Mellanox OFED Driver for VMware vSphere 5.5 User Manual

Rev 2.3.3

www.mellanox.com

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

Release	Date	Description
2.3.3	September, 2015	Added Section 3.3.2, “VXLAN”, on page 35
2.3.2	April, 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 and Ethernet adapter cards. It is also intended for application developers.

Common Abbreviations and Acronyms

Table 2 - Abbreviations and Acronyms (Sheet 1 of 2)

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
iSER	iSCSI RDMA Protocol
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
IPoIB	IP over InfiniBand
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet

Table 2 - Abbreviations and Acronyms (Sheet 2 of 2)

Abbreviation / Acronym	Whole Word / Description
SDP	Sockets Direct Protocol
SL	Service Level
SRP	SCSI RDMA Protocol
MPI	Message Passing Interface
EoIB	Ethernet over Infiniband
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane
vHBA	Virtual SCSI Host Bus adapter
uDAPL	User Direct Access Programming Library

Glossary

The following is a list of concepts and terms related to InfiniBand in general and to Subnet Managers in particular. It is included here for ease of reference, but the main reference remains the *InfiniBand Architecture Specification*.

Table 3 - Glossary (Sheet 1 of 2)

Channel Adapter (CA), Host Channel Adapter (HCA)	An IB device that terminates an IB link and executes transport functions. This may be an HCA (Host CA) or a TCA (Target CA).
HCA Card	A network adapter card based on an InfiniBand channel adapter device.
IB Devices	Integrated circuit implementing InfiniBand compliant communication.
IB Cluster/Fabric/ Subnet	A set of IB devices connected by IB cables.
In-Band	A term assigned to administration activities traversing the IB connectivity only.
Local Identifier (ID)	An address assigned to a port (data sink or source point) by the Subnet Manager, unique within the subnet, used for directing packets within the subnet.
Local Device/Node/ System	The IB Host Channel Adapter (HCA) Card installed on the machine running IBDIAG tools.

Table 3 - Glossary (Sheet 2 of 2)

Local Port	The IB port of the HCA through which IBDIAG tools connect to the IB fabric.
Master Subnet Manager	The Subnet Manager that is authoritative, that has the reference configuration information for the subnet. See Subnet Manager.
Multicast Forwarding Tables	A table that exists in every switch providing the list of ports to forward received multicast packet. The table is organized by MLID.
Network Interface Card (NIC)	A network adapter card that plugs into the PCI Express slot and provides one or more ports to an Ethernet network.
Standby Subnet Manager	A Subnet Manager that is currently quiescent, and not in the role of a Master Subnet Manager, by agency of the master SM. See Subnet Manager.
Subnet Administrator (SA)	An application (normally part of the Subnet Manager) that implements the interface for querying and manipulating subnet management data.
Subnet Manager (SM)	One of several entities involved in the configuration and control of the an IB fabric.
Unicast Linear Forwarding Tables (LFT)	A table that exists in every switch providing the port through which packets should be sent to each LID.
Virtual Protocol Interconnect (VPI)	A Mellanox Technologies technology that allows Mellanox channel adapter devices (ConnectX®) to simultaneously connect to an InfiniBand subnet and a 10GigE/40GigE subnet (each subnet connects to one of the adapter ports)

Related Documentation

Table 4 - Reference Documents

Document Name	Description
InfiniBand Architecture Specification, Vol. 1, Release 1.2.1	The InfiniBand Architecture Specification that is provided by IBTA
IEEE Std 802.3ae™-2002 (Amendment to IEEE Std 802.3-2002) Document # PDF: SS94996	Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation
Firmware Release Notes for Mellanox adapter devices	See the Release Notes PDF file relevant to your adapter device. For further information please refer to the Mellanox website. www.mellanox.com -> Support -> Firmware Download
MFT User Manual	Mellanox Firmware Tools User's Manual. For further information please refer to the Mellanox website. www.mellanox.com -> Products -> InfiniBand/ VPI Driver -> Firmware Tools
MFT Release Notes	Release Notes for the Mellanox Firmware Tools. For further information please refer to the Mellanox website. www.mellanox.com -> Products -> InfiniBand/ VPI Driver -> Firmware Tools
VMware vSphere 5.1 Documentation Center	http://pubs.vmware.com/vsphere-51/index.jsp#com.vmware.vsphere.networking.doc/GUID-ACE7C3A0-7682-443C-B353-824945704042.html

1 Introduction

Mellanox OFED ESXi is a software stack based on the OpenFabrics (OFED) Linux stack adapted for VMware, and operates across all Mellanox network adapter solutions supporting up to 56Gb/s InfiniBand (IB) or up to 40Gb/s Ethernet (ETH) and 2.5 or 5.0 GT/s PCI Express 2.0 and 3.0 uplinks to servers.

All Mellanox network adapter cards are compatible with OpenFabrics-based RDMA protocols and software, and are supported with major operating system distributions.

The following sub-sections briefly describe the various components of the Mellanox OFED stack.

1.1 mlx4 Driver

`mlx4` is the low level driver implementation for the ConnectX® family adapters designed by Mellanox Technologies. ConnectX® family adapters can operate as an InfiniBand adapter, or as an Ethernet NIC. The OFED driver supports InfiniBand and Ethernet NIC configurations. To accommodate the supported configurations, the driver is split into the following modules:

mlx4_core

Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation.

mlx4_ib

Handles InfiniBand-specific functions and plugs into the InfiniBand midlayer

mlx4_en

A 10/40GigE driver under drivers/net/ethernet/mellanox/mlx4 that handles Ethernet specific functions and plugs into the netdev mid-layer

1.2 Mid-layer Core

Core services include: management interface (MAD), and Subnet Administrator (SA) interface. The stack includes components for both user-mode and kernel applications. The core services run in the kernel and expose an interface to user-mode for verbs and management.

1.3 ULPs

IPoIB

The IP over IB (IPoIB) driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand connected or datagram transport service. IPoIB pre-appends the IP datagrams with an encapsulation header, and sends the outcome over the InfiniBand transport service. The transport service is Unreliable Datagram (UD) by default, but it may also be configured to be Reliable Connected (RC). The interface supports unicast, multicast and broadcast.

1.4 InfiniBand Subnet Manager

All InfiniBand-compliant ULPs require a proper operation of a Subnet Manager (SM) running on the InfiniBand fabric, at all times. An SM can run on any node or on an IB switch. OpenSM is an InfiniBand-compliant Subnet Manager, and it is not part of the ESX OFED driver. It can be run either on a switch or on a VM configured as pass through.

1.5 Mellanox Firmware Tools

The Mellanox Firmware Tools (MFT) package is a set of firmware management tools for a single node. MFT can be used for:

- Generating a standard or customized Mellanox firmware image
- Burning a firmware image to a single node

MFT includes the following tools:

- flint: burns a firmware binary image or an expansion ROM image to the Flash device of a Mellanox network adapter/bridge/switch device. It includes query functions to the burnt firmware image and to the binary image file.
- Debug utilities: A set of debug utilities (e.g., itrace, fwtrace, mlxdump, mstdump, mlx-mcg, wqdump, mcra, i2c, mget_temp, and pckt_drop)

For additional details, please refer to the MFT User's Manual
www.mellanox.com -> Products -> InfiniBand/VPI Driver -> Firmware Tools.

1.6 Mellanox OFED ESXi Package

1.6.1 Software Components

MLNX_OFED_ESXi contains the following software components:

- Mellanox Host Channel Adapter Drivers
  - mlx4 which is split into multiple modules:
    - mlx-compat
    - mlx4_core (low-level helper)
    - mlx4_ib (IB)
    - mlx4_en (Ethernet)
- Mid-layer core
  - Verbs, MADs, SA
- Upper Layer Protocols (ULPs)
  - IPoIB,

1.7 Module Parameters

1.7.1 mlx4 Module Parameters

To set **mlx4** parameters:

```
esxcli system module parameters set -m mlx4_core -p <parameter>=<value>
```

and/or

```
esxcli system module parameters set -m mlx4_ib -p <parameter>=<value>
```

and/or

```
esxcli system module parameters set -m mlx4_en -p <parameter>=<value>
```

To show all parameters which were set until now:

```
esxcfg-module -g <module name>
```

Parameters which are not set by the user, remain on default value.

The following sections list the available `mlx4` parameters.

1.7.1.1 mlx4_ib Parameters

sm_guid_assign:	Enable SM alias_GUID assignment if sm_guid_assign > 0 (Default: 1) (int)
dev_assign_str[1]:	Map device function numbers to IB device numbers (e.g.'0000:04:00.0-0,002b:1c:0b.a-1,...'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for IB device numbers (e.g. 1). Max supported devices - 32 (string)

1. In the current version, this parameter is using decimal number to describe the InfiniBand device and not hexadecimal number as it was in previous versions in order to uniform the mapping of device function numbers to InfiniBand device numbers as defined for other module parameters (e.g. num_vfs and probe_vf).
   For example to map mlx4_15 to device function number 04:00.0 in the current version we use "`options mlx4_ib dev_assign_str=04:00.0-15`" as opposed to the previous version in which we used "`options mlx4_ib dev_assign_str=04:00.0-f`"

1.7.1.2 mlx4_core Parameters

set_4k_mtu:	(Obsolete) attempt to set 4K MTU to all ConnectX ports (int)
debug_level:	Enable debug tracing if > 0 (int)
msi_x:	0 - don't use MSI-X, 1 - use MSI-X, >1 - limit number of MSI-X irqs to msi_x (non-SRIOV only) (int)
enable_sys_tune:	Tune the cpu's for better performance (default 0) (int)
block_loopback:	Block multicast loopback packets if > 0 (default: 1) (int)
num_vfs:	Either a single value (e.g. '5') to define uniform num_vfs value for all devices functions or a string to map device function numbers to their num_vfs values (e.g. '0000:04:00.0-5,002b:1c:0b.a-15'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for num_vfs value (e.g. 15). (string)

probe_vf:	Either a single value (e.g. '3') to indicate that the Hypervisor driver itself should activate this number of VFs for each HCA on the host, or a string to map device function numbers to their probe_vf values (e.g. '0000:04:00.0-3,002b:1c:0b.a-13'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for probe_vf value (e.g. 13). (string)
log_num_mgm_entry_size:	log mgm size, that defines the num of qp per mcg, for example: 10 gives 248.range: 7 <= log_num_mgm_entry_size <= 12. To activate device managed flow steering when available, set to -1 (int)
fast_drop:	Enable fast packet drop when no recieve WQEs are posted (int)
enable_64b_cqe_eqe:	Enable 64 byte CQEs/EQEs when the FW supports this if non-zero (default: 1) (int)
log_num_mac:	Log2 max number of MACs per ETH port (1-7) (int)
log_num_vlan:	(Obsolete) Log2 max number of VLANs per ETH port (0-7) (int)
log_mtts_per_seg:	Log2 number of MTT entries per segment (0-7) (default: 0) (int)
port_type_array:	Specifies the protocol type of the ports. Valid port types: 1-ib, 2-eth
log_num_qp:	log maximum number of QPs per HCA (default: 18) (int)
log_num_srq:	log maximum number of SRQs per HCA (default: 16) (int)
log_rdmarc_per_qp:	log number of RDMARC buffers per QP (default: 4) (int)
log_num_cq:	log maximum number of CQs per HCA (default: 12) (int)
log_num_mcg:	log maximum number of multicast groups per HCA (default: 13) (int)
log_num_mpt:	log maximum number of memory protection table entries per HCA (default: 19) (int)
log_num_mtt:	log maximum number of memory translation table segments per HCA (default: max(20, 2*MTTs for register all of the host memory limited to 30)) (int)
internal_err_reset:	Reset device on internal errors if non-zero (default is 1) (int)
enable_vxlan_offloads	Enables VXLAN offloads when supported by NIC (default: 1)(int)

1.7.1.3 mlx4_en Parameters

inline_thold:	Threshold for using inline data (int) Default and max value is 104 bytes. Saves PCI read operation transaction, packet less then threshold size will be copied to hw buffer directly.
pfctx:	Priority based Flow Control policy on TX[7:0]. Per priority bit mask (uint)
pfcrx:	Priority based Flow Control policy on RX[7:0]. Per priority bit mask (uint)
enable_device_rss	Enables RSS mode (Receive Side Scaling). Device RSS mode means received data is distributed by the network adapter to up to 16 rx rings. It use up to 4/8/16 rx rings according to number of CPUs available on the host.

1.8 Device Capabilities

Normally, an application needs to query the device capabilities before attempting to create a resource. It is essential for the application to be able to operate over different devices with different capabilities.

Specifically, when creating a QP, the user needs to specify the maximum number of outstanding work requests that the QP supports. This value should not exceed the queried capabilities. However, even when you specify a number that does not exceed the queried capability, the verbs can still fail since some other factors such as the number of scatter/gather entries requested, or the size of the inline data required, affect the maximum possible work requests. Hence an application should try to decrease this size (halving is a good new value) and retry until it succeeds.

2 Installation

This chapter describes how to install and test the Mellanox OFED for ESX package on a single host machine with Mellanox InfiniBand and/or Ethernet adapter hardware installed.

2.1 Hardware and Software Requirements

Table 1 - Software and Hardware Requirements

Requirements	Description
Platforms	A server platform with an adapter card based on one of the following Mellanox Technologies' InfiniBand HCA devices: • MT27508 ConnectX®-3 (VPI, IB, EN) (firmware: fw-ConnectX3) • MT4103 ConnectX®-3 Pro (VPI, IB, EN) (firmware: fw-ConnectX3Pro)
Device ID	For the latest list of device IDs, please visit Mellanox website.
Operating System	ESXi 5.5 operating system.
Installer Privileges	The installation requires administrator privileges on the target machine.

2.2 Installing Mellanox OFED Driver for VMware vSphere

Please uninstall any previous Mellanox driver packages prior to installing the new version.

➢ ***To install the driver:***

1. Log into the ESXi server with root permissions.
2. Install the driver.

```
#> esxcli software vib install -d <path>/<bundle_file>
```

Example:

```
#> esxcli software vib install -d <path>/MLNX-OFED-ESX-2 3 1 0-10EM-550 0 0 1331820.zip
```

3. Reboot the machine.
4. Verify the driver was installed successfully.

```
# esxcli software vib list|grep Mellanox
net-ib-core                  2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
net-ib-ipoib                 2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
net-ib-mad                   2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
net-ib-sa                    2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
net-mlx-compat               2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
net-mlx4-core                2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
net-mlx4-en                  2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
net-mlx4-ib                  2.3.1.0-10EM.550.0.0.1331820         Mellanox  PartnerSupported  2015-03-02
```

After the installation process, all kernel modules are loaded automatically upon boot.

2.3 Removing Mellanox OFED Driver

Please unload the driver before removing it.

➢ ***To remove all the drivers:***

1. Log into the ESXi server with root permissions.
2. List the existing OFED driver modules. (see Step 4 in Section 2.2, on page 15)
3. Remove each module.

```
#> esxcli software vib remove -n net-ib-ipoib
#> esxcli software vib remove -n net-mlx4-ib
#> esxcli software vib remove -n net-ib-sa
#> esxcli software vib remove -n net-ib-mad
#> esxcli software vib remove -n net-ib-core
#> esxcli software vib remove -n net-mlx4-en
#> esxcli software vib remove -n net-mlx4-core
#> esxcli software vib remove -n net-mlx-compat
```

To remove the modules, the command must be run in the same order as shown in the example above.

4. Reboot the server.

2.4 Loading/Unloading Driver Kernel Modules

➢ ***To unload the driver:***

```
#> /opt/mellanox/bin/openibd.sh stop
```

➢ ***To load the driver:***

```
#> /opt/mellanox/bin/openibd.sh start
```

➢ ***To restart the driver:***

```
#> /opt/mellanox/bin/openibd.sh restart
```

2.5 Firmware Programming

1. Download the bootable binary image (md5sum: e7b3e9357ca4045fabe2e8a95d951343) from the Mellanox Firmware Tools (MFT) site.

2. Install the image according to the steps described in the README file.

The following procedure requires custom boot image downloading, mounting and booting from a USB device.

3 Features Overview and Configuration

3.1 Ethernet Network

3.1.1 Interface

3.1.1.1 Port Type Management

ConnectX®-3/ConnectX®-3 Pro ports can be configured to work as InfiniBand or Ethernet ports. The port type depends on card type. In case of a VPI card, the default type is IB. If you wish to change the port type, use the `mlxconfig` script.

For further information, please refer to section “mlxconfig - Changing Device Configuration Tool” in the MFT User Manual (www.mellanox.com > Products > Software > Firmware Tools).

To use a VPI card as an Ethernet only card, run:

```
/opt/mellanox/bin/mlxconfig -d /dev/mt4099_pciconf0 set LINK_TYPE_P1=2 LINK_TYPE_P2=2
```

The protocol types are:

- Port Type 1 = IB
- Port Type 2 = Ethernet

ESXi driver does not support setting one port as IB and one port as Ethernet. Both ports should be set to the same port type.

3.1.2 Ethtool

ethtool is a standard utility for controlling network drivers and hardware, particularly for wired Ethernet devices. It can be used to:

- Get identification and diagnostic information
- Get extended device statistics
- Control speed, duplex, autonegotiation and flow control for Ethernet devices
- Control checksum offload and other hardware offload features
- Control DMA ring sizes and interrupt moderation

The following are the ethtool supported options:

Table 2 - ethtool Supported Options

Options	Description

ethtool -i eth<x>	Checks driver and device information.
	For example:
	`#> ethtool -i eth2`
	`driver: mlx4_en (MT_0DD0120009_CX3)`
	`version: 2.1.6 (Aug 2013)`
	`firmware-version: 2.30.3000`
	`bus-info: 0000:1a:00.0`

Table 2 - ethtool Supported Options

Options	Description
ethtool -k eth<x>	Queries the stateless offload status. Stateless offload which are not supported will be notified as OFF and "Function not implemented".
ethtool -c eth<x>	Queries interrupt coalescing settings.
ethtool -C eth<x> adaptive-rx on|off	Enables/disables adaptive interrupt moderation. By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern.
ethtool -C eth<x> [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high N]	Sets the values for packet rate limits and for moderation time high and low values. • Above an upper limit of packet rate, adaptive moderation will set the moderation time to its highest value. • Below a lower limit of packet rate, the moderation time will be set to its lowest value.
ethtool -C eth<x> [rx-usecs N] [rx-frames N]	Sets the interrupt coalescing settings when the adaptive moderation is disabled. **Note:** usec settings correspond to the time to wait after the *last* packet is sent/received before triggering an interrupt.
ethtool -a eth<x>	Queries the pause frame settings.
ethtool -A eth<x> [rx on|off] [tx on|off]	Sets the pause frame settings.
ethtool -g eth<x>	Queries the ring size values.
ethtool -S eth<x>	Obtains additional device statistics.
ethtool -s eth<x> msglvl [N]	Changes the current driver message level.

3.1.3 Checksum Offload

MLNX_OFED supports the following Receive IP/L4 Checksum Offload modes:

- CHECKSUM_UNNECESSARY: By setting this mode the driver indicates to the Linux Networking Stack that the hardware successfully validated the IP and L4 checksum so the Linux Networking Stack does not need to deal with IP/L4 Checksum validation.

  Checksum Unnecessary is passed to the OS when all of the following are true:

  - Ethtool -k <DEV> shows rx-checksumming: on
  - Received TCP/UDP packet and both IP checksum and L4 protocol checksum are correct.
- CHECKSUM_COMPLETE: When the checksum validation cannot be done or fails, the driver still reports to the OS the calculated by hardware checksum value. This allows accelerating checksum validation in Linux Networking Stack, since it does not have to calculate the whole checksum including payload by itself.

  Checksum Complete is passed to OS when all of the following are true:

- Ethtool -k <DEV> shows rx-checksumming: on
- Using ConnectX®-3, firmware version 2.31.7000 and up
- Received IpV4/IpV6 non TCP/UDP packet

- CHECKSUM_NONE: By setting this mode the driver indicates to the Linux Networking Stack that the hardware failed to validate the IP or L4 checksum so the Linux Networking Stack must calculate and validate the IP/L4 Checksum.

  Checksum None is passed to OS for all other cases.

3.1.3.1 Counters

Counters are used to provide information about how well an operating system, an application, a service, or a driver is performing. The counter data helps determine system bottlenecks and fine-tune the system and application performance. The operating system, network, and devices provide counter data that an application can consume to provide users with a graphical view of how well the system is performing.

The counter index is a QP attribute given in the QP context. Multiple QPs may be associated with the same counter set, If multiple QPs share the same counter its value represents the cumulative total.

- ConnectX®-3 supports 127 different counters which are allocated as follow:
  - 4 counters reserved for PF - 2 counters for each port
  - 2 counters reserved for VF - 1 counter for each port
  - All other counters if exist are allocated by demand
- ESX can read Virtual Functions' port counters through sysfs located under:

```
# /proc/sysfs/class/net/vmnic*/vf*/statistics/
```

To display the network device Ethernet statistics, you can run:

```
Ethtool -S <devname>
```

Counter	Description
rx_packets	Total packets successfully received.
rx_bytes	Total bytes in successfully received packets.
rx_multicast_packets	Total multicast packets successfully received.
rx_broadcast_packets	Total broadcast packets successfully received.
rx_errors	Number of receive packets that contained errors preventing them from being deliverable to a higher-layer protocol.
rx_dropped	Number of receive packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.
rx_length_errors	Number of received frames that were dropped due to an error in frame length
rx_over_errors	Number of received frames that were dropped due to hardware port receive buffer overflow
rx_crc_errors	Number of received frames with a bad CRC that are not runts, jabbers, or alignment errors

Counter	Description
rx_jabbers	Number of received frames with a length greater than MTU octets and a bad CRC
rx_in_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1500:46] (42 is also counted for VLAN-tagged frames)
rx_out_range_length_error	Number of received frames with a length/type field value in the (decimal) range [1535:1501]
rx_lt_64_bytes_packets	Number of received 64-or-less-octet frames
rx_127_bytes_packets	Number of received 65-to-127-octet frames
rx_255_bytes_packets	Number of received 128-to-255-octet frames
rx_511_bytes_packets	Number of received 256-to-511-octet frames
rx_1023_bytes_packets	Number of received 512-to-1023-octet frames
rx_1518_bytes_packets	Number of received 1024-to-1518-octet frames
rx_1522_bytes_packets	Number of received 1519-to-1522-octet frames
rx_1548_bytes_packets	Number of received 1523-to-1548-octet frames
rx_gt_1548_bytes_packets	Number of received 1549-or-greater-octet frames
tx_packets	Total packets successfully transmitted.
tx_bytes	Total bytes in successfully transmitted packets.
tx_multicast_packets	Total multicast packets successfully transmitted.
tx_broadcast_packets	Total broadcast packets successfully transmitted.
tx_errors	Number of frames that failed to transmit
tx_dropped	Number of transmitted frames that were dropped
tx_lt_64_bytes_packets	Number of transmitted 64-or-less-octet frames
tx_127_bytes_packets	Number of transmitted 65-to-127-octet frames
tx_255_bytes_packets	Number of transmitted 128-to-255-octet frames
tx_511_bytes_packets	Number of transmitted 256-to-511-octet frames
tx_1023_bytes_packets	Number of transmitted 512-to-1023-octet frames
tx_1518_bytes_packets	Number of transmitted 1024-to-1518-octet frames
tx_1522_bytes_packets	Number of transmitted 1519-to-1522-octet frames
tx_1548_bytes_packets	Number of transmitted 1523-to-1548-octet frames
tx_gt_1548_bytes_packets	Number of transmitted 1549-or-greater-octet frames
rx_prio_<i>_packets	Total packets successfully received with priority i.
rx_prio_<i>_bytes	Total bytes in successfully received packets with priority i.
rx_novlan_packets	Total packets successfully received with no VLAN priority.
rx_novlan_bytes	Total bytes in successfully received packets with no VLAN priority.
tx_prio_<i>_packets	Total packets successfully transmitted with priority i.
tx_prio_<i>_bytes	Total bytes in successfully transmitted packets with priority i.

Counter	Description
tx_novlan_packets	Total packets successfully transmitted with no VLAN priority.
tx_novlan_bytes	Total bytes in successfully transmitted packets with no VLAN priority.
rx_pause_prio_<i>	The total number of PAUSE frames received from the far-end port
rx_pause_duration_prio_<i>	The total time in microseconds that far-end port was requested to pause transmission of packets.
rx_pause_transition_prio_<i>	The number of receiver transitions from XON state (paused) to XOFF state (non-paused)
tx_pause_prio_<i>	The total number of PAUSE frames sent to the far-end port
tx_pause_duration_prio_<i>	The total time in microseconds that transmission of packets has been paused
tx_pause_transition_prio_<i>	The number of transmitter transitions from XON state (paused) to XOFF state (non-paused)
vport<i>_rx_unicast_packets	Unicast packets received successfully
vport<i>_rx_unicast_bytes	Unicast packet bytes received successfully
vport<i>_rx_multicast_packets	Multicast packets received successfully
vport<i>_rx_multicast_bytes	Multicast packet bytes received successfully
vport<i>_rx_broadcast_packets	Broadcast packets received successfully
vport<i>_rx_broadcast_bytes	Broadcast packet bytes received successfully
vport<i>_rx_dropped	Received packets discarded due to luck of software receive buffers (WQEs). Important indication to weather RX completion routines are keeping up with hardware ingress packet rate
vport<i>_rx_errors	Received packets discarded due to receive error condition
vport_rx_filtered	Received packets dropped due to packet check that failed. For example: Incorrect VLAN, incorrect Ethertype, unavailable queue/QP or loopback prevention
vport<i>_tx_unicast_packets	Unicast packets sent successfully
vport<i>_tx_unicast_bytes	Unicast packet bytes sent successfully
vport<i>_tx_multicast_packets	Multicast packets sent successfully
vport<i>_tx_multicast_bytes	Multicast packet bytes sent successfully
vport<i>_tx_broadcast_packets	Broadcast packets sent successfully
vport<i>_tx_broadcast_bytes	Broadcast packet bytes sent successfully
vport<i>_tx_errors	Packets dropped due to transmit errors
rx_lro_aggregated	Number of packets processed by the LRO mechanism
rx_lro_flushed	Number of offloaded packets the LRO mechanism passed to kernel
rx_lro_no_desc	LRO mechanism has no room to receive packets from the adapter. In normal condition, it should not increase

Counter	Description
rx_alloc_failed	Number of times failed preparing receive descriptor
rx_csum_good	Number of packets received with good checksum
rx_csum_none	Number of packets received with no checksum indication
tx_chksum_offload	Number of packets transmitted with checksum offload
tx_queue_stopped	Number of times transmit queue suspended
tx_wake_queue	Number of times transmit queue resumed
tx_timeout	Number of times transmitter timeout
tx_tso_packets	Number of packet that were aggregated
rx<i>_packets	Total packets successfully received on ring i
rx<i>_bytes	Total bytes in successfully received packets on ring i.
tx<i>_packets	Total packets successfully transmitted on ring i.
tx<i>_bytes	Total bytes in successfully transmitted packets on ring i.

3.1.4 RDMA over Converged Ethernet (RoCE)

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® EN with RoCE uses the proven and efficient RDMA transport to provide the platform for deploying RDMA technology in mainstream data center application at 10GigE and 40GigE link-speed. ConnectX® EN 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.

RoCE encapsulates IB transport in the following Ethernet packet

- RoCEv1 - dedicated ether type (0x8915)

RoCE on ConnectX® EN cards can be enabled in MLNX OFED ESXi only on VMs which are associated with SR-IOV EN Virtual Functions.

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 and therefore APM is not supported

- The GID table for each port is populated with N+1 entries where N is the number of IP addresses that are assigned to all network devices associated with the port including VLAN devices, alias devices and bonding masters. The only exception to this rule is a bonding master of a slave in a DOWN state. In that case, a matching GID to the IP address of the master will not be present in the GID table of the slave's port
- The first entry in the GID table (at index 0) for each port is always present and equal to the link local IPv6 address of the net device that is associated with the port. Note that even if the link local IPv6 address is not set, index 0 is still populated.
- GID format can be of 2 types, IPv4 and IPv6. IPv4 GID is a IPv4-mapped IPv6 address[1] while IPv6 GID is the IPv6 address itself
- 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 off-loaded by the hardware and does not go through Ethernet network driver. RoCE traffic is counted in the same place where InfiniBand traffic is counted; /sys/class/infiniband/<device>/ports/<port number>/counters/

3.1.4.1 Prerequisites

The following are the driver's prerequisites in order to set or configure RoCE:

- ConnectX®-3 firmware version 2.33.5100
- ConnectX®-3 Pro firmware version 2.33.5100
- For OEM adapters, the required firmware version is 2.33.5050
- All InfiniBand verbs applications which run over InfiniBand verbs should work on RoCE links if they use GRH headers.
- All ports must be set to use Ethernet protocol

3.1.4.2 Running and Configuring RoCE on ESXi VMs

RoCE on ESXi VMs can run only on VMs which are associated with SR-IOV EN Virtual Functions.

In order to function reliably, RoCE requires a form of flow control. While it is possible to use global flow control, this is normally undesirable, for performance reasons.

The normal and optimal way to use RoCE is to use Priority Flow Control (PFC). To use PFC, it must be enabled on all endpoints and switches in the flow path.

On ESXi, the PFC settings should be set on the ESXi host only and not on the VMs as the ESXi host is the one to control PFC settings. PFC settings can be changed using the mlx4_en parameters `pfctx` and `pfcrx`. For further information, please refer to Section 1.7.1.3, "mlx4_en Parameters", on page 14.

For further information on how to use and run RoCE on the VM, please refer t the VM's driver User Manual. Additional information can be found at the *RoCE Over L2 Network Enabled with PFC* User Guide:

http://www.mellanox.com/related-docs/prod_software/RoCE_with_Priority_Flow_Control_Application_Guide.pdf

1. For the IPv4 address A.B.C.D the corresponding IPv4-mapped IPv6 address is ::ffff.A.B.C.D

3.1.4.2.1 Configuring SwitchX® Based Switch System

➢ ***To enable RoCE, the SwitchX should be configured as follows:***

- Ports facing the host should be configured as access ports, and either use global pause or Port Control Protocol (PCP) for priority flow control
- Ports facing the network should be configured as trunk ports, and use Port Control Protocol (PCP) for priority flow control

For further information on how to configure SwitchX, please refer to MLNX-OS User Manual.

3.2 InfiniBand Network

3.2.1 Port Type Management

Please see Section 3.1.1.1, “Port Type Management”, on page 18.

3.2.2 Interface Configuration

VMware ESXi Server settings can be configured using the vSphere Client. Once the InfiniBand OFED driver is installed and configured, the administrator can make use of InfiniBand software available on the VMware ESXi Server machine. The InfiniBand package provides networking and storage over InfiniBand. The following sub-sections describe their configuration.

This section includes instructions for configuring various module parameters.

From ESXi use the following command to view all the available module parameters and default settings.

```
#> esxcli system module parameters list -m <module name>
```

When using ESXi, use vMA or remote CLI vicfg-module.pl to configure the module parameters in a similar way to what is done in the Service Console (COS) for ESXi.

3.2.2.1 Subnet Manager

The driver package requires InfiniBand Subnet Manager (SM) to run on the subnet. The driver package does not include an SM.

If your fabric includes a managed switch/gateway, please refer to the vendor's user's guide to activate the built-in SM.

If your fabric does not include a managed switch/gateway, an SM application should be installed on at least one non-ESXi Server machine in the subnet. You can download an InfiniBand SM such as OpenSM from www.openfabrics.org under the Downloads section.

3.2.2.2 Networking

The InfiniBand package includes a networking module called IPoIB, which causes each InfiniBand port on the VMware ESXi Server machine to be exposed as one or more physical network adapters, also referred to as uplinks or vmnics. To verify that all supported InfiniBand ports on the host are recognized and up, perform the following steps:

1. Connect to the VMware ESXi Server machine using the interface of VMware vSphere Client.
2. Select the "Configuration" tab.
3. Click the "Network Adapters" entry which appears in the "Hardware" list.

   A "Network Adapters" list is displayed, describing per uplink the "Device" it resides on, the port "Speed", the port "Configured" state, and the "vSwitch" name it connects to.

To create and configure virtual network adapters connected to InfiniBand uplinks, follow the instructions in the ESXi Server Configuration Guide document.

All features supported by Ethernet adapter uplinks are also supported by InfiniBand port uplinks (e.g., VMware® VMotionTM, NIC teaming, and High Availability), and their setting is performed transparently.

3.2.2.3 Virtual Local Area Network (VLAN) Support

To support VLAN for VMware ESXi Server users, one of the elements on the virtual or physical network must tag the Ethernet frames with an 802.1Q tag. There are three different configuration modes to tag and untag the frames as virtual machine frames:

1. Virtual Machine Guest Tagging (VGT Mode).
2. ESXi Server Virtual Switch Tagging (VST Mode).
3. External Switch Tagging (EST Mode).

EST is supported for Ethernet switches and can be used beyond IB/Eth Gateways transparently to VMware ESXi Servers within the InfiniBand subnet.

To configure VLAN for InfiniBand networking, the following entities may need to be configured according to the mode you intend to use:

- Subnet Manager Configuration

  Ethernet VLANs are implemented on InfiniBand using Partition Keys (See RFC 4392 for information). Thus, the InfiniBand network must be configured first. This can be done by configuring the Subnet Manager (SM) on your subnet. Note that this configuration is needed for both VLAN configuration modes, VGT and VST.

  For further information on the InfiniBand Partition Keys configuration for IPoIB, see the Subnet Manager manual installed in your subnet.

  The maximum number of Partition Keys available on Mellanox HCAs is:

  - 128 for ConnectX® IB family
  - Check with IB switch documentation for the number of supported partition keys

- Guest Operating System Configuration

  For VGT mode, VLANs need to be configured in the installed guest operating system. This procedure may vary for different operating systems. See your guest operating system manual on VLAN configuration.

  In addition, for each new interface created within the virtual machine, at least one packet should be transmitted. For example:

  Create a new interface (e.g., <eth1>) with IP address <ip1>.

➢ ***To guarantee that a packet is transmitted from the new interface:***

```
arping -I <eth1> <ip1> -c 1
```

- Virtual Switch Configuration

  For VST mode, the virtual switch using an InfiniBand uplink needs to be configured. For further information, see the *ESXi Server 3 Configuration Guide* and *ESXi Server 3 802.1Q VLAN Solutions* documents.

3.2.2.4 Maximum Transmit Unit (MTU) Configuration

On VMware ESXi Server machines, the MTU is set to 1500 bytes by default. IPoIB supports larger values and allows Jumbo Frames (JF) traffic up to 4052 bytes on VI3 and 4092 bytes on vSphere 5. The maximum value of JF supported by the InfiniBand device is:

- 2044 bytes for the InfiniHost III family
- 4052 / 4092 bytes for ConnectX® IB family (vSphere 5)

Running a datagram IPoIB MTU of 4092 requires that the InfiniBand MTU is set to 4k.

It is the administrator's responsibility to make sure that all the machines in the network support and work with the same MTU value. For operating systems other than VMware ESXi Server, the default value is set to 2044 bytes.

The procedure for changing the MTU may vary, depending on the OS. For example, to change it to 1500 bytes:

- On Linux - if the IPoIB interface is named ib0:

```
ifconfig ib0 mtu 1500
```

- On Microsoft® Windows - execute the following steps:
  a. Open "Network Connections"
  b. Select the IPoIB adapter and right click on it
  c. Select "Properties"
  d. Press "Configure" and then go to the "Advanced" tab
  e. Select the payload MTU size and change it to 1500
  f. Make sure that the firmware of the HCAs and the switches supports the MTU you wish to set.
  g. Configure your Subnet Manager (SM) to set the MTU value in the configuration file. The SM configuration for MTU value is per Partition Key (PKey).

     For example, to enable 4K MTUs on a default PKey using the OpenSM SM6, log into the Linux machine (running OpenSM) and perform the following commands:
  h. Edit the file:

```
/usr/local/ofed/etc/opensm/partitions.conf
```

     and include the line:

```
key0=0x7fff,ipoib,mtu=5 : ALL=full;
```

  i. Restart OpenSM:

```
/etc/init.d/opensmd restart
```

To enable 4k mtu support: run **"esxcli system module parameters set -m=mlx4_core -p=mtu_4k=1"**.
Changes will take effect after the reboot.

3.3 Virtualization

3.3.1 Single Root IO Virtualization (SR-IOV)

Single Root IO Virtualization (SR-IOV) is a technology that allows a physical PCIe device to present itself multiple times through the PCIe bus. This technology enables multiple virtual instances of the device with separate resources. Mellanox adapters are capable of exposing in ConnectX®-3 adapter cards up to 126 virtual instances called Virtual Functions (VFs). These virtual functions can then be provisioned separately. Each VF can be seen as an addition device connected to the Physical Function. It shares the same resources with the Physical Function.

SR-IOV is commonly used in conjunction with an SR-IOV enabled hypervisor to provide virtual machines direct hardware access to network resources hence increasing its performance.

In this chapter we will demonstrate setup and configuration of SR-IOV in a ESXi environment using Mellanox ConnectX® adapter cards family.

Please note, the maximum supported VFs in SR-IOV EN is 32.
The number of adapters on a server with SR-IOV EN can be calculated as follow:
(4 + min(number_of_cores, 16)*2 + number_of_vfs*3)*number_of_mellanox_adapters <= 128[1]

1. Each ESXi host has a total of 256 interrupt vectors. When the host boots, devices on the host such as storage controllers, physical network adapters, and USB controllers consume a subset of the 256 vectors. If these devices require more than 128 vectors, the maximum number of potentially supported VFs is reduced.

3.3.1.1 System Requirements

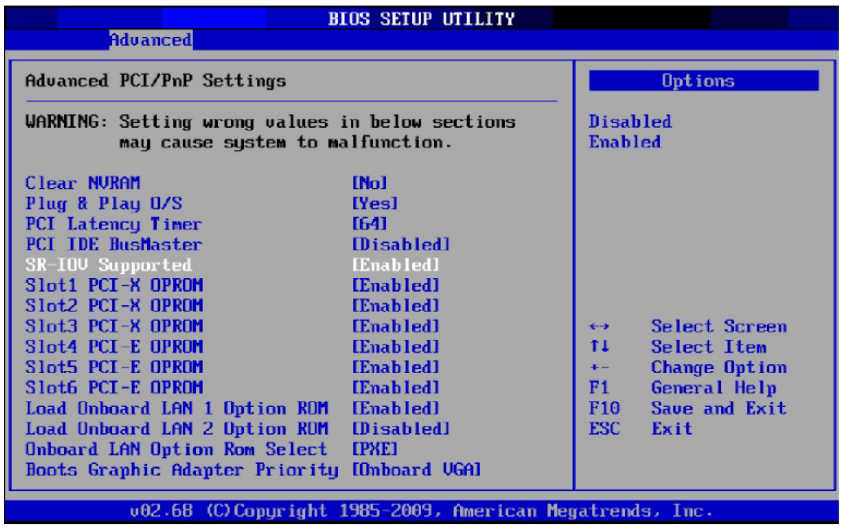
To set up an SR-IOV environment, the following is required:

- MLNX_OFED ESXi Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Mellanox ConnectX® Adapter Card family with SR-IOV capability

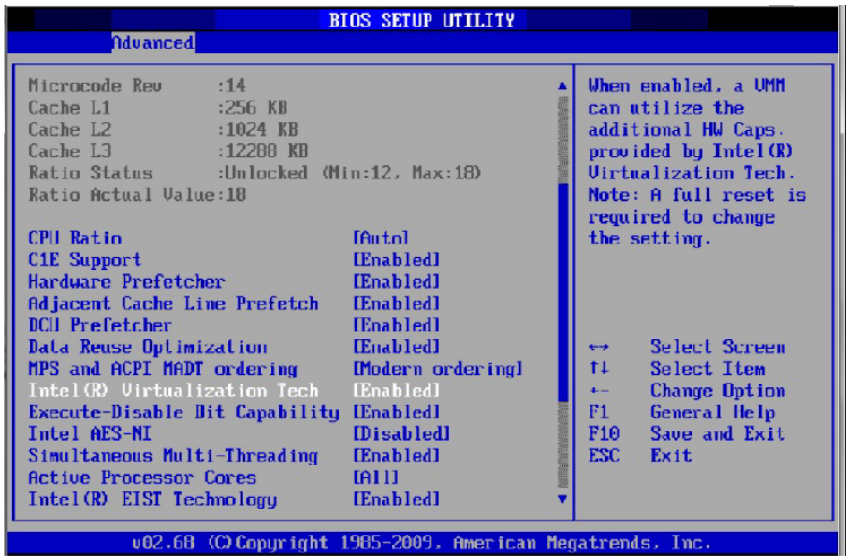
3.3.1.2 Setting Up SR-IOV

Depending on your system, perform the steps below to set up your BIOS. The figures used in this section are for illustration purposes only. For further information, please refer to the appropriate BIOS User Manual:

Step 1. Enable "SR-IOV" in the system BIOS.



Step 2. Enable "Intel Virtualization Technology".



Step 3. Install ESXi 5.5 that supports SR-IOV.

Step 4. Install the MLNX_OFED driver for ESXi.

SR-IOV can be enabled and managed by running the mlxconfig tool and setting the `SRIOV_EN` parameter to `"1"` and the maximum number of VFs required without re-burning the firmware

```
/opt/mellanox/bin/mlxconfig -d /dev/mt4099_pciconf0 set SRIOV_EN=1 NUM_OF_VFS=16
```

If you have a VPI card and you would like to use it as an Ethernet only card, you may also run the following command:

```
/opt/mellanox/bin/mlxconfig -d /dev/mt4099_pciconf0 set LINK_TYPE_P1=2 LINK_TYPE_P2=2
```

For further information, please refer to section *"mlxconfig - Changing Device Configuration Tool"* in the MFT User Manual (www.mellanox.com > Products > Software > Firmware Tools).

Step 5. Set the driver module parameters for Virtual Function instances as follow:

- For SR-IOV Ethernet:

```
esxcli system module parameters set -m mlx4_core -p 'num_vfs=<VFs over Port1, VFs
over port2, 0> port_type_array=2'
```

- For SR-IOV InfiniBand:

```
esxcli system module parameters set -m mlx4_core -p 'num_vfs=<Number of VFs> port_-
type_array=1'
```

Parameter	Recommended Value

num_vfs	• If absent, or zero: no VFs will be available
	• If its value is a single number in the range of 0-63: The driver will enable the `num_vfs` VFs on the HCA and this will be applied to all ConnectX® HCAs on the host.
	Note: In case of dual port card, single number value should be used only on IB ports, for SRIOV-EN use triplet.
	• If its a triplet x,y,z (applies only if all ports are configured as Ethernet) the driver creates:
	  • x single port VFs on physical port 1
	  • y single port VFs on physical port 2 (applies only if such a port exist)
	  • z n-port VFs (where n is the number of physical ports on device).
	This applies to all ConnectX® HCAs on the host.
	Note: It is highly recommended to set z to 0 (no dual port VF), since while using dual port VF on ESXi driver, the second port will not support PT_OPS. To have 2 ports Virtual Machine over virtual functions it is recommended to connect 2 single port VFs to a dual port VM.
	• If its format is a string: The string specifies the `num_vfs` parameter separately per installed HCA.
	The string format is: "bb:dd.f-v,bb:dd.f-v,…"
	  • bb:dd.f = bus:device.function of the PF of the HCA
	  • v = number of VFs to enable for that HCA which is either a single value or a triplet, as described above.

Parameter	Recommended Value

num_vfs	For example:
	• `num_vfs=5` - The driver will enable 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host
	• `num_vfs=00:04.0-5,00:07.0-8` - The driver will enable 5 VFs on the HCA positioned in BDF 00:04.0 and 8 on the one in 00:07.0)
	• `num_vfs=1,2,3` - The driver will enable 1 VF on physical port 1, 2 VFs on physical port 2 and 3 dual port VFs (applies only to dual port HCA when all ports are Ethernet ports).
	• `num_vfs=00:04.0-5;6;7,00:07.0-8;9;10` - The driver will enable:
	  • HCA positioned in BDF 00:04.0
	  • 5 single VFs on port 1
	  • 6 single VFs on port 2
	  • 7 dual port VFs
	  • HCA positioned in BDF 00:07.0
	  • 8 single VFs on port 1
	  • 9 single VFs on port 2
	  • 10 dual port VFs
	Applies when all ports are configure as Ethernet in dual port HCAs
	Notes:
	• PFs not included in the above list will not have SR-IOV enabled.
	• Triplets and single port VFs are only valid when all ports are configured as Ethernet. When an InfiniBand port exists, only `num_vfs=a` syntax is valid where `"a"` is a single value that represents the number of VFs.
	• The second parameter in a triplet is valid only when there are more than 1 physical port.
	In a triplet, `x+z<=16` and `y+z<=16`, the maximum number of VFs on each physical port must be 16.

port_type_array	Specifies the protocol type of the ports.

The example above loads the driver with 5 VFs (num_vfs). However, the number of VFs varies upon the working mode requirements.

The protocol types are:

- Port Type 1 = IB
- Port Type 2 = Ethernet
  - port_type_array=2 (Ethernet)
  - port_type_array=1 (IB)

Port type depends on card type. In case of a VPI card, the default is IB.

ESXi driver does not support setting one port as IB and one port as Ethernet. Both ports should be set to the same port type.

Step 6. Reboot the server.

If the SR-IOV is not supported by the server, the machine might not come out of boot/load.

Step 7. Load the driver and verify the SR-IOV is supported. Run:

```
~ # lspci |grep Mellanox
0000:06:00.0 Network controller: Mellanox Technologies MT27520 Family [vmnic3]
0000:06:00.1 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_0]
0000:06:00.2 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_1]
0000:06:00.3 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_2]
0000:06:00.4 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_3]
0000:06:00.5 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_4]
0000:06:00.6 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_5]
0000:06:00.7 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_6]
0000:06:01.0 Network controller: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Func-
tion] [PF_0.6.0_VF_7]
```

Where:

- “06:00" represents the Physical Function
- “06:00.**X**" represents the Virtual Function connected to the Physical Function

3.3.1.3 Assigning a Virtual Function to a Virtual Machine in the vSphere Web Client

After you enable the Virtual Functions on the host, each of them becomes available as a PCI device.

➢ ***To assign Virtual Function to a Virtual Machine in the vSphere Web Client***

Step 1. Locate the virtual machine in the vSphere Web Client.

Step a. To locate a virtual machine, select a datacenter, folder, cluster, resource pool, or host and click the Related Objects tab.

Step b. Click Virtual Machines and select the virtual machine from the list.

Step 2. Power off the virtual machine.

Step 3. Click the Manage tab of the virtual machine, and select Settings > VM Hardware.

Step 4. Click Edit.

Step 5. Expand the Memory section, and set the Limit to Unlimited.

Step 6. From the New device drop-down menu select PCI Device and click Add.

Step 7. From the New PCI device drop-down menu select the virtual function and click OK.

Step 8. Power on the virtual machine.

Adding a virtual function as a PCI device to a virtual machine sets memory reservation to the memory size of the virtual machine.

3.3.1.4 Configuring the Passthrough Device for a Virtual Function in the vSphere Web Client

After you configure a virtual machine with a virtual function as a PCI device, you can configure the virtual function with a static MAC address and a default VLAN with the help of the vSphere Web Client.

In the virtual machine configuration .vmx file, you can assign a static MAC address and a default VLAN to the virtual function.

➢ ***To configure the Passthrough Device for a Virtual Function in the vSphere Web Client***

Step 1. Locate the virtual machine in the vSphere Web Client.

Step a. To locate a virtual machine, select a datacenter, folder, cluster, resource pool, or host and click the Related Objects tab.

Step b. Click Virtual Machines and select the virtual machine from the list.

Step 2. Power off the virtual machine.

Step 3. Click the Manage tab of the virtual machine and select Settings.

Step 4. Click the VM Options tab and expand Advanced.

Step 5. Click Edit Configuration.

Step 6. To assign a static MAC address, add or edit the following parameters.

Parameter	Value
pciPassthruX.MACAddressType	static
pciPassthruX.MACAddress	MAC_address_of_the_virtual_function

X next to `pciPassthru` stands for the sequence number of the PCI device in the virtual machine. For example, 0 in pciPassthru0 represents the settings of the PCI device added first to the virtual machine.

Step 7. To assign a default VLAN, add or edit the pciPassthruX.defaultVlan parameter according to the following value guidelines. X next to pciPassthru stands for the sequence number of the PCI device in the virtual machine.

Option	Description
1-4094	Allow tagged only and do NOT allow guest VLAN tagging.
4095	Allow untagged only and allow guest VLAN tagging.

Step 8. Click OK.

Step 9. Power on the virtual machine.

3.3.1.5 VLAN Guest Tagging (VGT) and VLAN Switch Tagging (VST)

When running ETH ports on VGT, the ports may be configured to simply pass through packets as is from VFs (VLAN Guest Tagging), or the administrator may configure the Hypervisor to silently force packets to be associated with a VLAN/Qos (VLAN Switch Tagging).

In the latter case, untagged or priority-tagged outgoing packets from the guest will have the VLAN tag inserted, and incoming packets will have the VLAN tag removed. Any vlan-tagged packets sent by the VF are silently dropped.

3.3.2 VXLAN

3.3.2.1 Prerequisites

- HCA: ConnectX-3 Pro
- Firmware 2.35.5100 or higher
- DMFS enabled

3.3.2.2 Enabling VXLAN

To enable the VXLAN offloads support load the `mlx4_core` driver with Device-Managed Flow-steering (DMFS) enabled. DMFS is the default steering mode.

Make sure that the VXLAN offload module parameter is enabled:

```
enable_vxlan_offloads=1
```

For further information, please refer to section Section 1.7, “Module Parameters”, on page 12.

3.3.2.3 Important Notes

- VXLAN tunneling adds 50 bytes (14-eth + 20-ip + 8-udp + 8-vxlan) to the VM Ethernet frame. Please verify that either the MTU of the NIC who sends the packets, e.g. the VM virtio-net NIC or the host side veth device or the uplink takes into account the tunneling overhead. Meaning, the MTU of the sending NIC has to be decremented by 50 bytes (e.g 1450 instead of 1500), or the uplink NIC MTU has to be incremented by 50 bytes (e.g 1550 instead of 1500)

4 Troubleshooting

You may be able to easily resolve the issues described in this section. If a problem persists and you are unable to resolve it yourself please contact your Mellanox representative or Mellanox Support at support@mellanox.com.

4.1 General Related Issues

Table 3 - General Related Issues

Issue	Cause	Solution
The system panics when it is booted with a failed adapter installed.	Malfunction hardware component	1. Remove the failed adapter. 2. Reboot the system.
Mellanox adapter is not identified as a PCI device.	PCI slot or adapter PCI connector dysfunctionality	1. Run `lspci`. 2. Reseat the adapter in its PCI slot or insert the adapter to a different PCI slot. If the PCI slot confirmed to be functional, the adapter should be replaced.
Mellanox adapters are not installed in the system.	Misidentification of the Mellanox adapter installed	Run the command below to identify the Mellanox adapter installed. `lspci | grep Mellanox'`

4.2 Ethernet Related Issues

Table 4 - Ethernet Related Issues

Issue	Cause	Solution
No link.	Mis-configuration of the switch port or using a cable not supporting link rate.	• Ensure the switch port is not down • Ensure the switch port rate is configured to the same rate as the adapter's port
Degraded performance is measured when having a mixed rate environment (10GbE, 40GbE and 56GbE).	Sending traffic from a node with a higher rate to a node with lower rate.	Enable Flow Control on both switch's ports and nodes: • On the server side run: `ethtool -A <interface> rx on tx on` • On the switch side run the following command on the relevant interface: `send on force` and `receive on force`

Table 4 - Ethernet Related Issues

Issue	Cause	Solution
No link with break-out cable.	Misuse of the break-out cable or misconfiguration of the switch's split ports	• Use supported ports on the switch with proper configuration. For further information, please refer to the MLNX_OS User Manual. • Make sure the QSFP break-out cable side is connected to the SwitchX.

4.3 InfiniBand Related Issues

Table 5 - InfiniBand Related Issues

Issue	Cause	Solution
The following messages is logged after loading the driver: `multicast join failed with status -22`	Trying to join a multicast group that does not exist or exceeding the number of multicast groups supported by the SM.	If this message is logged often, check for the multicast group's join requirements as the node might not meet them. Note: If this message is logged after driver load, it may safely be ignored.
Logical link fails to come up while port logical state is **Initializing.**	The logical port state is in the Initializing state while pending the SM for the LID assignment.	1. Verify an SM is running in the fabric. Run '`sminfo`' from any connected Virtual Machine. 2. If SM is not running, activate the SM on a Virtual Machine or on managed switch.

4.4 InfiniBand/Ethernet Related Issues

Table 6 - InfiniBand/Ethernet Related Issues

Issue	Cause	Solution
Physical link fails to negotiate to maximum supported rate.	The adapter is running an outdated firmware.	Install the latest firmware on the adapter.
Physical link fails to come up while port physical state is **Polling**.	The cable is not connected to the port or the port on the other end of the cable is disabled.	Ensure that the cable is connected on both ends or use a known working cable

Table 6 - InfiniBand/Ethernet Related Issues

Issue	Cause	Solution
Physical link fails to come up while port physical state is **Disabled**.	The port was manually disabled.	Restart the driver: `/etc/init.d/openibd restart`

4.5 Installation Related Issues

Table 7 - Installation Related Issues

Issue	Cause	Solution
Driver installation fails.	The install script may fail for the following reasons: • Failed to uninstall the previous installation due to dependencies being used • The operating system is not supported	• Uninstall the previous driver before installing the new one • Use a supported operating system and kernel

4.6 SR-IOV Related Issues

Table 8 - SR-IOV Related Issues

Issue	Cause	Solution
Failed to enable SR-IOV. The following message is reported in dmesg: `mlx4_core 0000:xx:xx.0: Failed to enable SR-IOV, continuing without SR-IOV (err = -22)`	The number of VFs configured in the driver is higher than configured in the firmware.	1. Check the firmware SR-IOV configuration, run the mlxconfig tool (/usr/mellanox/bin/mlxconfig). 2. Set the number of VFs for the driver lower than the number of VFs set by firmware.
Failed to enable SR-IOV. The following message is reported in dmesg: `mlx4_core 0000:xx:xx.0: Failed to enable SR-IOV, continuing without SR-IOV (err = -12)`	SR-IOV is disabled in the BIOS.	Check that the SR-IOV is enabled in the BIOS (see Section 3.3.1.2, “Setting Up SR-IOV”, on page 30).

Table 8 - SR-IOV Related Issues

Issue	Cause	Solution
Unable to stop the driver or unload the module with the following on screen message: `"Unable to unload module mlx4_core :Busy"`	Active VFs are used by the active Virtual Machines	1. Unload the active Virtual Machines which use the VFs. 2. Stop the driver, run `/usr/mellanox/bin/openibd stop`.