load_drivers script and modify the Linux Ethernet interface names (such as, $INTERFACE_1 and $INTERFACE_2) for your environment:

```
#!/bin/bash

# Linux SkyHawk interface names
INTERFACE_1=em1     ## Modify this to match your environment!
INTERFACE_2=em2     ## Modify this to match your environment!

# Directory containing the files
BINARIES=/DPDK_RH65.218.0

# Huge page allocation
mkdir -p /mnt/huge
mount -t hugetlbfs nodev /mnt/huge
echo 128 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_hugepages
echo 128 > /sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr_hugepages
echo 128 > /sys/devices/system/node/node2/hugepages/hugepages-2048kB/nr_hugepages
echo 128 > /sys/devices/system/node/node3/hugepages/hugepages-2048kB/nr_hugepages

# Load the be2net driver:
insmod $BINARIES/be2net.ko

/* For SR-IOV configurations, install the be2net driver with the required number of
VFs. For RSS support, <num_vfs> is 6 or less. If RSS is disabled, <num_vfs> is 12 or
less. */
# insmod korg/be2net.ko num_vfs=<num_vfs>

# Load the SURF hub and provider modules:
insmod $BINARIES/surf_hub.ko
insmod $BINARIES/surf_provider.ko

ip link set $INTERFACE_1 up
```

```
ethtool -s $INTERFACE_1 speed 10000 duplex full
ip link set $INTERFACE_2 up
ethtool -s $INTERFACE_2 speed 10000 duplex full

/* Reduce the number of RX & TX queues that the be2net driver uses. This frees up the
queues and allows the PMD to use them */
ethtool -L $INTERFACE_1 combined 1
ethtool -L $INTERFACE_2 combined 1

/*Since all traffic will be handled by the PMD, bring down the NIC driver interface */
ifconfig $INTERFACE_1 down
ifconfig $INTERFACE_2 down

/* Assign a trivial and unique address to the NIC interfaces to prevent a MAC address
collision in the adapter's tables */
ifconfig $INTERFACE_1 hw ether 00:00:00:00:01
ifconfig $INTERFACE_2 hw ether 00:00:00:00:02

# start the DPDK SURF kernel module with RSS capability enabled by default
# Up to 6 VFs are supported with RSS enabled
insmod $BINARIES/dpdk_surf.ko
# Up to 12 VFs are supported with RSS disabled using the following command:
# insmod $BINARIES/dpdk_surf.ko max_rxq_per_port=1
```

3. Mark the load_drivers script executable:

```
chmod 777 /DPDK_RH65.218.0/load_drivers
```

2.5.2 Create the run_testpmd Script

Use the following steps to create the run_testpmd script.

1. Determine the PCI device IDs for your two adapter ports:

```
lspci | grep Emulex
```

2. Create the /DPDK_RH65.218.0/run_testpmd script:

```
#!/bin/bash

BINARIES=/DPDK_RH65.218.0

P0=0000:01:00.00     ## Modify this to match your environment!
P1=0000:01:00.01     ## Modify this to match your environment!

./testpmd -c ff -n 3 -d /$BINARIES/librte_pmd_oce.so -w $P0 -w $P1 -- -i
--nb-cores=2 --nb-ports=2
```

3. Mark the run_testpmd script executable:

```
chmod 777 /DPDK_RH65.218.0/run_testpmd
```

2.6 Run a Test

Use the following steps to run a test with the scripts.

1. Verify the correct cabling:
   a. Connect port 0 directly to port 1 on your adapter.
   b. If possible, verify that both link LEDs are on.
2. Generate the DPDK-PMD traffic:

```
cd /DPDK_RH65.218.0
./load_drivers
./run_testpmd
start tx_first
<<< NOTE: Wait for about 10-15 seconds >>>
stop
quit
```

2.6.1 Sample Output

```
EAL: Detected lcore 0 as core 0 on socket 0
EAL: Detected lcore 1 as core 0 on socket 1
EAL: Detected lcore 2 as core 1 on socket 0
EAL: Detected lcore 3 as core 1 on socket 1
EAL: Detected lcore 4 as core 2 on socket 0
EAL: Detected lcore 5 as core 2 on socket 1
EAL: Detected lcore 6 as core 3 on socket 0
EAL: Detected lcore 7 as core 3 on socket 1
EAL: Detected lcore 8 as core 4 on socket 0
EAL: Detected lcore 9 as core 4 on socket 1
EAL: Detected lcore 10 as core 5 on socket 0
EAL: Detected lcore 11 as core 5 on socket 1
EAL: Detected lcore 12 as core 8 on socket 0
EAL: Detected lcore 13 as core 8 on socket 1
EAL: Detected lcore 14 as core 9 on socket 0
EAL: Detected lcore 15 as core 9 on socket 1
EAL: Detected lcore 16 as core 10 on socket 0
EAL: Detected lcore 17 as core 10 on socket 1
EAL: Detected lcore 18 as core 11 on socket 0
EAL: Detected lcore 19 as core 11 on socket 1
EAL: Detected lcore 20 as core 12 on socket 0
EAL: Detected lcore 21 as core 12 on socket 1
EAL: Detected lcore 22 as core 13 on socket 0
EAL: Detected lcore 23 as core 13 on socket 1
EAL: Support maximum 64 logical core(s) by configuration.
EAL: Detected 24 lcore(s)
EAL: Setting up memory...
EAL: Ask a virtual area of 0xc800000 bytes
EAL: Virtual area found at 0x7f9d18c00000 (size = 0xc800000)
EAL: Ask a virtual area of 0x1000000 bytes
EAL: Virtual area found at 0x7f9d17a00000 (size = 0x1000000)
```

```
EAL: Ask a virtual area of 0x2000000 bytes
EAL: Virtual area found at 0x7f9d15800000 (size = 0x2000000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d15200000 (size = 0x400000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d14c00000 (size = 0x400000)
EAL: Ask a virtual area of 0x1800000 bytes
EAL: Virtual area found at 0x7f9d13200000 (size = 0x1800000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d12c00000 (size = 0x400000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d12600000 (size = 0x400000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d12000000 (size = 0x400000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d11a00000 (size = 0x400000)
EAL: Ask a virtual area of 0xe00000 bytes
EAL: Virtual area found at 0x7f9d10a00000 (size = 0xe00000)
EAL: Ask a virtual area of 0x1000000 bytes
EAL: Virtual area found at 0x7f9d0f800000 (size = 0x1000000)
EAL: Ask a virtual area of 0x1c00000 bytes
EAL: Virtual area found at 0x7f9d0da00000 (size = 0x1c00000)
EAL: Ask a virtual area of 0x4800000 bytes
EAL: Virtual area found at 0x7f9d09000000 (size = 0x4800000)
EAL: Ask a virtual area of 0x800000 bytes
EAL: Virtual area found at 0x7f9d08600000 (size = 0x800000)
EAL: Ask a virtual area of 0xc00000 bytes
EAL: Virtual area found at 0x7f9d07800000 (size = 0xc00000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d07200000 (size = 0x400000)
EAL: Ask a virtual area of 0x1000000 bytes
EAL: Virtual area found at 0x7f9d06000000 (size = 0x1000000)
EAL: Ask a virtual area of 0x2000000 bytes
EAL: Virtual area found at 0x7f9d03e00000 (size = 0x2000000)
EAL: Ask a virtual area of 0x800000 bytes
EAL: Virtual area found at 0x7f9d03400000 (size = 0x800000)
EAL: Ask a virtual area of 0x400000 bytes
EAL: Virtual area found at 0x7f9d02e00000 (size = 0x400000)
EAL: Ask a virtual area of 0x200000 bytes
EAL: Virtual area found at 0x7f9d02a00000 (size = 0x200000)
EAL: Requesting 128 pages of size 2MB from socket 0
EAL: Requesting 128 pages of size 2MB from socket 1
EAL: TSC frequency is ~2300000 KHz
EAL: open shared lib //DPDK_RH65.218.0/librte_pmd_oce.so
PMD: librte_pmd_oce by 6WIND registered
EAL: Master core 0 is ready (tid=26419800)
EAL: Core 4 is ready (tid=fffec700)
EAL: Core 5 is ready (tid=ff5eb700)
EAL: Core 6 is ready (tid=febea700)
EAL: Core 7 is ready (tid=fe1e9700)
EAL: Core 3 is ready (tid=9ed700)
```

```
EAL: Core 2 is ready (tid=13ee700)
EAL: Core 1 is ready (tid=1def700)
EAL: PCI device 0000:01:00.0 on NUMA socket 0
EAL:   probe driver: 10df:720 librte_pmd_oce
EAL: PCI device 0000:01:00.1 on NUMA socket 0
EAL:   probe driver: 10df:720 librte_pmd_oce
Interactive-mode selected
Configuring Port 0 (socket 0)
Port 0: 00:90:FA:30:97:D6
Configuring Port 1 (socket 0)
Port 1: 00:90:FA:30:97:DA
Checking link statuses...
Port 0 Link Up - speed 10000 Mbps - full-duplex
Port 1 Link Up - speed 10000 Mbps - full-duplex
Done

testpmd> start tx_first

  io packet forwarding - CRC stripping disabled - packets/burst=32
  nb forwarding cores=2 - nb forwarding ports=2
  RX queues=1 - RX desc=128 - RX free threshold=0
  RX threshold registers: pthresh=8 hthresh=8 wthresh=0
  TX queues=1 - TX desc=512 - TX free threshold=0
  TX threshold registers: pthresh=32 hthresh=0 wthresh=0
  TX RS bit threshold=0 - TXQ flags=0x0

<<< NOTE: Wait for about 10-15 seconds >>>
testpmd> stop
Telling cores to stop...
Waiting for lcores to finish...

  ---------------------- Forward statistics for port 0  ----------------------
  RX-packets: 24013794       RX-dropped: 0             RX-total: 24013794
  TX-packets: 24004546       TX-dropped: 0             TX-total: 24004546
  ----------------------------------------------------------------------------

  ---------------------- Forward statistics for port 1  ----------------------
  RX-packets: 24004514       RX-dropped: 0             RX-total: 24004514
  TX-packets: 24013826       TX-dropped: 0             TX-total: 24013826
  ----------------------------------------------------------------------------

  ++++++++++++++ Accumulated forward statistics for all ports++++++++++++++
  RX-packets: 48018308       RX-dropped: 0             RX-total: 48018308
  TX-packets: 48018372       TX-dropped: 0             TX-total: 48018372
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Done.
testpmd> quit
Stopping port 0...done
Stopping port 1...done
bye...
```

3 Limitations

Please note the following limitations when installing and using the Emulex PMD.

3.1 Multi-channel Limitations

If you install the Emulex PMD on only one or two channels on the port, it must be installed on the third and/or fourth channel. This is necessary because freeing up the resources for the first or second channel does not release the required RSS resources.

3.2 SR-IOV Limitations

The Emulex PMD has the following limitations when using SR-IOV:

- With RSS enabled, up to six VFs are supported with one PF in a SR-IOV configuration. In this configuration, the PMD running on the VFs has access to two RSS rings each.

  RSS is enabled by default and the following command is typically used in the load_drivers script (see section 2.5.1, "Create the load_drivers Script," on page 12):

  ```
  # insmod $BINARIES/dpdk_surf.ko
  ```

- With RSS disabled, up to twelve VFS are supported with one PF. To disable RSS, use the following command in the load_drivers script (see section 2.5.1, "Create the load_drivers Script," on page 12):

  ```
  # insmod $BINARIES/dpdk_surf.ko max_rxq_per_port=1
  ```

- SR-IOV VFs are not supported if multichannel is enabled (that is, if UMC, Flex10, or NIC Partitioning (NPAR) are used).
- In a 4-port adapter, up to six VFs are supported on each of the first three ports. The fourth port does not support any VFs because of limited resources.
- Alternative requester ID interpretation (ARI) has not been fully qualified.

Note: SR-IOV configuration information is available in the *Emulex Drivers for Linux User Manual*.

3.3 FCoE and iSCSI Limitations

The Emulex PMD has the following limitations when using FCoE or iSCSI:

- Concurrent DPDK and FCoE traffic on the same port has not been fully tested and is not supported for 2-port and 4-port adapters.
- Concurrent DPDK and iSCSI traffic on the same port has not been fully tested and is not supported for 4-port adapters.

4 Tunings and Operational Considerations

4.1 Tunable Parameters via DPDK Applications

DPDK applications include multiple tunable parameters. For details, see the documentation at the DPDK project site:

```
http://www.dpdk.org/
```

4.2 Recommended RX/TX Queues

For the Emulex PMD, at least three pairs of RX/TX queues are needed for good performance.

4.3 Receive and Transmit Tunings

The DPDK API includes a set of receive/transmit (RX/TX) configuration thresholds used to tune the behavior of Emulex PMD receive and transmit functions.

4.3.1 Tuning the Receive Function of the Emulex PMD

The Emulex PMD manages the RX free threshold that is supplied in the RX queue configuration data structure at RX queue creation.

The RX Free Threshold parameter drives the notification of the replenished RX queue descriptors to the adapter by the receive function of the PMD that manages the adapter.

The receive function of the PMD only notifies accumulated RX queue entries once their total number is greater than or equal to the RX free threshold, and notifies a number of RX entries that is equal to this threshold when this situation occurs.

The value of the RX free threshold must be a multiple of the number of queue entries per CPU cache line (eight RX queue entries on Intel CPUs with a 64-byte cache line).

This method minimizes the number of expensive notification operations that perform a 32-bit write access to an adapter doorbell register.

4.3.2 Tuning the Transmit Function of the Emulex PMD

The Emulex PMD manages the following TX configuration parameters supplied in the DPDK API to tune the behavior of the transmit function:

- TX Completion Threshold – can be set with the --txrst=N parameter of the testpmd application.
- TX Free Threshold – can be set with the --txfreet=N parameter of the testpmd application.
- TX Write-Back Threshold – can be set with the --txwt=N parameter of the testpmd application.

4.3.2.1 Transmit Completion Threshold

The TX completion threshold drives the rate at which completed TX queue entries are notified by the adapter.

The transmit function of the Emulex PMD does not systematically set the Completion bit in the TX Queue Header entry (that precedes Queue data entrie(s)) of each output packet in the TX Queue. Instead, it only sets the Completion bit once the total number of TX Queue entries supplied to the adapter with their Completion bit unset reaches the value of the TX completion threshold.

The TX completion threshold must meet the following constraints:

- Be less than the number of TQ Queue entries
- Be a multiple of the number of TX Queue entries per CPU cache line (four TX queue entries on Intel CPUs with a 64-byte cache line).

This method minimizes the number of write-back memory accesses performed by the adapter in the TX Completion Queue, and the actual number of Completion Queue entries exchanged for a given output packet rate.

4.3.2.2 Transmit Free Threshold

The configuration of TX queues includes a TX Free Threshold parameter that is the number of used TX queue entries that must be reached before the transmit function of the PMD first checks for so-called “free” TX queue entries whose completion has been notified by the adapter.

The transmit function of the PMD only looks for valid entries in a TX Completion Queue associated with a TX queue once the number of used TX queue entries reaches the value of the TX free threshold.

The value of the TX Free threshold must satisfy the following constraints:

- Be less than the number of TX Queue entries.
- Be a multiple of the number of TX Completion Queue entries per CPU cache line.
- Be a multiple of the TX Completion Threshold times the number of TX Completion Queue entries per CPU cache line.

4.3.2.3 Transmit Write-Back Threshold

The TX write-back threshold drives the rate at which processed TX completion queues entries are notified to the adapter by the transmit function.

The transmit function of the PMD accumulates processed TX Completion Queue entries and only notifies them once their total number is greater than or equal to the TX Write-Back Threshold parameter. The notification includes the threshold value (minimum) when this situation occurs.

The value of the TX Write-Back Threshold parameter must be a multiple of the number of TX Completion Queue entries per CPU cache line (four 16-byte TX Completion entries on Intel CPUs with a 64-byte cache line).

This method minimizes the number of expensive notification operations that perform a 32-bit write access to an adapter doorbell register.

4.4 Disabling VLAN Tag Stripping

The OneConnect OCe14000-series adapters enable the VLAN tag stripping option by default. This option may cause some DPDK applications that expect VLAN tags to be included in received packets to not work properly, such as the 6WINDGate application.

This option is a port-wide setting and disabling it requires the PMD driver to issue a command by a PF on the port. If an application requires VLAN tag stripping to be disabled on a VF, then load the PMD on a PF on the port (which will issue the command).

Note: The Windows operating systems expects VLAN tags to be stripped, so disabling the VLAN tag stripping option on a port which is running Windows VMs will cause issues in the Windows VM.

4.5 Disabling LLDP

Some DPDK applications may require Link Layer Discovery Protocol (LLDP) to be disabled. LLDP can only be disabled by a PF, since it is a port-wide setting. The hbacmd utility can be invoked on the host with a PF to disable LLDP on the port if it is required by a virtual machine (VM).