Emulex® Poll Mode Driver for OneConnect® Adapters

User Guide

Version 11.2
December 30, 2016

DRVPMD-OCA-UG112

Corporate Headquarters
San Jose, CA

Website
www.broadcom.com

Table of Contents

Chapter 1: Overview

1.1 Purpose

Broadcom has partnered with 6WIND to provide an Emulex® poll mode driver (PMD) for the OCe14000-series 10Gb and 40Gb Ethernet network adapters and converged network adapters.

The Emulex PMD delivers the power of the data plane development kit (DPDK) to next generation network functions virtualization (NFV) workloads, and enables customers to choose the best networking connectivity for their server deployments.

The Emulex PMD enables 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 Internet Small Computer Systems Interface (iSCSI) support
- Universal Multichannel (UMC) configuration support
  - PMD installation is available on all four channels per port provided that the PMD is installed on three or four channels.
  - Four physical functions (PFs) exist per port.
- Single root input/output virtualization (SR-IOV) configuration support
  - On Emulex OCe14000B-series adapters, DPDK SR-IOV is not supported on the last port of the adapter.
    - With receive side scaling (RSS) enabled, up to nine virtual functions (VFs) are supported with one PF in an SR-IOV configuration. In this configuration, the PMD running on the VFs has access to two RSS rings each.
    - With RSS disabled, up to 14 VFs are supported with one PF in an SR-IOV configuration. For more information on specifying command line parameters on the `dpdk_surf.ko` module, see Section 2.3.4, max_rxq_per_port <int>.
    - In a 4-port adapter, up to nine VFs are supported on each of the first three ports. The fourth port does not support any VFs because of limited resources.
    - SR-IOV-enabled systems support up to 14 VFs on each port (does not apply to HP Flex-10 systems).
    - HP Flex-10 systems with SR-IOV-enabled support up to eight VFs on each port.

1.3 References

- *Emulex Drivers for Linux for OneConnect Adapters User Guide.*
  This manual includes SR-IOV configuration information and is available on the Broadcom website.
- Emulex PMD for DPDK program information. Refer to the Broadcom website.
- 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 multichannel
VF	virtual function
VLAN	virtual local area network
VM	virtual machine

Chapter 2: Configuring the Emulex PMD

2.1 Basic Setup

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

2.1.1 Download the Required Files

Download the following files:

- `dpdk-1.7.1.tar.gz`, `dpdk-1.8.0.tar.gz`, or `dpdk-2.0`

  You can download these files directly from the dpdk.org website, 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 **Other Versions** to see additional available files and formats from the browsing interface.
- Emulex Linux Ethernet (`be2net`) source driver files (`be2net-<build version>.src.rpm`)

  These files are available in your OEM package or at the Broadcom website.
- Emulex firmware (`OneConnect-Flash-11.x.xxx.x-x64.iso`)

2.1.2 Flash the Firmware

To flash the firmware, perform these steps:

1. Flash the Emulex firmware to your OCe14000-series adapter using the Emulex OneConnect® Flash Utility.
2. Configure the OCe14000-series adapter to use the "NIC-Only" profile by using the PXE Boot BIOS.
3. Install Red Hat Enterprise Linux (RHEL) 6.5 64-bit, RHEL 7.0, or RHEL 7.1 64-bit with the **Software Development Workstation** option.
4. 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
   ```
5. Reboot, log in as root, and verify that `be2net.ko` is not loaded:

   ```
   lsmod | grep be2net
   ```

2.1.3 Install Linux Ethernet (be2net) Source Driver Files

To install the Linux Ethernet (`be2net`) source driver files, perform this step:

On the system under test, with the appropriate file (`be2net-<build version>.src.rpm`) downloaded as described in Section 2.1, Basic Setup, 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 Section 2.2.2, Build the Emulex SURF and PMD Binaries, and the `cp` command line in Section 2.2.3, Copy the Binaries.

2.2 Build and Copy the Binaries

2.2.1 Build the DPDK Binaries

To build the DPDK binaries, perform this step:

On the system under test, with the `dpdk-<version>.tar.gz` file downloaded as described in Section 2.1.1, Download the Required Files, build the DPDK binaries. For example:

```
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
```

2.2.2 Build the Emulex SURF and PMD Binaries

To build the Emulex SURF and PMD Binaries, perform this step:

Use the `Palau_11.x.xxx.x_DPDK_PMD_Internal.zip` file downloaded as described in Section 2.1.1, Download the Required Files, to build the SURF and PMD binaries. For example:

```
cd /
mkdir 11.x.xxx.0
cd 11.x.xxx.0
cp Palau_11.x.xxx.x_DPDK_PMD_Internal.zip .
unzip Palau_11.x.xxx.x_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
```

2.2.3 Copy the Binaries

```
cd /
mkdir DPDK_RH65.218.0
cd /11.x.xxx.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
```

2.3 Parameters for the dpdk_surf.ko Module

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

2.3.1 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.

2.3.2 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.

2.3.3 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. The maximum value is 32.

NOTE This parameter must be increased to a number greater than 8 when the total number of `be2net` network interface card (NIC) functions in a system exceed 8. The total number of `be2net` NIC functions present in a system must not exceed 32 to use DPDK.

2.3.4 max_rxq_per_port <int>

The `max_rxq_per_port` parameter specifies the maximum number of receive (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 might fail in SR-IOV with more than nine VFs enabled per port. To support 10 to 14 VFs per port, you must request a single non-RSS RX queue by setting `max_rxq_per_port` to 1. This action forces the Emulex PMD to create only one non-RSS RX queue per OCe14000-series adapter port.

2.4 Creating the load_drivers and run_testpmd Scripts

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

2.4.1 Creating the load_drivers Script

To create the `load_drivers` script, perform these steps:

1. Gather the Linux Ethernet interface names for the 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 NPAR configurations add the following delay after loading the be2net
driver:
sleep 2

/* 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. */

/* Check for the file to set sriov_numvfs. */
# find /sys -name sriov_numvfs
/sys/devices/pci0000:00/0000:00:02.2/0000:04:00.0/sriov_numvfs
/sys/devices/pci0000:00/0000:00:02.2/0000:04:00.1/sriov_numvfs
```

```
/* Set the VF number for each PF interface. */
# echo <num_vfs> >
/sys/devices/pci0000:00/0000:00:02.2/0000:04:00.0/sriov_numvfs
# echo <num_vfs>
>/sys/devices/pci0000:00/0000:00:02.2/0000:04:00.1/sriov_numvfs

# 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
```

2.4.2 Creating the run_testpmd Script

To create the `run_testpmd` script, perform these steps:

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
```

2.5 Running a Test

To run a test with the scripts, perform these steps:

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
```

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...
```

Chapter 3: Limitations

The following limitations apply when installing and using the Emulex PMD.

3.1 Multichannel Limitations

If you install the Emulex PMD on only one or two channels on the port, it must be installed on the third channel, fourth channel, or both. This installation 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 nine VFs are supported with one PF in an 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:

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

  For additional information, see Section 2.4.1, Creating the load_drivers Script.
- With RSS disabled, up to 14 VFS are supported with one PF. To disable RSS, use the following command in the `load_drivers` script:

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

  For additional information, see Section 2.4.1, Creating the load_drivers Script.
- In a 4-port adapter, up to four 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. For SR-IOV configuration information, refer to the *Emulex Drivers for Linux for OneConnect Adapters User Guide*.

3.3 RHEL 7.2 with DPDK 2.0 Limitations

RHEL 7.2 does not compile with DPDK 2.0.

Chapter 4: Tuning and Operational Considerations

4.1 Tunable Parameters Using DPDK Applications

DPDK applications include multiple tunable parameters. For details about these parameters, refer to the documentation at the DPDK project site:

http://www.dpdk.org/

4.2 Recommended RX/TX Queues

For best performance, the Emulex PMD requires at least three pairs of receive/transmit (RX/TX) queues.

4.3 Receive and Transmit Tunings

The DPDK API includes a set of RX/TX configuration thresholds that tune the 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 notifies only accumulated RX queue entries after 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 central processing unit (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 entries) of each output packet in the TX Queue. Instead, it only sets the Completion bit when 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 requirements:

- 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).

These settings minimize 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 when 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 requirements:

- 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 notifies them only when 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

OCe14000-series adapters enable the VLAN tag stripping option by default. This option might cause some DPDK applications that expect virtual LAN (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, load the PMD on a PF on the port (which will issue the command).

NOTE The Windows operating system expects VLAN tags to be stripped, so disabling the VLAN tag stripping option on a port that is running Windows virtual machines (VMs) causes 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, because 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 VM.


BROADCOM