ESXi Driver User Manual

Table of Contents

1 Overview 5

2 Installation 6

2.1 Hardware and Software Requirements 6

2.2 Identifying the Supported Driver and Firmware 6

2.3 Installing Firmware 6

2.4 Installing NATIVE ESXi Driver for VMware vSphere 6

2.5 Removing the Previous Driver Version 7

2.6 Installing nmlxcli Management Tool 7

2.7 Configuration 7

3 Features Overview and Configuration 8

3.1 Module Parameters 8

3.1.1 Setting and Viewing Driver Parameter Values 8

3.1.1.1 Set Parameters 8

3.1.1.2 Set Array Parameters 8

3.1.1.3 View Currently Configured Parameters 8

3.1.1.4 List Available Parameters and Descriptions 8

3.1.2 Parameters 9

3.1.2.1 nmlx5_core 9

3.1.2.2 nmlx5_rdma 10

3.2 nmxcli Management Tools 10

3.2.1 Overview 10

3.2.2 Executing nmlxcli commands 11

3.2.2.1 esxcli command reference 11

3.2.2.1.1 esxcli mellanox uplink 11

3.2.2.1.2 esxcli mellanox uplink link 11

3.2.2.1.3 esxcli mellanox uplink coalesce 11

3.2.2.1.4 esxcli mellanox uplink stats 11

3.2.2.1.5 esxcli mellanox uplink flows stat 12

3.2.2.1.6 esxcli mellanox uplink ecn 12

3.2.2.1.7 esxcli mellanox uplink sniffer 13

3.2.2.1.8 esxcli mellanox uplink fw 13

3.2.2.1.9 esxcli mellanox uplink rss 13

3.2.2.1.10 esxcli mellanox uplink dvx 13

3.2.2.1.11 esxcli mellanox uplink vfs 13

3.2.2.1.12 esxcli mellanox uplink conntrack 14

3.2.2.1.13 esxcli mellanox nvconfig 14

3.2.3 Configuration Manager 14
3.2.3.1 Applying a Profile with nmlxcli 14
3.2.3.1.1 Options 14
3.2.3.2 Applying a Profile with mlxconfig 15
3.2.3.3 Order of Operations 15
3.2.3.3.1 CX Profile 15
3.2.3.3.2 BF_AS_CX Profile 15
3.2.3.3.3 DPU Profile 16
3.2.3.4 Individual configuration values 16
3.2.3.4.1 CX Profile Configurations 16
3.2.3.4.2 BF_AS_CX Profile Configurations 17
3.2.3.4.3 DPU Profile Configurations 18
3.3 NVIDIA nmlx5 ESXi Driver Counters 20
3.3.1 Introduction 20
3.3.1.1 Counter Types 20
3.3.1.2 Interfaces 20
3.3.1.3 Other Markers 20
3.3.1.4 Naming Conventions 21
3.3.1.5 Ring / Channel Counters 21
3.3.1.5.1 Ring RX Counters (RX[i]_*) 21
3.3.1.5.2 Ring TX Counters (TX[i]_*) 22
3.3.1.6 Software Port Counters 23
3.3.1.7 vPort Counters 24
3.3.1.8 Physical Port Counters 25
3.3.1.8.1 IEEE 802.3 Counters 25
3.3.1.8.2 RFC 2863 Counters 27
3.3.1.8.3 RFC 2819 Counters 27
3.3.1.8.4 PHY Statistical Counters 28
3.3.1.8.5 Ethernet Extended Counters 28
3.3.1.9 Priority Port Counters 29
3.3.1.9.1 PFC Counters 29
3.3.1.9.2 PFC Stall Counters 30
3.3.1.10 PCIe / Device Counters 30
3.3.1.10.1 PCIe Performance Counters 30
3.3.1.10.2 PCIe Timers-and-States (TAS) Counters 31
3.3.1.11 Queue Counters 31
3.3.1.12 vNIC Environment Counters 31
3.3.1.13 Temperature Sensor Counters 32
3.3.1.14 VMware-Specific Counters 33
3.3.1.15 Standard Uplink Stats (vmk_UplinkStats) 33
3.3.1.16 Flow Steering Counters 34

3.3.1.17 RDMA Counters 35
3.3.1.17.1 RDMA Traffic Counters 35
3.3.1.17.2 RDMA Extended Counters 35
3.3.1.17.3 RDMA Software Counters 36
3.4 Hardware Large Receive Offload (HW LRO) 37
3.4.1 Overview 37
3.4.1.1 HW LRO and ENS Software LRO 37
3.4.2 Requirements 37
3.4.3 How It Works 38
3.4.4 When Segments Are Not Merged 38
3.4.5 Limitations 39
3.4.6 Configuration 39
3.4.6.1 lro_timeout (module parameter of nmlx5_core) 39
3.4.6.2 data_path_control (module parameter of nmlx5_core) 40
3.4.7 Verifying HW LRO 40
3.4.7.1 Driver log 40
3.4.7.2 Statistics 41
3.4.8 Troubleshooting 41
4 Document Revision History 43

1 Overview

NVIDIA® native ESXi drivers deliver high-performance, efficient networking for VMware vSphere environments by leveraging advanced hardware offloads available on NVIDIA® networking adapters. These drivers help provide near bare-metal performance in virtualized environments by reducing CPU overhead and accelerating data movement through capabilities such as RDMA over Converged Ethernet (RoCE).

With NVIDIA® ConnectX®-4 and later adapters, ESXi 6.5 and later supports a broad range of Ethernet speeds, including 10, 25, 40, 50, 100, and 200GbE. This enables high-throughput connectivity for demanding workloads such as virtualization, storage, cloud infrastructure, AI, and high-performance computing.

By combining native ESXi driver support with NVIDIA hardware acceleration, VMware vSphere deployments can achieve improved network efficiency, lower latency, and higher bandwidth utilization while maintaining compatibility with standard ESXi networking and management workflows.

2 Installation

This chapter describes how to install and test the NATIVE ESXi Driver for VMware vSphere and the NVIDIA nmlxcli management tool on a single host machine with NVIDIA Ethernet adapter hardware installed.

2.1 Hardware and Software Requirements

Requirements	Description
Platforms	A server platform with an adapter card based on one of the following HCA devices: • ConnectX-4 Lx onwards • BlueField-2/BlueField-3
Operating System	ESXi 6.7 onwards
Installer Privileges	The installation requires administrator privileges on the target machine.

2.2 Identifying the Supported Driver and Firmware

To find the supported driver / firmware versions, see Broadcom | VMware | Hardware Compatibility Guide .

2.3 Installing Firmware

Once the firmware version has been identified, it can be downloaded here Firmware Downloads. The page includes installation instructions.

2.4 Installing NATIVE ESXi Driver for VMware vSphere

> ⚠ Uninstall any previous driver packages prior to installing the new version. See "Removing the Previous Driver" for further information.

For further information on how to install the driver for VMware vSphere, see Patching ESXi host using Command Line.

To install the driver, perform the following steps:

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

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

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

2.5 Removing the Previous Driver Version

> ⚠ Unload the driver before removing it.

To remove all the drivers, perform the following steps:

1. Log into the ESXi server with root permissions.
2. List all the existing NATIVE ESXi driver modules.
3. Remove each module.

```
#> esxcli software vib remove -n nmlx5-core
```

2.6 Installing nmlxcli Management Tool

To install nmlxcli, perform the following steps:

1. Download the tool from ConnectX® Ethernet Driver for VMware® ESXi Server.
2. Log in to the ESXi server with root permissions.
3. Install the driver.

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

4. Reboot the machine
5. Verify that the driver was installed successfully.

2.7 Configuration

Once everything is installed, refer to these resources to configure the setup:

1. Applying FW configurations with nmlxcli, see Configuration Manager.
2. Configuring UPT/DVX(EDPIO) on host, see Enhanced Network Offloads and Advanced NIC Passthrough.
3. Configuring driver module parameters, see Module Parameters.

3 Features Overview and Configuration

The chapter contains the following sections:

- Module Parameters
- nmxcli Management Tools
    - Configuration Manager
- NVIDIA nmlx5 ESXi Driver Counters
- Hardware Large Receive Offload (HW LRO)

3.1 Module Parameters

3.1.1 Setting and Viewing Driver Parameter Values

A comprehensive guide is available at Configuring advanced driver module parameters in ESX/ESXi.

3.1.1.1 Set Parameters

Parameter settings are persistent across reboots.

```
esxcfg-module -s "param1=value1 param2=value2" nmlx5_core
esxcfg-module -s "param1=value1" nmlx5_rdma
```

3.1.1.2 Set Array Parameters

```
esxcfg-module -s "max_vfs=8,8" nmlx5_core
```

3.1.1.3 View Currently Configured Parameters

```
esxcfg-module -g nmlx5_core
esxcfg-module -g nmlx5_rdma
```

3.1.1.4 List Available Parameters and Descriptions

```
esxcli system module parameters list -m nmlx5_core
esxcli system module parameters list -m nmlx5_rdma
```

3.1.2 Parameters

3.1.2.1 nmlx5_core

Parameter	Description	Type	Default	Range	Notes
`max_vfs`	Number of PCI VFs to initialize per PF/port	uint[]	`0`	`0` – `128` per entry (device may limit); max 16 entries; `0` = disabled	Array is one value per port/ PF
`max_dvx_vfs`	Max PCI VFs to initialize as DVX devices	uint[]	`0`	`0` – `128` per entry; max 16 entries	Must align with `max_vfs`
`vf_mempool_extend`	Add X MB to the VF mempool size	int	`0`	≥ `0`	Ignored if `max_vfs=0`; use only if IB/RoCE VFs run out of memory;
`enable_nmlx_debug`	Enable debug prints	int	`0`	`0` = disabled, `1` = enabled	Deprecated; driver forces `0` and warns if set to `1`
`data_path_control`	Datapath control options bitmask	ulong	`0`	`b0` (`1`): disable striding RQ	
`mst_recovery`	Enable recovery mode (only NMST module loaded)	int	`0`	`0` = disabled, `1` = enabled	
`supported_num_ports`	Total number of ports to reserve memory for	uint	`1`	`0` – `16` ; `0` = preserve no memory	
`trust_state`	Port policy for switch priority and packet color from incoming packets	int	`1`	`1` = TRUST_PCP, `2` = TRUST_DSCP	
`trust_dscp_same_priority`	Enable TRUST_DSCP and map all priorities to the same class	uint	`0`	`0` = disabled, `1` = enabled	
`upt_provider_mode`	UPT datapath provider mode	uint	`0`	`0` = PREFER_DPA_FALLBACK_TO_ACE, `1` = FORCE_DPA, `2` = FORCE_ACE	

3.1.2.2 nmlx5_rdma

Parameter	Description	Type	Default	Range	Notes
`enable_nmlx_debug`	Enable debug prints	int	`0`	`0` = disabled, `1` = enabled	Deprecated; driver forces `0` and warns if set to `1`
`pcp_force`	PCP value to force on outgoing RoCE traffic	int	`-1`	`-1` = disabled, `0` – `7` = PCP value	Cannot be active when `dscp_to_pcp=1`
`dscp_force`	DSCP value to force on outgoing RoCE traffic	int	`-1`	`-1` = disabled, `0` – `63` = DSCP value	
`dscp_to_pcp`	Map the three MSBs of DSCP to PCP on outgoing RoCE traffic	int	`0`	`0` = disabled, `1` = enabled	Cannot be enabled when `pcp_force` is active (`!=-1`)
`roce_version`	RoCE versions enable option	int	`3`	`1` = RoCE v1, `2` = RoCE v2, `3` = RoCE v1 + v2	Not in inbox builds (VMKAPI ≥ 2.10)

3.2 nmxcli Management Tools

3.2.1 Overview

`nmlxcli` is the command-line management and diagnostics utility for NVIDIA native ESXi NIC drivers (`nmlx5`). It talks directly to the loaded kernel driver and lets the user:

- Discover Mellanox uplinks and read their general, link, and status information
- Read private (driver-level) statistics: rings, RX queues, flows, FW pages, HW tracer
- Inspect RSS engines, DVX VFs, VPORTs, and connection-tracking objects
- Configure ECN / RoCE congestion control
- Enable/disable the port sniffer (including RDMA traffic)
- Collect debug dumps: steering tables, FW resource dumps, CR dumps, FW tracer logs
- Apply non-volatile device configuration (nvconfig) profiles
- Set the E-Switch admin state for all SR-IOV VFs

The tool is installed at `/opt/mellanox/bin/nmlxcli` and is also registered as an esxcli plugin under the `mellanox` namespace (not all `nmlxcli` operations are available on `esxcli`).

Requirements: an ESXi host with the `nmlx5` native driver loaded, and root privileges. If the driver isn't loaded or its version doesn't match the tool, `nmlxcli` reports a driver-compatibility error (and logs it to syslog).

3.2.2 Executing nmlxcli commands

```
esxcli mellanox <namespace/command>...
```

This is the supported, user-facing interface. The output is formatted, there's built-in help (`--help` on any namespace), and input validation.

3.2.2.1 esxcli command reference

3.2.2.1.1 esxcli mellanox uplink

Command	Purpose	Required options
`list`	List all Mellanox uplinks with PCI, MAC, driver/FW version, WoL support	—
`info`	General information about one uplink	`-u <vmnicX>`
`state`	Uplink state (enabled/disabled/promisc/...)	`-u`
`timeouts`	Uplink timeout configuration, in milliseconds	`-u`

3.2.2.1.2 esxcli mellanox uplink link

Command	Purpose
`info`	Link status, duplex, speed, port type, auto-negotiation, transceiver, advertised link modes

3.2.2.1.3 esxcli mellanox uplink coalesce

Command	Purpose
`get`	Rx/Tx usecs and frames, adaptive Rx/Tx state, sample interval
`adaptive get`	Low/high packet-rate thresholds and their Rx/Tx usecs and frames

3.2.2.1.4 esxcli mellanox uplink stats

Command	Purpose
`all`	All private ring statistics

Command	Purpose
`activeRx`	Active RX queue-oriented statistics

3.2.2.1.5 `esxcli mellanox uplink flows stat`

Command	Purpose
`query`	Flow counters: created / destroyed / in-use. `-u` optional (all uplinks if omitted)
`clean`	Reset the flow counters. Note: results may become momentarily inconsistent

3.2.2.1.6 `esxcli mellanox uplink ecn`

ECN (congestion control for RoCE) is managed per protocol. Two protocols exist:

- `rRoceRp` — Reaction Point (the sender that reduces its rate on congestion notification)
- `rRoceNp` — Notification Point (the receiver that generates CNPs)

Each has the same verbs:

Command	Purpose
`enable` / `disable`	Turn the protocol on or off
`query`	Show whether the protocol is enabled
`get`	Show all tuning parameters
`set`	Change one or more tuning parameters

Plus:

Command	Purpose
`statistics get`	ECN counters
`statistics clear`	Reset ECN counters

`rRoceRp set` parameters (all optional, all take integers) control the rate-reduction/recovery algorithm: `--clamp-tgt-rate`, `--clamp-tgt-rate-after`, `--rpg-time-reset`, `--rpg-byte-reset`, `--rpg-threshold`, `--rpg-max-rate`, `--rpg-ai-rate`, `--rpg-hai-rate`, `--rpg-gd`, `--rpg-min-dec-fac`, `--rpg-min-rate`, `--rate-first-cnp`, `--dce-tcp-g`, `--dce-tcp-rtt`, `--rate-reduce-period`, `--init-alpha`.

`rRoceNp set` parameters control CNP generation: `--min-time-cnp`, `--cnp-dscp` (0–63), `--cnp-prio-mode` (0 = use configured 802.1p priority, 1 = copy from received packet), `--cnp-802-p-prio` (0–7).

Run `esxcli mellanox uplink ecn rRoceRp set --help` for the per-parameter descriptions and valid ranges.

3.2.2.1.7 esxcli mellanox uplink sniffer

Command	Purpose
`enable`	Enable the sniffer, including RDMA traffic
`disable`	Disable the sniffer
`query`	Show current sniffer state

3.2.2.1.8 esxcli mellanox uplink fw

Command	Purpose
`tracer enable` / `disable`	Turn FW tracing on/off for an uplink
`tracer state`	Tracer state plus counters (printed/excluded logs, events, errors, lost events, buffer overruns). `-u` optional
`resource menu`	List the available FW resource-dump segments. `-i <index>` narrows to one entry
`resource dump`	Dump a FW resource segment to a file

`resource dump` requires `-u`, `-s <segment-type>`, and `-f <file-path>`. Optional narrowing: `-i/-j` (indices), `-o/-p` (object counts), `-v` (VHCA ID).

3.2.2.1.9 esxcli mellanox uplink rss

Command	Purpose
`list`	All RSS engines on the uplink
`priQ`	One RSS engine, selected by primary OS/VMK queue ID (`-q`)

3.2.2.1.10 esxcli mellanox uplink dvx

Command	Purpose
`list`	All DVX VFs on the uplink

3.2.2.1.11 esxcli mellanox uplink vfs

Command	Purpose
`setAdminState`	Set the E-Switch admin state for **all** VF vports (`-z 0` = UP, `-z 1` = AUTO). The PF is left unchanged

3.2.2.1.12 `esxcli mellanox uplink conntrack`

Command	Purpose
`dumpVmk`	Connection-tracking objects in VMkernel format
`dumpHw`	Connection-tracking objects in hardware format

Both dump everything by default; `-i <hw-id>` dumps a single object.

3.2.2.1.13 `esxcli mellanox nvconfig`

Command	Purpose
`profile`	Apply a non-volatile configuration profile to a device

Requires `-x <sbdf>` (PCI address, function must be 0) and `-w <profile>`:

- `0` — DEFAULT: reset all configuration to factory defaults
- `1` — DPU: configure both host and embedded CPU on BlueField devices
- `2` — BF_AS_CX: configure a BlueField device to act as a plain NIC
- `3` — CX: configure a ConnectX device

Some profiles can only configure the currently-enabled PFs; in that case the tool tells you to **re-apply the profile after a reboot** to cover the remaining PFs.

3.2.3 Configuration Manager

The configuration manager is used to apply device configurations in bulk using predefined configuration profiles. These profiles help ensure correct and optimal device operation.

3.2.3.1 Applying a Profile with nmlxcli

Use the following command:

```
<nmlxcli path> --sbdf <sbdf> --nv-config-profile <profile number>
```

Or, using the shortform:

```
<nmlxcli path> -x <sbdf> -w <profile number>
```

3.2.3.1.1 Options

3.2.3.1.1.1 `-w` , `--nv-config-profile`

Specifies the configuration profile to apply:

- 0 – DEFAULT — device default configuration

- 1 – DPU — for BlueField-3 devices operating in DPU mode
- 2 – BF_AS_CX — for BlueField-3 devices operating in NIC mode
- 3 – CX — for ConnectX-6 Dx, ConnectX-6 Lx, ConnectX-7, and ConnectX-8 devices

3.2.3.1.1.2 `-x` , `--sbdf`

Specifies the SBDF/PCI address corresponding to PF0.

Example:

```
0000:4b:00.0
```

> ⚠ Do not use the address of another PF (e.g. 0000:4b:00.1)

3.2.3.2 Applying a Profile with mlxconfig

> ⚠ **mlxconfig** requires MFT tools (not bundled with **nmlxcli** or the driver). For a complete configuration, **nmlxcli** provides a simpler solution with the configuration manager.

When applying profiles with **mlxconfig** , the order of operations remains the same as with **nmlxcli** .

To reset the device configuration to its default values, run:

```
<mlxconfig path> -d <sbdf or mst_device> reset
```

To apply a complete configuration profile, each required configuration must be set explicitly:

```
<mlxconfig path> -d <sbdf or mst_device> set <CONFIG1>=<VALUE1> <CONFIG2>=<VALUE2> ...
```

The exact configurations for each profile are specified below.

3.2.3.3 Order of Operations

3.2.3.3.1 CX Profile

1. Reset configurations to their default values.
2. Set the CX profile configurations.
3. Power cycle the system.

3.2.3.3.2 BF_AS_CX Profile

1. Configure the device to operate in NIC mode.
2. Reset configurations to their default values.
3. Set the **BF_AS_CX** profile configurations.
4. Power cycle the system.

3.2.3.3.3 DPU Profile

1. Configure the device to operate in DPU mode.
2. Reset configurations to their default values.
3. Set the DPU profile configurations on the host side.
4. Set the DPU profile configurations on the DPU side.
5. Power cycle the system.

Notes:

- Resetting configurations can be done by applying the **DEFAULT** profile with **nmlxcli**, or by running the reset command with **mlxconfig**.
- With **nmlxcli**, the required profile values can be applied using a single command.
- With **mlxconfig**, each configuration must be set explicitly, and per-PF configurations must be applied on every relevant PF.

3.2.3.4 Individual configuration values

These settings are applied automatically by the Config Manager and do not require manual application.

This section is relevant for reference, or if the user is using MFT tools to configure their setup instead of using the Config Manager.

3.2.3.4.1 CX Profile Configurations

3.2.3.4.1.1 Global: Apply Once

Configuration	Value	Notes
PF_BAR2_ENABLE	0	
PF_BAR2_SIZE	0	
PER_PF_NUM_SF	1	
SRIOV_EN	1	
PF_NUM_OF_VF_VALID	1	
NUM_PF_MSIX	127	
LAG_RESOURCE_ALLOCATION	1	Pre-Alloc
VF_MIGRATION_MODE	2	Enable
UPT_EMULATION_ENABLE	1	ConnectX-7 and ConnectX-8 only

3.2.3.4.1.2 Physical Port: Apply Once

Configuration	Value	Notes
LLDP_NB_RX_MODE_P1 / LLDP_NB_RX_MODE_P2	2	All
LLDP_NB_TX_MODE_P1 / LLDP_NB_TX_MODE_P2	2	All
LINK_TYPE_P1 / LINK_TYPE_P2	2	Ethernet

3.2.3.4.1.3 Per PF: Apply on All PFs

Configuration	Value	Notes
PF_SF_BAR_SIZE	10	
PF_TOTAL_SF	1	ConnectX-6 Dx, ConnectX-6 Lx
PF_TOTAL_SF	65	ConnectX-7, ConnectX-8
PF_NUM_OF_VF	8	ConnectX-6 Dx, ConnectX-6 Lx
PF_NUM_OF_VF	64	ConnectX-7, ConnectX-8

3.2.3.4.2 BF_AS_CX Profile Configurations

3.2.3.4.2.1 Global: Apply Once

Configuration	Value	Notes
PF_BAR2_ENABLE	0	
PF_BAR2_SIZE	0	
PER_PF_NUM_SF	1	
SRIOV_EN	1	
PF_NUM_OF_VF_VALID	1	
NUM_PF_MSIX	127	
LAG_RESOURCE_ALLOCATION	1	Pre-Alloc
VF_MIGRATION_MODE	2	Enable
UPT_EMULATION_ENABLE	1	
INTERNAL_CPU_OFFLOAD_ENGINE	1	Disabled
INTERNAL_CPU_RSHIM	1	Disabled

3.2.3.4.2.2 Physical Port: Apply Once

Configuration	Value	Notes
LLDP_NB_RX_MODE_P1 / LLDP_NB_RX_MODE_P2	2	All
LLDP_NB_TX_MODE_P1 / LLDP_NB_TX_MODE_P2	2	All
LINK_TYPE_P1 / LINK_TYPE_P2	2	Ethernet

3.2.3.4.2.3 Per PF: Apply on All PFs

Configuration	Value
PF_SF_BAR_SIZE	10
PF_TOTAL_SF	65
PF_NUM_OF_VF	64

3.2.3.4.3 DPU Profile Configurations

3.2.3.4.3.1 Global: Apply Once

Configuration	Value	Notes
PF_BAR2_ENABLE	0	
PF_BAR2_SIZE	0	
PER_PF_NUM_SF	1	
SRIOV_EN	1	
PF_NUM_OF_VF_VALID	1	
NUM_PF_MSIX	127	
LAG_RESOURCE_ALLOCATION	1	Pre-Alloc
VF_MIGRATION_MODE	2	Enable
UPT_EMULATION_ENABLE	1	
INTERNAL_CPU_OFFLOAD_ENGINE	0	Enabled
DPA_AUTHENTICATION	1	
NUM_OF_PF	N+1	N = default number of PFs

3.2.3.4.3.2 Physical Port: Apply Once

Configuration	Value	Notes
`LLDP_NB_RX_MODE_P1 / LLDP_NB_RX_MODE_P2`	2	All
`LLDP_NB_TX_MODE_P1 / LLDP_NB_TX_MODE_P2`	2	All
`LINK_TYPE_P1 / LINK_TYPE_P2`	2	Ethernet

3.2.3.4.3.3 Host-Side PF Configurations

Apply the following on all host-side PFs, except the communication channel PF, which is `PF #N`.

Configuration	Value
`PF_TOTAL_SF`	65
`PF_SF_BAR_SIZE`	10
`PF_NUM_OF_VF`	64
`PF_DEVICE_ID_ENABLE`	0
`SILENT_MODE`	0
`PORT_OWNER`	1

3.2.3.4.3.4 DPU-Side ECPF Configurations

Apply the following on all DPU-side ECPFs, except the communication channel ECPF, which is `ECPF #N`.

Configuration	Value
`PF_TOTAL_SF`	1
`PF_SF_BAR_SIZE`	10
`PF_NUM_OF_VF`	0
`PF_DEVICE_ID_ENABLE`	0
`SILENT_MODE`	0
`PORT_OWNER`	1

3.2.3.4.3.5 Communication Channel Configuration

Apply the following on `PF #N` on the host and `ECPF #N` on the DPU.

Configuration	Value
PF_TOTAL_SF	0
PF_SF_BAR_SIZE	0
PF_NUM_OF_VF	0
PF_DEVICE_ID_ENABLE	1
PF_DEVICE_ID	0xC2D1
SILENT_MODE	1
PORT_OWNER	0

3.3 NVIDIA nmlx5 ESXi Driver Counters

3.3.1 Introduction

This document describes the statistics counters for the NVIDIA nmlx5 native ESXi driver. It is based on the Linux "Understanding mlx5 ethtool Counters" article. Counters that are identical to Linux reuse the Linux descriptions, while ESXi-specific counters are described based on the Adapters PRM or, when not available there, inferred from the driver source.

3.3.1.1 Counter Types

- Informative – Counts traffic and is used for load estimation and debugging.
- Acceleration [A] – Counts traffic accelerated by the driver or hardware. These counters are counted in addition to the corresponding informative counter.
- Error – An increment may indicate an issue.

3.3.1.2 Interfaces

The Interfaces column indicates where each counter can be observed:

- Uplink – The counter value contributes to a field shown by **`esxcli network nic stats get -n vmnicX`**. The counter name itself is not printed there; its value is aggregated.
- Private – Per-NIC private statistics, which provide the main human-readable counter dump. Command: **`localcli --plugin-dir /usr/lib/vmware/esxcli/int networkinternal nic privstats get -n vmnicX`**
- nmlxcli – The **`nmlxcli`** management CLI: **`-r/--ring-stats-q/--queue-stats-F/--flows-stat`**
- RDMA – RDMA device statistics exposed through the vmkernel RDMA stats interface. Command: **`esxcli rdma device stats get -d vmrdmaX`**

3.3.1.3 Other Markers

- [A] – Acceleration counter.

- [SW-inferred] – Driver software counter with no PRM or Linux reference; the description is inferred from the source code.

3.3.1.4 Naming Conventions

- Ring/channel RX and TX counters are printed as: `RX<i><field> / TX<i><field>`
- Software Port, vPort, Physical Port, and PCIe counters use the field name directly.
- Per-priority counters use priority index `p` (`0–7`); PFC counters use `%s = global` or `prio<p>` .

3.3.1.5 Ring / Channel Counters

Per-ring/per-channel counters. RX counters are collected from the receive queue (RQ), and TX counters are collected from the send queue (SQ).

3.3.1.5.1 Ring RX Counters (`RX[i]_*`)

Counter	Description	Type	Interfaces
`RX[i]_packets`	Number of packets received on ring `i` .	Informative	Private, nmlxcli
`RX[i]_bytes`	Number of bytes received on ring `i` .	Informative	Private, nmlxcli
`RX[i]_csumComplete`	Packets received with `CHECKSUM_COMPLETE` on ring `i` [A].	Acceleration	Private, nmlxcli
`RX[i]_csumUnnecessaryInner`	Packets received with inner encapsulation and `CHECKSUM_UNNECESSARY` on ring `i` [A].	Acceleration	Private, nmlxcli
`RX[i]_csumNone`	Packets received without hardware checksum acceleration on ring `i` [A].	Acceleration	Private, nmlxcli
`RX[i]_wqeErr`	Number of incorrect opcodes received on ring `i` .	Error	Private, nmlxcli
`RX[i]_buffAllocErr`	Number of failures to allocate a buffer for received packets on ring `i` .	Error	Private, nmlxcli
`RX[i]_encap`	Number of received packets carrying inner tunnel encapsulation on ring `i` . [SW-inferred]	Acceleration	Private, nmlxcli
`RX[i]_dropped`	Number of packets dropped by the driver receive path on ring `i` . [SW-inferred]	Error	Private, nmlxcli
`RX[i]_outOfBuffer`	Number of times the receive queue on ring `i` had no software buffers available for incoming adapter traffic.	Error	Private, nmlxcli, Uplink

3.3.1.5.2 Ring TX Counters (`TX[i]_*`)

Counter	Description	Type	Interfaces
`TX[i]_packets`	Number of packets transmitted on ring `i`.	Informative	Private, nmlxcli
`TX[i]_bytes`	Number of bytes transmitted on ring `i`.	Informative	Private, nmlxcli
`TX[i]_tsoPackets`	Number of TSO packets transmitted on ring `i` [A].	Acceleration	Private, nmlxcli
`TX[i]_tsoBytes`	Number of TSO bytes transmitted on ring `i` [A].	Acceleration	Private, nmlxcli
`TX[i]_tsoInnerPackets`	Number of TSO packets with inner encapsulation transmitted on ring `i` [A].	Acceleration	Private, nmlxcli
`TX[i]_tsoInnerBytes`	Number of TSO bytes with inner encapsulation transmitted on ring `i` [A].	Acceleration	Private, nmlxcli
`TX[i]_csumPartialInner`	Packets transmitted with inner encapsulation and `CHECKSUM_PARTIAL` on ring `i` [A].	Acceleration	Private, nmlxcli
`TX[i]_nop`	Number of NOP WQEs inserted into the SQ of ring `i` when reaching the end of the cyclic buffer. This is a normal condition.	Informative	Private, nmlxcli
`TX[i]_encap`	Number of transmitted packets that were tunnel-encapsulated on ring `i`. [SW-inferred]	Acceleration	Private, nmlxcli
`TX[i]_csumNone`	Packets transmitted without hardware checksum acceleration on ring `i`.	Informative	Private, nmlxcli
`TX[i]_stopped`	Number of events where the SQ on ring `i` became full. If this increases, check the number of buffers allocated for transmission.	Error	Private, nmlxcli
`TX[i]_wake`	Number of events where the SQ on ring `i` transitioned from full to not full.	Error	Private, nmlxcli
`TX[i]_dropped`	Number of transmitted packets dropped on ring `i`, for example due to DMA mapping failure. If this increases, check the number of buffers allocated for transmission.	Error	Private, nmlxcli
`TX[i]_xmitMore`	Number of packets sent with the `xmit_more` indication set, meaning no doorbell was issued, on ring `i` [A].	Acceleration	Private, nmlxcli
`TX[i]_mBufChainsHandled`	ENS: Number of mbuf packet chains handled by the SQ on ring `i`. [SW-inferred]	Informative	Private, nmlxcli
`TX[i]_mBufSGsHandled`	ENS: Number of mbuf scatter-gather elements handled by the SQ on ring `i`. [SW-inferred]	Informative	Private, nmlxcli

3.3.1.6 Software Port Counters

Software aggregation of per-ring counters, along with additional driver-level software counters.

Counter	Description	Type	Interfaces
`rxSwPackets`	Number of packets received, aggregated in software across all rings.	Informative	Private, Uplink
`rxSwBytes`	Number of bytes received, aggregated in software across all rings.	Informative	Private, Uplink
`txSwPackets`	Number of packets transmitted, aggregated in software across all rings.	Informative	Private, Uplink
`txSwBytes`	Number of bytes transmitted, aggregated in software across all rings.	Informative	Private
`txSwTsoPackets`	Number of TSO packets transmitted [A].	Acceleration	Private
`txSwTsoBytes`	Number of TSO bytes transmitted [A].	Acceleration	Private
`txSwTsoInnerPackets`	Number of TSO packets with inner encapsulation transmitted [A].	Acceleration	Private
`txSwTsoInnerBytes`	Number of TSO bytes with inner encapsulation transmitted [A].	Acceleration	Private
`rxSwCsumUnnecessary`	Packets received with `CHECKSUM_UNNECESSARY` [A].	Acceleration	Private
`rxSwCsumNone`	Packets received without hardware checksum acceleration [A].	Acceleration	Private
`rxSwCsumComplete`	Packets received with `CHECKSUM_COMPLETE` [A].	Acceleration	Private
`rxSwCsumUnnecessaryInner`	Packets received with inner encapsulation and `CHECKSUM_UNNECESSARY` [A].	Acceleration	Private
`txSwCsumPartial`	Packets transmitted with `CHECKSUM_PARTIAL` [A].	Acceleration	Private
`txSwCsumPartialInner`	Packets transmitted with inner encapsulation and `CHECKSUM_PARTIAL` [A].	Acceleration	Private
`txSwQueueStopped`	Number of events where an SQ became full. If this increases, check the number of buffers allocated for transmission.	Error	Private
`txSwQueueWake`	Number of events where an SQ transitioned from full to not full.	Error	Private
`txSwQueueDropped`	Number of transmitted packets dropped, aggregated across all rings.	Error	Private, Uplink

Counter	Description	Type	Interfaces
txSwXmitMore	Number of packets sent with the xmit_more indication set, meaning no doorbell was issued [A].	Acceleration	Private
rxSwWqeErr	Number of incorrect opcodes received, aggregated across all rings.	Error	Private, Uplink
rxSwBuffAllocErr	Number of failures to allocate a buffer for received packets, aggregated across all rings.	Error	Private
linkDownEventsPhy	Number of times the link operational state changed to down. If this increases, it may indicate port flapping; consider replacing the cable or transceiver.	Error	Private

3.3.1.7 vPort Counters

Counters on the eSwitch port connected to the vNIC. These include Ethernet and Raw Ethernet traffic, with separate counters for RDMA/RoCE traffic.

Counter	Description	Type	Interfaces
rxVportUnicastPackets	Unicast packets received and steered to a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private
rxVportUnicastBytes	Unicast bytes received and steered to a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private
txVportUnicastPackets	Unicast packets transmitted and steered from a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private
txVportUnicastBytes	Unicast bytes transmitted and steered from a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private, Uplink
rxVportMulticastPackets	Multicast packets received and steered to a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private, Uplink
rxVportMulticastBytes	Multicast bytes received and steered to a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private
txVportMulticastPackets	Multicast packets transmitted and steered from a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private, Uplink
txVportMulticastBytes	Multicast bytes transmitted and steered from a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private, Uplink
rxVportBroadcastPackets	Broadcast packets received and steered to a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private, Uplink
rxVportBroadcastBytes	Broadcast bytes received and steered to a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private

Counter	Description	Type	Interfaces
txVportBroadcastPackets	Broadcast packets transmitted and steered from a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private, Uplink
txVportBroadcastBytes	Broadcast bytes transmitted and steered from a port, including Raw Ethernet QP traffic and excluding RDMA traffic.	Informative	Private, Uplink
rxVportRdmaUnicastPackets	RDMA unicast packets received and steered to a port, including RoCE/UD/RC traffic [A].	Acceleration	Private
rxVportRdmaUnicastBytes	RDMA unicast bytes received and steered to a port, including RoCE/UD/RC traffic [A].	Acceleration	Private
txVportRdmaUnicastPackets	RDMA unicast packets transmitted and steered from a port, including RoCE/UD/RC traffic [A].	Acceleration	Private
txVportRdmaUnicastBytes	RDMA unicast bytes transmitted and steered from a port, including RoCE/UD/RC traffic [A].	Acceleration	Private
rxVportRdmaMulticastPackets	RDMA multicast packets received and steered to a port, including RoCE/UD/RC traffic [A].	Acceleration	Private
rxVportRdmaMulticastBytes	RDMA multicast bytes received and steered to a port, including RoCE/UD/RC traffic [A].	Acceleration	Private
txVportRdmaMulticastPackets	RDMA multicast packets transmitted and steered from a port, including RoCE/UD/RC traffic [A].	Acceleration	Private
txVportRdmaMulticastBytes	RDMA multicast bytes transmitted and steered from a port, including RoCE/UD/RC traffic [A].	Acceleration	Private

3.3.1.8 Physical Port Counters

Counters on the external port that connects the adapter to the network. These counters are not exposed to virtual machines.

3.3.1.8.1 IEEE 802.3 Counters

Counter	Description	Type	Interfaces
txPacketsPhy	Number of packets transmitted on the physical port.	Informative	Private
rxPacketsPhy	Number of packets received on the physical port. Excludes packets discarded due to FCS, frame size, and similar errors.	Informative	Private
rxCrcErrorsPhy	Number of received packets dropped due to FCS errors on the physical port. If this increases at a high rate, check link quality using rxSymbolErrorsPhy and rxCorrectedBitsPhy.	Error	Private, Uplink

Counter	Description	Type	Interfaces
txBytesPhy	Number of bytes transmitted on the physical port.	Informative	Private
rxBytesPhy	Number of bytes received on the physical port, including the Ethernet header and FCS.	Informative	Private
txMulticastPhy	Number of multicast packets transmitted on the physical port.	Informative	Private
txBroadcastPhy	Number of broadcast packets transmitted on the physical port.	Informative	Private
rxMulticastPhy	Number of multicast packets received on the physical port.	Informative	Private
rxBroadcastPhy	Number of broadcast packets received on the physical port.	Informative	Private
rxInRangeLenErrorsPhy	Number of received packets dropped due to length/type errors on the physical port.	Error	Private, Uplink
rxOutOfRangeLenPhy	Number of received packets dropped because their length exceeded the allowed range. If this increases, the peer may be using a larger MTU; align MTU settings.	Error	Private, Uplink
rxOversizePktsPhy	Number of received packets dropped because their length exceeded the MTU. If this increases, the peer may be using a larger MTU; align MTU settings.	Error	Private, Uplink
rxSymbolErrPhy	Number of received packets dropped due to physical coding, or symbol, errors on the physical port.	Error	Private, Uplink
txMacControlPhy	Number of MAC control packets transmitted on the physical port.	Informative	Private
rxMacControlPhy	Number of MAC control packets received on the physical port.	Informative	Private
rxUnsupportedOpPhy	Number of MAC control packets received with an unsupported opcode.	Error	Private
rxPauseCtrlPhy	Number of link-layer pause packets received. If this increases, the network may be congested.	Informative	Private
txPauseCtrlPhy	Number of link-layer pause packets transmitted. If this increases, the NIC may be congested.	Informative	Private

3.3.1.8.2 RFC 2863 Counters

Counter	Description	Type	Interfaces
rxDiscardsPhy	Number of received packets dropped due to lack of buffers. If this increases, the adapter is congested and cannot absorb traffic from the network.	Error	Private
txDiscardsPhy	Number of packets discarded during transmission even though no errors were detected, such as link down, head-of-line drop, or pause conditions.	Error	Private, Uplink
txErrorsPhy	Number of transmitted packets dropped because their length exceeded the MTU.	Error	Private

3.3.1.8.3 RFC 2819 Counters

Counter	Description	Type	Interfaces
rxUndersizePktsPhy	Received packets dropped because their length was less than 64 bytes. If this increases, the peer may have a non-standard MTU or may be sending malformed packets.	Error	Private
rxFragmentsPhy	Received packets dropped because their length was less than 64 bytes and they had an FCS error. If this increases, the peer may have a non-standard MTU.	Error	Private
rxJabbersPhy	Received packets dropped because their length was greater than 64 bytes and they had an FCS error.	Error	Private
rx_64_bytesPhy	Packets received on the physical port with a size of 64 bytes.	Informative	Private
rx_65_to_127_bytesPhy	Packets received with a size of 65 to 127 bytes.	Informative	Private
rx_128_to_255_bytesPhy	Packets received with a size of 128 to 255 bytes.	Informative	Private
rx_256_to_511_bytesPhy	Packets received with a size of 256 to 511 bytes.	Informative	Private
rx_512_to_1023_bytesPhy	Packets received with a size of 512 to 1023 bytes.	Informative	Private
rx_1024_to_1518_bytesPhy	Packets received with a size of 1024 to 1518 bytes.	Informative	Private

Counter	Description	Type	Interfaces
rx_1519_to_2047_bytesPhy	Packets received with a size of 1519 to 2047 bytes.	Informative	Private
rx_2048_to_4095_bytesPhy	Packets received with a size of 2048 to 4095 bytes.	Informative	Private
rx_4096_to_8191_bytesPhy	Packets received with a size of 4096 to 8191 bytes.	Informative	Private
rx_8192_to_10239_bytesPhy	Packets received with a size of 8192 to 10239 bytes.	Informative	Private

3.3.1.8.4 PHY Statistical Counters

Counter	Description	Type	Interfaces
rxSymbolErrorsPhy	Symbol errors that were not corrected by FEC, or occurred when FEC was not active. Equivalent to Linux rx_pcs_symbol_err_phy. If this increases, the link may be experiencing high BER and traffic loss; consider replacing the cable or transceiver.	Error	Private
rxCorrectedBitsPhy	Number of bits corrected on this port by the active FEC mode, RS or FC. If this increases, the link may be experiencing high BER.	Error	Private
rxErrLane_0_Phy	Physical raw errors on lane 0 before FEC correction. If this increases, the link may be experiencing high BER and traffic loss; review together with rxCorrectedBitsPhy.	Error	Private
rxErrLane_1_Phy	Physical raw errors on lane 1 before FEC correction. See rxErrLane_0_Phy.	Error	Private
rxErrLane_2_Phy	Physical raw errors on lane 2 before FEC correction. See rxErrLane_0_Phy.	Error	Private
rxErrLane_3_Phy	Physical raw errors on lane 3 before FEC correction. See rxErrLane_0_Phy.	Error	Private

3.3.1.8.5 Ethernet Extended Counters

Counter	Description	Type	Interfaces
rxBufferPassedThresPhy	Number of events where the port receive buffer exceeded 85% utilization.	Informative	Private

3.3.1.9 Priority Port Counters

Physical port counters reported per L2 priority, where `p` is the priority index `0–7`.

Counter	Description	Type	Interfaces
`rxPrio[p]_bytes`	Bytes received with priority `p` on the physical port. Also counts packets without VLAN.	Informative	Private
`rxPrio[p]_packets`	Packets received with priority `p` on the physical port. Also counts packets without VLAN.	Informative	Private
`txPrio[p]_bytes`	Bytes transmitted with priority `p` on the physical port.	Informative	Private
`txPrio[p]_packets`	Packets transmitted with priority `p` on the physical port.	Informative	Private
`rxPrio[p]_buf_discard`	Packets discarded by the device due to lack of per-host receive buffers on priority `p`.	Informative	Private
`rxPrio[p]_cong_discard`	Packets discarded by the device due to per-host congestion, such as WRED, on priority `p`.	Informative	Private
`rxPrio[p]_marked`	Packets ECN-marked by the device due to per-host congestion on priority `p`.	Informative	Private

3.3.1.9.1 PFC Counters

`<scope>` can be either `global` or `prio<p>`.

Counter	Description	Type	Interfaces
`rx_<scope>_pause`	Pause packets received for the specified scope or priority. Available only if PFC is enabled on that priority. If this increases, the network may be congested.	Informative	Private
`rx_<scope>_pause_duration`	Duration, in microseconds, of received pause frames for the specified scope or priority. This represents the time during which the port did not transmit traffic. PFC must be enabled.	Informative	Private
`tx_<scope>_pause`	Pause packets transmitted for the specified scope or priority. PFC must be enabled. If this increases, the adapter may be congested.	Informative	Private
`tx_<scope>_pause_duration`	Duration, in microseconds, of transmitted pause frames for the specified scope or priority. PFC must be enabled.	Informative	Private
`rx_<scope>_pause_transition`	Number of Xoff-to-Xon transitions for the specified scope or priority. PFC must be enabled.	Informative	Private

3.3.1.9.2 PFC Stall Counters

Counter	Description	Type	Interfaces
`txPauseStormWarningEvents`	Number of times the device transmitted pause frames for an extended period.	Informative	Private
`txPauseStormErrorEvents`	Number of times the device transmitted pause frames for an extended period, reached a timeout, and disabled pause-frame transmission. Drops may occur while pause-frame transmission is disabled.	Error	Private

3.3.1.10 PCIe / Device Counters

3.3.1.10.1 PCIe Performance Counters

Counter	Description	Type	Interfaces
`rxPciSignalIntegrity`	Physical-layer PCIe signal-integrity errors, such as recovery transitions caused by framing or CRC errors. If this increases, try a different slot and ensure the latest firmware and BIOS are installed.	Error	Private
`txPciSignalIntegrity`	Physical-layer PCIe signal-integrity errors caused by recovery transitions initiated by the link partner. If this increases, try a different slot and ensure the latest firmware and BIOS are installed.	Error	Private
`outboundPciBufferOverflow`	Number of packets dropped due to lack of PCIe buffers on the receive path from the NIC port to the host. If this increases at a high rate, the host receive rate may exceed PCIe capacity.	Informative	Private
`outboundPciStalledRd`	Percentage of time during the last second that the NIC had outbound non-posted read requests but could not perform them due to insufficient non-posted credits.	Informative	Private
`outboundPciStalledWr`	Percentage of time during the last second that the NIC had outbound posted write requests but could not perform them due to insufficient posted credits.	Informative	Private
`outboundPciStalledRdEvents`	Number of events where `outboundPciStalledRd` exceeded the configured threshold.	Informative	Private
`outboundPciStalledWrEvents`	Number of events where `outboundPciStalledWr` exceeded the configured threshold.	Informative	Private

3.3.1.10.2 PCIe Timers-and-States (TAS) Counters

Counter	Description	Type	Interfaces
`txPciTransportNonfatalMsg`	Number of PCIe Non-Fatal error messages sent.	Error	Private
`txPciTransportFatalMsg`	Number of PCIe Fatal error messages sent.	Error	Private

3.3.1.11 Queue Counters

Counter	Description	Type	Interfaces
`outOfBuffer`	Number of drops caused by lack of WQEs for the associated QPs/RQs.	Error	Private
`downOutOfBuffer`	Out-of-buffer drops counted while the interface was down. [SW-inferred]	Error	Private

3.3.1.12 vNIC Environment Counters

vNIC-level environment and error counters reported by firmware through `QUERY_VNIC_ENV` / vNIC Diagnostic Statistics. Each counter is reported only when firmware marks it as valid.

Counter	Description	Type	Interfaces
`currentQUnderProcessorHandle`	Current number of queues handled through the exception flow due to an asynchronous error or errored command.	Informative	Private
`totalQUnderProcessorHandle`	Total number of queues handled through the exception flow due to an asynchronous error or errored command.	Informative	Private
`sendQueuePriorityUpdateFlow`	Total number of QP/SQ priority or SL update events.	Informative	Private
`nicReceiveSteeringDiscard`	Number of packets that completed NIC receive flow-table steering but were discarded because they did not match any flow in the final flow table. Equivalent to Linux `rx_steer_missed_packets`.	Error	Private
`receiveDiscardVportDown`	Number of packets steered to a vPort and discarded because the vPort was not in a receive-capable state.	Error	Private
`transmitDiscardVportDown`	Number of packets transmitted by a vNIC and discarded because the vPort was not in a transmit-capable state.	Error	Private
`asyncEqOverrun`	Number of times an EQ mapped to asynchronous events was overrun. This is not necessarily reset during driver initialization.	Error	Private

Counter	Description	Type	Interfaces
`compEqOverrun`	Number of times an EQ mapped to completion events was overrun. This is not necessarily reset during driver initialization.	Error	Private
`nicReceiveSteeringDepth`	Current evaluation of the longest steering path a packet may traverse through the NIC receive steering table, in device-specific units.	Informative	Private
`invalidCommand`	Number of commands issued by the VF that failed for reasons other than quota exceeded.	Error	Private
`quotaExceededCommand`	Number of commands issued by the VF that failed due to quota limits.	Error	Private
`internalRqOutOfBuffer`	Number of packets dropped due to lack of receive WQEs for internal device RQs.	Error	Private
`cqOverrun`	Number of times CQs entered an error state due to overflow, when the device attempted to post a CQE into a full CQ buffer.	Error	Private
`ethWqeTooSmall`	Total number of packets reaching an Ethernet RQ that were too large to fit in the WQE.	Error	Private
`fwCpuUsage`	Number of firmware CPU cycles used by the vNIC. This can help detect excessive resource usage.	Informative	Private
`errorsHandledByFw`	Number of errors requiring firmware handling for a vNIC. This can help detect excessive resource usage.	Informative	Private
`fwCpuTimestamp`	Free-running firmware CPU timestamp used to calculate the usage percentage of `fwCpuUsage`.	Informative	Private
`generatedPktSteeringfail`	Number of packets generated by the vNIC that experienced an unexpected steering failure at any point in the steering flow.	Error	Private
`handledPktSteeringFail`	Number of packets handled by the vNIC that experienced an unexpected steering failure in the steering flow owned by the vNIC, including the FDB for the eSwitch owner.	Error	Private

3.3.1.13 Temperature Sensor Counters

Counter	Description	Type	Interfaces
`asicSensorTemperature`	Current ASIC temperature in degrees Celsius, reported by the MTMP virtual ASIC sensor, `sensor_index 0`, representing the maximum temperature across all real ASIC diodes.	Informative	Private

3.3.1.14 VMware-Specific Counters

Counter	Description	Type	Interfaces
watchdogReset	Number of times the driver TX watchdog fired and triggered a queue or device reset. If this increases, the transmit path may be hanging; check firmware and link state. [SW-inferred]	Error	Private

3.3.1.15 Standard Uplink Stats (vmk_UplinkStats)

These counters are displayed by: esxcli network nic stats get -n vmnicX

They are fixed generic fields, each derived from one or more counters listed above. The Source column describes the mapping or formula.

Counter	Description	Source	Type	Interfaces
rxPkts	Packets received.	rxSwPackets	Informative	Uplink
rxBytes	Bytes received.	rxSwBytes	Informative	Uplink
txPkts	Packets transmitted.	txSwPackets	Informative	Uplink
txBytes	Bytes transmitted.	txVportUnicastBytes + txVportMulticastBytes + txVportBroadcastBytes	Informative	Uplink
rxDrops	Received packets dropped.	rxSwWqeErr	Error	Uplink
txDrops	Transmitted packets dropped.	txSwQueueDropped	Error	Uplink
rxMissErrors	Receive misses due to lack of buffers.	Sum of per-channel RX[i]_outOfBuffer	Error	Uplink
rxLengthErrors	Receive length errors.	rxInRangeLenErrorsPhy + rxOutOfRangeLenPhy + rxOversizePktsPhy	Error	Uplink
rxCRCErrors	Receive CRC/FCS errors.	rxCrcErrorsPhy	Error	Uplink
rxFrameAlignErrors	Receive frame alignment errors.	802.3 aAlignmentErrors , no dedicated named counter	Error	Uplink
txAbortedErrors	Transmit aborted errors.	txDiscardsPhy	Error	Uplink
txCarrierErrors	Transmit carrier errors.	rxSymbolErrPhy	Error	Uplink

Counter	Description	Source	Type	Interfaces
`rxErrors`	Total receive errors.	`rxLengthErrors + rxCRCErrors + rxFrameAlignErrors`	Error	Uplink
`txErrors`	Total transmit errors.	`txAbortedErrors + txCarrierErrors`	Error	Uplink
`rxMulticastPkts`	Multicast packets received.	`rxVportMulticastPackets`	Informative	Uplink
`rxBroadcastPkts`	Broadcast packets received.	`rxVportBroadcastPackets`	Informative	Uplink
`txMulticastPkts`	Multicast packets transmitted.	`txVportMulticastPackets`	Informative	Uplink
`txBroadcastPkts`	Broadcast packets transmitted.	`txVportBroadcastPackets`	Informative	Uplink

3.3.1.16 Flow Steering Counters

Reported by: `nmlxcli -Fnmlxcli --flows-stat`

These are per-uplink flow-resource counters. Each category reports the number of created and destroyed objects. The CLI also derives `inUse = created - destroyed`.

Counter Category	Description	Type	Interfaces
`applyedFlows.{created,destroyed}`	Number of applied flow rules created or destroyed. [SW-inferred]	Informative	nmlxcli
`encapDecap.{created,destroyed}`	Number of encap/decap flow actions created or destroyed. [SW-inferred]	Informative	nmlxcli
`tag.{created,destroyed}`	Number of tag, or flow-tag, actions created or destroyed. [SW-inferred]	Informative	nmlxcli
`metadata.{created,destroyed}`	Number of metadata set/copy actions created or destroyed. [SW-inferred]	Informative	nmlxcli
`ct.{created,destroyed}`	Number of connection-tracking, or CT, objects created or destroyed. [SW-inferred]	Informative	nmlxcli
`ctBulk.{created,destroyed}`	Number of CT bulk resources created or destroyed. [SW-inferred]	Informative	nmlxcli
`countersBulk.{created,destroyed}`	Number of flow-counter bulk resources created or destroyed. [SW-inferred]	Informative	nmlxcli

3.3.1.17 RDMA Counters

These counters are reported by the RDMA device through the vmkernel RDMA statistics interface and are separate from the Ethernet interfaces described above.

3.3.1.17.1 RDMA Traffic Counters

Sourced from the per-vNIC vPort counters for RoCE/UD/RC traffic.

Counter	Description	Type	Interfaces
`rxUcastPkts`	RDMA unicast packets received, including RoCE/UD/RC traffic.	Informative	RDMA
`rxMcastPkts`	RDMA multicast packets received, including RoCE/UD/RC traffic.	Informative	RDMA
`rxPackets`	Total RDMA packets received: `rxUcastPkts + rxMcastPkts`.	Informative	RDMA
`txUcastPkts`	RDMA unicast packets transmitted, including RoCE/UD/RC traffic.	Informative	RDMA
`txMcastPkts`	RDMA multicast packets transmitted, including RoCE/UD/RC traffic.	Informative	RDMA
`txPackets`	Total RDMA packets transmitted: `txUcastPkts + txMcastPkts`.	Informative	RDMA
`rxUcastBytes`	RDMA unicast bytes received.	Informative	RDMA
`rxMcastBytes`	RDMA multicast bytes received.	Informative	RDMA
`rxBytes`	Total RDMA bytes received: `rxUcastBytes + rxMcastBytes`.	Informative	RDMA
`txUcastBytes`	RDMA unicast bytes transmitted.	Informative	RDMA
`txMcastBytes`	RDMA multicast bytes transmitted.	Informative	RDMA
`txBytes`	Total RDMA bytes transmitted: `txUcastBytes + txMcastBytes`.	Informative	RDMA
`rxErrors`	Number of received RDMA error packets, based on vPort `received_errors`.	Error	RDMA
`txErrors`	Number of transmitted RDMA error packets, based on vPort `transmit_errors`.	Error	RDMA

3.3.1.17.2 RDMA Extended Counters

IB Q-counter set bound to each RDMA QP.

Counter	Description	Type	Interfaces
rxDrops	Sum of per-port out_of_buffer events, indicating drops that occurred because the responder had no posted RQ WQE when a packet arrived.	Error	RDMA(ext)
retransCnt	Sum of per-port retransmission-cause counters: local_ack_timeout_err, rnr_nak_retry_err, duplicate_request, implied_nak_seq_err, and packet_seq_err.	Error	RDMA(ext)
txDrops	Not exposed by the Q-counter set; always reported as 0.	Error	RDMA(ext)

3.3.1.17.3 RDMA Software Counters

Driver-maintained object and event counters. [SW-inferred]

Counter	Description	Type	Interfaces
qpCnt	Current number of QPs allocated on the device.	Informative	RDMA
qpResetStateCnt	Number of QPs currently in the RESET state.	Informative	RDMA
qpInitStateCnt	Number of QPs currently in the INIT state.	Informative	RDMA
qpRtrStateCnt	Number of QPs currently in the RTR, or Ready to Receive, state.	Informative	RDMA
qpRtsStateCnt	Number of QPs currently in the RTS, or Ready to Send, state.	Informative	RDMA
qpSqdStateCnt	Number of QPs currently in the SQD, or Send Queue Drained, state.	Informative	RDMA
qpSqeStateCnt	Number of QPs currently in the SQE, or Send Queue Error, state.	Informative	RDMA
qpErrStateCnt	Number of QPs currently in the ERROR state.	Error	RDMA
qpEventCnt	Number of QP asynchronous events delivered.	Informative	RDMA
cqCnt	Current number of CQs allocated on the device.	Informative	RDMA
cqEventCnt	Number of CQ asynchronous events delivered.	Informative	RDMA
srqCnt	Current number of SRQs allocated, when SRQ is supported.	Informative	RDMA
srqEventCnt	Number of SRQ asynchronous events delivered, when SRQ is supported.	Informative	RDMA
pdCnt	Current number of Protection Domains allocated.	Informative	RDMA

Counter	Description	Type	Interfaces
mrCnt	Current number of Memory Regions allocated.	Informative	RDMA
ahCnt	Current number of Address Handles allocated.	Informative	RDMA

3.4 Hardware Large Receive Offload (HW LRO)

3.4.1 Overview

Large Receive Offload (LRO) reduces the CPU cost of receiving TCP traffic. Instead of passing every small TCP segment to the ESXi network stack, the NVIDIA adapter merges consecutive in-order segments of the same TCP connection into one large packet (up to about 64 KB). The network stack then handles one large packet instead of many small ones.

With hardware LRO, the merging is done by the adapter itself, so it uses no host CPU. The `nmlx5_core` driver then updates the IP and TCP headers of the merged packet and passes it to ESXi.

Typical benefits:

- Lower CPU use per received byte.
- Higher receive throughput for large TCP transfers (for example, vMotion, storage, and bulk data traffic).

3.4.1.1 HW LRO and ENS Software LRO

From ESXi 9.0, the Enhanced Network Stack (ENS, also called UENS) is the default networking stack. ENS has its own software LRO. When the driver reports HW LRO support, ENS turns its software LRO off for that uplink, and HW LRO is used instead.

This means that traffic the adapter does not merge (see When Segments Are Not Merged and Limitations) is also not merged in software. Account for this when comparing HW LRO with software LRO, especially for encapsulated (overlay) traffic, where HW LRO performs no merging.

3.4.2 Requirements

HW LRO is used only when all of the following are true:

Requirement	Details
ESXi version	ESXi 8.0 U3 or later.
Adapter	An NVIDIA ConnectX / BlueField adapter whose firmware supports LRO. Support is detected from the adapter firmware at driver load.
Datapath mode	The uplink must run in ENS (Enhanced Data Path, EDP) mode. HW LRO is not used in the standard (non-ENS) datapath. From ESXi 9.0, ENS is the default.

Requirement	Details
Striding RQ	Striding receive queues (multi-packet receive buffers) must be supported by the adapter and not disabled by the user. See `data_path_control` below.
Traffic	TCP over IPv4 or IPv6.

If any requirement is not met, the driver does not report HW LRO to ESXi, and packets are received normally. In this case ENS software LRO can be used.

3.4.3 How It Works

1. When the uplink starts in ENS mode, the driver checks the adapter capabilities. If LRO can be used, the driver reports HW LRO support (IPv4 and IPv6) to ESXi.
2. ESXi decides which RX queues need LRO and requests them from the driver.
3. For each such queue, the driver enables LRO in the adapter for both IPv4 and IPv6 TCP, with:
    - Maximum merged packet size of about 64 KB.
    - A merge timeout set by the `lro_timeout` module parameter.
4. The adapter merges in-order segments of the same TCP connection. A connection is identified by VLAN ID, source/destination IP address, and source/destination TCP port.
5. When a merge ends, the driver updates the IP and TCP headers of the merged packet (length, checksums, TTL/Hop Limit, ACK number, TCP window, and PSH flag) and passes it to ESXi.

The default RX queue does not use HW LRO. HW LRO is applied only to RX queues that ESXi requests with LRO.

3.4.4 When Segments Are Not Merged

The adapter merges only "clean", in-order TCP data segments. A segment is not merged (and an open merge is closed) in cases such as:

- The segment is out of order (unexpected TCP sequence number).
- The segment has one of the TCP flags SYN, FIN, RST, URG, or similar control flags set.
- The segment has TCP options other than the timestamp option, has IPv4 options, or has IPv6 extension headers.
- The segment is an IP fragment, or has a bad IP/TCP checksum.
- The TCP payload is smaller than the minimum segment size supported by the adapter. Pure ACKs (no payload) are never merged.
- The segment has the PSH flag set, and the adapter does not support merging PSH segments.
- The segment has the TCP timestamp option, and the adapter does not support merging timestamped segments. When supported, timestamp use must be the same in all merged segments, and timestamp values must not go backwards.
- ECN bits or other IP header fields differ from the previous segments.
- Adding the segment would exceed the maximum merged size (~64 KB).
- The merge timeout expired.

- The adapter has no free resources to start a new merge. The number of merges open at the same time is limited by the adapter, so with many active TCP flows on the same queue, some flows may not be merged.

These segments are still delivered to ESXi normally; they are just not merged.

3.4.5 Limitations

- ENS (EDP) mode only. HW LRO is not available in the standard datapath.
- No tunnel (overlay) LRO. HW LRO is not used for encapsulated traffic (for example, VXLAN or Geneve). Only non-tunneled TCP flows are merged. Because ENS software LRO is off when HW LRO is reported, encapsulated TCP traffic is not merged at all.
- TCP only. UDP and other protocols are not merged.
- Default RX queue does not use HW LRO. Traffic on the default queue is not merged.
- Maximum merged packet size is about 64 KB (64 KB minus headers).
- Adapter and firmware dependent behavior. The minimum segment size, support for merging PSH and timestamped segments, the available timeout values, and the number of merges open at the same time all depend on the adapter and its firmware. Results can differ between adapter models and firmware versions.
- Small per-packet driver cost. For each merged packet, the driver updates the IP and TCP headers and their checksums. This cost is per merged packet, not per received segment.
- No dedicated on/off switch in the driver. The driver offers HW LRO whenever the requirements are met; ESXi decides which queues use it. To fully prevent HW LRO in the driver, disable striding RQ (see `data_path_control` below). Note that this also disables the other benefits of striding RQ.

3.4.6 Configuration

3.4.6.1 `lro_timeout` (module parameter of `nmlx5_core`)

Sets how long the adapter waits for more segments before it closes a merge and delivers the merged packet. The value is an index (0-3) into four timeout values defined in the adapter firmware. With the default firmware settings, the indexes are:

Value	Timeout
0	8 µs
1	16 µs
2	32 µs (default)
3	1024 µs

- The parameter is an array: one value per port, in port order, separated by commas.
- Values above 3 are rejected and the module load fails.
- Ports without a value use the default (2).
- The actual timeout can vary between the configured value and twice that value.

Trade-off:

- A **longer** timeout merges more segments (less CPU, higher throughput) but can add receive latency.
- A **shorter** timeout lowers latency but merges fewer segments.

Example: use 16 µs on port 1 and 1024 µs on port 2:

```
esxcli system module parameters set -m nmlx5_core -p "lro_timeout=1,3"
```

A host reboot (or driver reload) is required for the change to take effect.

To view the current setting:

```
esxcli system module parameters list -m nmlx5_core | grep lro_timeout
```

3.4.6.2 `data_path_control` (module parameter of `nmlx5_core`)

Bit 0 of this parameter disables striding RQ. Because HW LRO needs striding RQ, setting this bit also disables HW LRO in the driver.

```
esxcli system module parameters set -m nmlx5_core -p "data_path_control=1"
```

A host reboot (or driver reload) is required. Set it back to **`0`** (default) to re-enable striding RQ and HW LRO.

3.4.7 Verifying HW LRO

3.4.7.1 Driver log

At uplink start, the driver writes to **`/var/log/vmkernel.log`** whether LRO is supported:

```
... Driver will use <N> sized multi packet wqe for uplink rx rings ...
... LRO is  supported
```

or:

```
... LRO is not supported
```

If you see **`Driver will use <N> sized single packet wqe for uplink rx rings`**, striding RQ is not in use, so HW LRO is not available.

When ESXi requests an RX queue with LRO, the driver logs:

```
... channel <N> configured with LRO
```

3.4.7.2 Statistics

The driver exposes per-RX-ring LRO counters in the uplink private statistics (for example, through `vsish -e get /net/pNics/vmnicX/stats`). The counters are shown for each ring as `RX[<ring>].<counter>`:

Counter	Meaning
`lroPackets`	Number of merged (LRO) packets passed to ESXi.
`lroSegments`	Total number of TCP segments merged into those packets.
`lroBytes`	Total bytes in merged packets.
`lroAbrtSz`	Merges closed because the maximum size was reached.
`lroAbrtTo`	Merges closed because the timeout expired.
`lroAbrtEx`	Merges closed because of an exception (for example, a segment that could not be merged).
`lroAbrtOt`	Merges closed for other reasons.

Useful checks:

- `lroSegments / lroPackets` is the average number of segments merged per packet. Higher means better LRO efficiency.
- Many `lroAbrtTo` with a low average may mean the timeout is too short for the traffic pattern. Consider a larger `lro_timeout`.
- Many `lroAbrtSz` means merges usually reach the maximum size, which is the best case for throughput.

3.4.8 Troubleshooting

Symptom	Possible cause / action
`LRO is not supported` in the log	The adapter firmware does not support LRO. Update the firmware.
`single packet wqe` in the log, no LRO message	Uplink is not in ENS mode, striding RQ is not supported, or `data_path_control` bit 0 is set.
LRO is supported but `lroPackets` stays 0	ESXi did not request LRO on the active queues, traffic is on the default queue, traffic is tunneled (VXLAN/Geneve), or the traffic is not suitable for merging (small segments, out of order, many control flags).
Lower merge rate than expected with many TCP flows	The adapter limits how many merges can be open at the same time. Spread flows across more RX queues (RSS), or check `lroAbrtTo` and consider a larger `lro_timeout`.
Encapsulated traffic gets less benefit than with software LRO	Expected. HW LRO does not merge encapsulated traffic, and ENS software LRO is off while HW LRO is reported.
Higher latency for small request/response traffic	Lower `lro_timeout` (for example, to 0 or 1). A host reboot (or driver reload) is required for the change to take effect.

Symptom	Possible cause / action
Module fails to load after changing `lro_timeout`	A value above 3 was used, or more values than ports were given. Fix the parameter.

4 Document Revision History

Date	Revision	Description of Change
November 2026	1.0	• Added section Hardware Large Receive Offload (HW LRO) • Updated section Installation
August 2026		Initial version of the document

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. Neither NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: “NVIDIA”) make any representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.



Trademarks
NVIDIA, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of NVIDIA Corporation and/or its affiliates in the U.S. and in other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright
© 2026 NVIDIA Corporation & affiliates. All Rights Reserved.

NVIDIA